EFFICIENT POLICY SPACE RESPONSE ORACLES

Anonymous authors
Paper under double-blind review

ABSTRACT

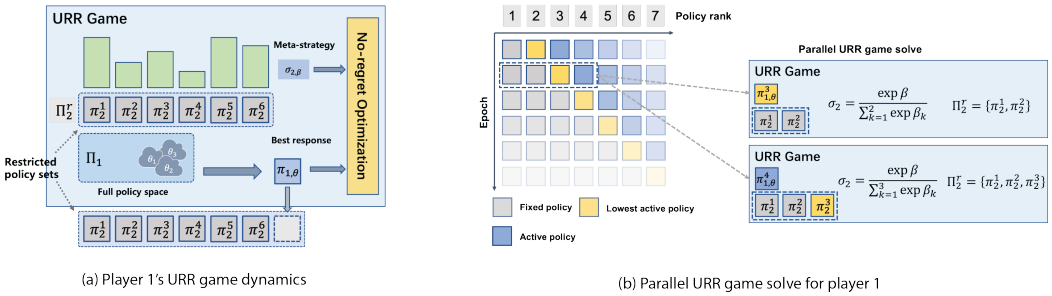
Policy Space Response Oracle methods (PSRO) provide a general solution to approximate Nash equilibrium in two-player zero-sum games but suffer from two drawbacks: (1) the *computational inefficiency* due to consistent meta-game evaluation via simulations, and (2) the *exploration inefficiency* due to learning best responses against fixed meta-strategies. In this work, we propose Efficient PSRO (EPSRO) that considerably improves the efficiency of the above two steps. Central to our development is the novel subroutine of *no-regret optimization* on solving *unrestricted-restricted (URR)* games. By modeling the EPSRO as URR game solving, one can compute the best responses and meta-strategies in a single forward pass without extra simulations. Theoretically, we prove that the proposed optimization procedures of EPSRO guarantee the monotonic improvement on the exploitability, which is absent in existing researches of PSRO. Furthermore, we prove that the no-regret optimization has a regret bound of $\mathcal{O}(\sqrt{T \log [(k^2+k)/2]})$, where k the size of restricted policy set. The pipeline of EPSRO is highly parallelized, making policy-space exploration more affordable in practice and thus more behavioral diversity. Empirical evaluations on various games report that EPSRO achieves a 50x speedup in wall-time and 2.5x data efficiency while obtaining comparable exploitability against existing PSRO methods.

1 INTRODUCTION

Policy Space Response Oracles (PSRO) (Lanctot et al., 2017) is a general reinforcement learning algorithm for zero-sum games, which has been applied in many non-trivial multi-agent learning tasks (Vinyals et al., 2019; Liu et al., 2021b). In general, PSRO approximates a Nash equilibrium (NE) by iteratively expanding a restricted game formed by a set of restricted policy sets, which is ideally much smaller than the original game. To achieve that, PSRO performs meta-game solving to produce a tuple of meta-strategies as an approximate NE; and learns best responses against the meta-strategies at each epoch.

Although PSRO can solve the original multi-agent learning problem in restricted games, it is still computationally expensive to solve meta-game and learn high-quality best responses. Specifically, solving a geometrically growing meta-game relies on numerous simulations across the Cartesian space of growing policy sets (Omidshafiei et al., 2019; Yang et al., 2020), which results in *computational inefficiency*; learning best responses against fixed opponent meta-strategies lacks adequate opponent exploration, which results in slow convergence, i.e., *exploration inefficiency*. An expected property for PSRO is the non-degenerate guarantee for best response learning, as the expanded policy sets towards a closer approximation of NE in this case. However, the exploration inefficiency empirically hinders the generation of high-quality responses under guarantee (Wang et al., 2021) even though the PSRO can substantially expand the policy sets. In the worst case, PSRO needs to add all feasible policies from the original game and thus generates large policy sets, making it more computational expensive and slow to converge (McAleer et al., 2020). A line of work to resolve the above issues is to utilize parallelism for best responses learning (Lanctot et al., 2017; Balduzzi et al., 2019; McAleer et al., 2020; Liu et al., 2021a), i.e., a kind of pre-training to improve the quality of best responses. However, they learn best responses still rely on numerous simulations and playing against fixed meta-strategies.

It is desirable for an efficient method to these problems. Our key insight is that the computation of meta-strategies can be free from dedicated simulations, and the learning of best responses should have adequate opponent exploration and be toward monotonic expansion on restricted policy sets.



(a) Player 1's URR game dynamics

(b) Parallel URR game solve for player 1

Figure 1: An overview of EPSRO. (a) EPSRO runs in a loop that learns best responses by solving URR games for each player, then expands the restricted policy sets with these $\pi_{i,\theta}$ at each epoch. (b) At each epoch, EPSRO runs parallel URR game solve for each active best response $\pi^j_{i,\theta}$ (Algorithm 4), where j is the level. In each URR game, $\pi^j_{i,\theta}$ plays against $\Pi^{r,k}_{-i}$ where $k = 1, \ldots, j-1$.

Previous work has proposed to learn a generalized best response as a monotonic expansion guarantee (Zinkevich et al., 2007a; Hansen et al., 2008). However, they only focus on tabular cases in which the policy space is limited. Moreover, they require nested meta-game solving to compute generalized best responses, resulting in high computational costs.

In this paper, we propose the Efficient PSRO (EPSRO) and formulate it as *unrestricted-restricted* (URR) games to solve the above two kinds of inefficiency. Compared to existing PSRO methods, EPSRO does not require any dedicated simulation to solve a restricted meta-game beforehand. Instead, EPSRO merges the best response learning and meta-strategies computation by performing URR game solving. In this case, each best response policy and its opponent meta-strategy compose of an approximate NE, and the algorithm outputs a tuple of meta-strategies as a solution. We prove that the learned best responses are guaranteed to expand the restricted policy sets in a monotonic manner, improving exploration efficiency. To efficiently solve the URR games, we leverage no-regret optimization (Daskalakis et al., 2011) and reinforcement learning (RL) (Sutton & Barto, 2018). As the EPSRO has growing policy sets, it always breaks the convergence when new policies are added. Thus, EPSRO needs to retrain meta-strategies from scratch to satisfy such a change, resulting in high retraining costs. As a solution to this issue, we propose a warm-start technique. Moreover, we introduce a pipeline URR solver to make the best response learning be parallelized, which further improves the training efficiency of the underlying reinforcement learning step. We analyze the algorithm performance and give a regret bound of $\mathcal{O}(\sqrt{T \log[(k^2+k)/2]})$, where k is the size of the final restricted policy set. This bound implies that our algorithm has a better tolerance for the growing size of policy sets than previous work (Appendix G.1). The empirical results show that EPSRO substantially improves efficiency and performs better than existing PSRO-based methods in the high-dimensional matrix, poker, card games, and multi-agent gathering tasks.

2 Preliminaries

Two-player Normal-form Games. A two-player normal-form game (Fudenberg & Tirole, 1991) can be formalized as a tuple (Π, U^{Π}). $\Pi = (\Pi_1, \Pi_2)$ the tuple of policy sets; $U^{\Pi} = (U^{\Pi_1}, U^{\Pi_2})$ the tuple of payoff matrices. Formally, $\forall i \in \{1, 2\}, U^{\Pi_i} : \Pi \to \mathbb{R}^{|\Pi_1| \times |\Pi_2|}$, in which each item represents the utility under a joint policy or accumulative reward in reinforcement learning terminology. Players in the game try to maximize their own expected utility by sampling policy from a mixture (distribution) σ_i over their policy sets, where $\forall i \in \{1, 2\}, \sigma_i \in \Delta(\Pi_i)$. For the sake of convenience, we use $-i$ to denote the other players except for i in the following content. A best response to a mixed-strategy σ_{-i} is defined as a strategy that has highest utility. It can be expressed as $\mathbf{BR}(\sigma_{-i}) = \arg\max_{\sigma_i} u_i(\sigma_i, \sigma_{-i})$, where $u_i(\cdot, \cdot)$ denotes the utility function of player i.

Policy Space Response Oracles (PSRO). Double Oracle (DO) methods (McMahan et al., 2003; Le Cong Dinh et al., 2021; McAleer et al., 2021) provide an iterative mechanism for finding an approximation of Nash equilibrium in normal-form games. These algorithms maintain an iteratively

expanded restricted policy set Π_i^r for each player. At each epoch, a Nash equilibrium $\sigma = (\sigma_i, \sigma_{-i})$ is computed for a restricted meta-game formed by a tuple of restricted policy sets $\Pi^r = (\Pi_i^r, \Pi_{-i}^r)$. Then, a best response to this Nash equilibrium for each player i is computed and added to its restricted policy set as $\Pi_i^r = \Pi_i^r \cup \{\mathbf{BR}(\sigma_{-i})\}$. As a generalization of DO, PSRO defines restricted meta-games with sets of reinforcement learning policies. In this case, the normal-form games in PSRO are an abstraction of feasible joint policy space instead of trivial matrix games. Thus, techniques for DO algorithms cannot be directly applied to PSRO cases; also, PSRO are challenged by higher complexity than DO due to nested reinforcement learning procedures. In practice, PSRO learns to approach an approximate NE at a level of precision $\epsilon \geq 0$ (Aziz, 2010). To evaluate the quality of the approximation, we use $\text{NASHCONV}(\sigma) = \sum_i u_i\left(\mathbf{BR}_i(\sigma_{-i}), \sigma_{-i}\right) - u_i(\sigma)$ to compute the exploitability of σ to oracles $\{\mathbf{BR}(\sigma_{-i}) \mid i = 1, 2\}$ (Johanson et al., 2011). σ is an exact Nash equilibrium if $\text{NASHCONV} = 0$. It is no difficulty to know that higher exploitation efficiency brings a lower exploitability.

We summarize the pseudo-code of PSRO in Algorithm 1. At each epoch, PSRO learns approximate best responses by running nested reinforcement learning algorithms. However, such a procedure is data-thirsty and has no guarantee of finding a high-quality best response to bring higher payoffs for a restricted policy set, especially in the case of complex tasks. After the best response learning, PSRO requires dedicated simulations to compute the missing items in U^{Π^r}. In general, the amount of simulations grows geometrically as $\mathcal{O}(M \cdot |\Pi_i^r|)$, where $|\Pi_i^r|$ and M denote the size of a restricted policy set and the number of simulations for each missing item, respectively.

Algorithm 1: VANILLA PSRO

Input: initial restricted policy sets $\Pi^r = (\Pi_1^r, \Pi_2^r)$
/* can be saved via URR games */
Input: empty payoff table U^{Π^r}
Input: meta-strategies $\sigma_i \sim \text{UNIFORM}(\Pi_i^r)$
while *not terminated* **do**
  for *player* $i \in \{1, 2\}$ **do**
    for *many episodes* **do**
      Train best response $\pi_{i,\theta}$ against $\pi_{-i} \sim \sigma_{-i}$
    $\Pi_i^r = \Pi_i^r \cup \{\pi_{i,\theta}\}$
  /* can be saved via URR games */
  Run simulations to compute missing entries in U^{Π^r}
  Compute a meta-strategy σ from U^{Π^r}
Output: current meta-strategy σ_i for player i

Algorithm 2: SIMPLIFIED PSRO WITH URR GAMES

Input: initial restricted policy sets $\Pi^r = (\Pi_1^r, \Pi_2^r)$
while *not terminated* **do**
  for *player* $i \in \{1, 2\}$ **do**
    Random initialize a best response $\pi_{i,\theta}$
    $(\pi_{i,\theta}, \sigma_{-i,\beta}) =$ SOLVEURR$(\pi_{i,\theta}, \Pi_{-i}^r)$
  $\Pi_i^r = \Pi_i^r \cup \{\pi_{i,\theta}\}$ for $i \in \{1, 2\}$
Output: current meta-strategy σ_i for player i

3 EPSRO: EFFICIENT PSRO

The keys to EPSRO's high efficiency lie in two aspects: (1) eliminating simulations for meta-strategies computing; (2) learning high-quality best responses. For the first aspect, we developed EPSRO on top of URR games (Section 3.1), it solves the computational inefficiency; for the second aspect, we perform no-regret optimization (Section 3.2) for URR solving, it solves the exploration inefficiency. Considering the influence of growing policy sets on training, we leverage warm-start learning and parallelism (Section 3.3) to save training costs. We summarize the pseudo code of EPSRO in Algorithm 4 and exhibit its overview in Figure 1.

3.1 MODELING EPSRO AS URR GAMES

Restricted meta-games are constructed for computing meta-strategies in previous PSRO methods. However, solving it relies on numerous simulations, which challenges the computational efficiency. In this work, we eliminate restricted meta-game solving by re-formulating PSRO as an *unrestricted-restricted* (URR) game. Although a similar concept was proposed in Zinkevich et al. (2007a), it requires a nested restricted meta-game procedure and only focuses on tabular cases. Thus, our URR game modeling exhibits more potential in comparison.

Definition 3.1 (*URR Game*)**.** An *unrestricted-restricted* game for player i is a tuple (Π_i, Π^r_{-i}). Π_i the i's full policy set; Π^r_{-i} the $-i$'s restricted policy set. In this game, the i models its policy as a function parameterized by θ, i.e. $\pi_{i,\theta} \in \Delta(\Pi_i)$; $-i$ models its strategy as a function parameterized by β, i.e. $\sigma_{-i,\beta} \in \Delta(\Pi^r_{-i})$. All players play against each other and converge to a Nash equilibrium $(\pi^\star_{i,\theta}, \sigma^\star_{-i,\beta})$ where $\pi^\star_{i,\theta} = \mathbf{BR}(\sigma^\star_{-i,\beta})$, and $\sigma^\star_{-i,\beta} = \mathbf{BR}(\pi^\star_{i,\theta})$.

As described in Definition 3.1, a URR game directly computes meta-strategies and best responses by performing adversarial learning instead of learning best responses to fixed meta-strategies which are derived from restricted meta-games solving. Therefore, URR games save dedicated simulations to achieve higher computational efficiency. Algorithm 2 lists the pseudo-code of an URR-based PSRO for comparison with the vanilla PSRO (Algorithm 1). Despite saving simulation costs, we need to investigate whether the expansion of its policy set is non-degenerate as it is a key principle to exploration efficiency. Conceptually, the non-degenerate policy set expansion means that a new added best response has to satisfy: (1) not be in existing gamescape; (2) bring utility improvements and exploitability decrease when playing against any opponent meta-strategies $\sigma_{-i} \in \Delta(\Pi^r_{-i})$. The **gamescape** is introduced in Balduzzi et al. (2019) to evaluate how closely a restricted policy set is to the full policy space Π. Though being originally defined in the field of self-play, the concept of gamescape can be naturally extended to URR games as follows.

Definition 3.2 (*URR Gamescape, derived from Balduzzi et al. (2019)*)**.** Given an URR game with payoff matrix $\mathbf{U}$, the corresponding empirical gamescape (EGS) is $\mathcal{G} := \{\textit{convex mixture of columns of } \mathbf{U}\}$.

Theorem 3.3 (*Monotonic Policy Space Expanding*)**.** *For any given epoch e and $e+1$, let (π^e_i, σ^e_{-i}) and $(\pi^{e+1}_i, \sigma^{e+1}_{-i})$ be Nash equilibrium of $\textbf{\textit{URR}}^e_i$ and $\textbf{\textit{URR}}^{e+1}_i$, respectively, where $\pi^e_i, \pi^{e+1}_i \in \Pi_i$, $\sigma^e_{-i} \in \Delta^e_{\Pi^r_{-i}}$ and $\sigma^{e+1}_{-i} \in \Delta^{e+1}_{\Pi^r_{-i}}$. The utilities of π^e_i against strategies σ^e_{-i} and σ^{e+1}_{-i} satisfies*

$$u_i(\pi^e_i, \sigma^{\mathbf{e}}_{-\mathbf{i}}) - u_i(\pi^e_i, \sigma^{\mathbf{e+1}}_{-\mathbf{i}}) \geq 0, \tag{1}$$

where $\Delta^e_{\Pi^r_{-i}}$ indicates $\Delta(\Pi^{r,e}_{-i})$. (See Appendix B.1)

Theorem 3.3 indicates that EPSRO has a monotonic utility improvement and policy space expansion with URR games. We further investigate that EPSRO has higher exploration efficiency than the naive PSRO (Section 4) and discuss this property theoretically (Appendix B.2).

3.2 Solving URR Games

We've built our EPSRO with URR and gave analysis for its policy set expansion, but how to efficiently solve URR games is still a question. A feasible method for the implementation is no-regret optimization (Bowling, 2004; Daskalakis et al., 2011), which is used to solve two-player zero-sum games. Under this framework, a learning algorithm could approximate the NE asymptotically by playing the same game repeatedly. We give its definition as follows.

Definition 3.4. Considering a sequence of mixed strategies for player i as $\pi_1, \pi_2, \ldots$, an algorithm of $-i$ that generates a sequence of mixed strategies $\sigma_1, \sigma_2, \ldots$ is called a no-regret algorithm if we have: $\lim_{T\to\infty} R_T/T = 0,\ R_T = \max_{\sigma\in\Delta_{\Pi_{-i}}} \sum_{t=1}^{T} (\pi_t^\top U\sigma - \pi_t^\top U\sigma_t)$.

A well-known no-regret algorithm is Multiplicative Weights Update (MWU) (Freund & Schapire, 1999), which updates strategy by considering the averaging loss vector to opponent strategies along the learning horizon and then achieves no-regret as the learning horizon towards infinite. We give its definition in Appendix B.3. In our implementation, we leverage a variant of MWU for meta-strategies learning and an off-policy reinforcement learning algorithm for best response learning. We list the pseudo-code in Algorithm 3. The opponent meta-strategy σ_{-i} is represented as a Boltzman distribution, which is parameterized by a vector $\beta = [\beta_1, \ldots, \beta_{|\Pi^r_{-i}|}]$. Thus, each support of σ_{-i} could be expressed as $\sigma_{-i}(j) = \frac{\exp\beta_j}{\sum_{i=1}^{n}\exp\beta_i}$. We update σ_{-i} by following Algorithm 5.

Algorithm 3: SOLVEURR

Input: URR game (Π_i, Π^r_{-i}), BR $\pi_{i,\theta} \sim \Delta(\Pi_i)$
Input: meta-strategies $\sigma_{-i,\beta} \sim \text{UNIFORM}(\Pi^r_{-i})$
while *not terminated* **do**
  Train best response $\pi_{i,\theta'} \leftarrow \pi_{i,\theta}$ against $\pi_{-i} \sim \sigma_{-i,\beta}$ with reinforcement learning step
  Update $\sigma_{-i,\beta'} \leftarrow \sigma_{-i,\beta}$ against $\pi_{i,\theta}$ by following Algorithm 5 in Appendix A
  $\pi_{i,\theta} \leftarrow \pi_{i,\theta'}$, $\sigma_{-i,\beta} \leftarrow \sigma_{-i,\beta'}$
Output: an approximate Nash $(\pi^\star_{i,\theta}, \sigma^\star_{-i,\beta})$

Warm-Starting Algorithm. In EPSRO, a critical issue for the meta-strategy learning is that the length of meta-strategy changes as the policy set expands. Therefore, we need to learn the meta-strategy with a new expanded β at each epoch. However, if we start learning from scratch, the cost of re-training grows more and more expensive as the policy set expands, resulting in a slow convergence rate. To tackle this issue, we propose a warm-starting technique that enables the meta-strategy starts from a non-trivial initialization. Since the warm-starting aims to save training costs to achieve a small enough regret when Π^r changes, so the key is to correctly initialize the regrets instead of only starting the strategy. Because a wrong initialization of the regrets will result in huge regrets in subsequent iterations, which is no better than starting from scratch. As pointed in Brown & Sandholm (2016), a feasible warm-starting should be a substitute strategy that does not violate the regret bound and the approximate NE of the last epoch.

Theorem 3.5 (*Theorem 1 in Brown & Sandholm (2016)*)**.** *In a two-player zero-sum game, if* $\frac{R_i^T}{T} \leq \epsilon_i$ *for both player* $i \in \{1, 2\}$*, then* $(\bar{\pi}_i, \bar{\sigma}_{-i})$ *is a* $(\epsilon_1 + \epsilon_2)$*-equilibrium, where* $\bar{\sigma}_{-i} = \sum_{t=1}^T \langle \sigma_{-i,t}, l_{-i,t} \rangle / \sum_{t=1}^T Tl_{-i,t}$*,* $\bar{\pi}_i = \sum_{t=1}^T \langle \pi_{i,t}, l_{i,t} \rangle / \sum_{t=1}^T Tl_{i,t}$*,* R_i^T *the summation of regrets of* T *iterations,* l_t *is the loss vector to the opponent policy set.*

Theorem 3.5 shows that if we use a regret-based average $\bar{\sigma}_{-i}$ (or $\bar{\pi}_i$) as the substitute of a sequence of $\sigma_{-i,t}$ (or $\pi_{i,t}$), the substitute can still hold the equilibrium. In EPSRO, as the restricted policy set grows from $\Pi^{r,e}_{-i}$ to $\Pi^{r,e+1}_{-i}$, we need to investigate whether there is a feasible substitute strategy $\bar{\sigma}^{'}_{-i} \in \Delta^{e+1}_{\Pi^r_{-i}}$ to $\bar{\sigma}_{-i} \in \Delta^{e}_{\Pi^r_{-i}}$ follows the regret guarantee of epoch e.

Theorem 3.6 (*See Appendix B.5*)**.** *Suppose a substitute policy of* $\bar{\sigma}_{-i} \in \Delta^e_{\Pi^r_{-i}}$ *is* $\bar{\sigma}^{'}_{-i} \in \Delta^{e+1}_{\Pi^r_{-i}}$*, and it satisfies* $u_i^{e+1}(\bar{\pi}_i, \bar{\sigma}^{'}_{-i}) = u_i^e(\bar{\pi}_i, \bar{\sigma}_{-i})$*, we have*

$$\max_{\sigma_{-i} \in \Delta^{e+1}_{\Pi^r_{-i}}} \frac{1}{T} \sum_{t=1}^T \left(u_{-i}^{e+1}(\pi_i^t, \sigma_{-i}) - u_{-i}^{e+1}(\pi_i^t, \bar{\sigma}^{'}_{-i}) \right) \leq \epsilon^T_{-i},$$

where ϵ^T_{-i} *is the regret bound of epoch* $e+1$*,* $\pi_i^t \in \Delta(\Pi_i)$*.*

The motivation behind Theorem 3.6 is that a strategy satisfies the regret bound at e can be a non-trivial regret initialization at $e+1$. Thus, once we compute such a $\bar{\sigma}^{'}_{-i}$ that satisfies $u_i(\bar{\pi}_i, \bar{\sigma}^{'}_{-i}) = u_i(\bar{\pi}_i, \bar{\sigma}_{-i})$, then it is a feasible warm-start at the next epoch to save the training time. As multiple substitute strategies exist that satisfy the above conditions, the latest best response may not contribute to a learned one, making the learning of meta-strategies stuck at an old policy set. To solve this problem, we can compute an initialization with some regularizer.

Lemma 3.7 (*See Appendix B.7*)**.** *Let* $k = |\Pi^{r,e+1}_{-i}|$*,* $\bar{\sigma}^{'}_{-i}$ *be parameterized by* $\beta_{-i} = [\beta_{-i,1}, \beta_{-i,2}, \ldots, \beta_{-i,k}]$*,* $\bar{\sigma}^{'}_{-i}(k)$ *the* k*-th item of* $\bar{\sigma}^{'}_{-i}$*,* $x = [\bar{\sigma}^{'}_{-i}(1), \ldots, \bar{\sigma}^{'}_{-i}(k-1)]^T$*,* $\bar{l}^e_{-i}$ *is* $-i$*'s average loss vector to* $\bar{\pi}_i$ *at epoch* e*. Then a feasible initial of* β^{e+1}_{-i} *could satisfy*

$$\beta^{e+1}_{-i} = \arg\min_{\beta_{-i}} \parallel (x - \bar{\sigma}_{-i})^\top \bar{l}^e_{-i} - \bar{\sigma}^{'}_{-i}(k) u_{-i}(\bar{\pi}_i, \pi^{e+1}_{-i}) \parallel_2 - \lambda H(\bar{\sigma}^{'}_{-i}), \tag{2}$$

where $\lambda > 0$*,* $H(\bar{\sigma}^{'}_{-i})$ *is the entropy of* $\bar{\sigma}^{'}_{-i}$*.*

Lemma 3.7 computes a β^{e+1}_{-i} for the initialization of σ^{e+1}_{-i} to satisfy Theorem 3.6. In addition, the best response $\pi_{i,\theta}$ also follows these conditions by continuous training instead of restarting parameters at each epoch. We now present the regret bound of EPSRO as follows.

Theorem 3.8 (*Regret Bound of EPSRO*). *Let $l_1, l_2, \ldots, l_T$ be a sequence of loss vectors player by an opponent, and $\langle \cdot, \cdot \rangle$ be the dot product, then EPSRO is a no-regret algorithm with (See Appendix B.8)*

$$\frac{1}{T}\left(\sum_{t=1}^{T}\langle \sigma_t, l_t\rangle - \min_{\sigma\in\Delta(\Pi_t^r)}\sum_{t=1}^{T}\langle \sigma, l_t\rangle\right) \leq \frac{\sqrt{\log\left[(k+1)k/2\right]}}{\sqrt{2T}},\ \textit{where } k \textit{ is the size of } \Pi^r.$$

Theorem 3.8 shows that EPSRO is no-regret when the policy set is finite and horizon tends to infinite. Though some complex games have continuous policy space, the number of effective policies is finite in general. We further give the convergence rate of EPSRO as follows.

Theorem 3.9 (*Convergence Rate of EPSRO*). *Let k, N denote the size of restricted policy sets Π_{-i}^r and Π_i. Then the learning of Algorithm 3 will converge to the Nash equilibrium with the rate (See Appendix B.9):*

$$\epsilon_T = \sqrt{\frac{\log\left[(k+1)k/2\right]}{2T}} + \sqrt{\frac{\log\left[(N+1)N/2\right]}{2T}}.$$

3.3 Pipeline URR Solver

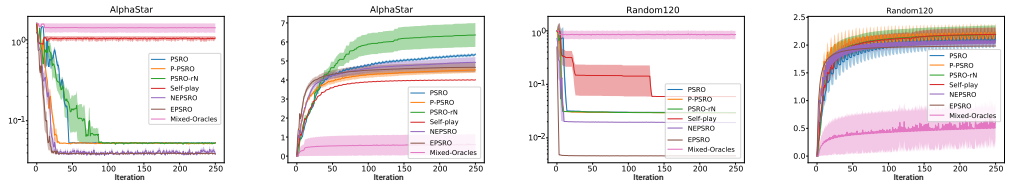
While Algorithm 3 is clear, the reinforcement learning step can take a long time to converge to a good response, especially in complex games. To overcome this inefficiency, we introduce a parallel solution for EPSRO. Inspired by Pipeline-PSRO (P-PSRO) (McAleer et al., 2020), we maintain a queue of ranked policies for each player i. An active policy $\pi_{i,\theta}^j$ ranked with j will be trained with parallized URR games. For each URR game, $\pi_{i,\theta}^j$ plays against an opponent meta-strategy $\sigma_{-i,\beta}^{<j}$ of $\Pi_{-i}^{r,<j}$, composed of lower ranked policies $\{\pi_{-i}^1, \ldots, \pi_{-i}^{j-1}\}$. Once a lowest active policy meets the stop condition (e.g. number of training episodes, we set it to 10,000 episodes in our implementation), it will be fixed, and a new active policy will be added into the queue with the highest level. Figure 1 illustrates an example dynamics, it able to scale up the no-regret optimization with convergence guarantee by maintaining a hierarchical pipeline of reinforcement learning policies as P-PSRO (Proposition 3.2 in McAleer et al. (2020)).

Algorithm 4: Efficient PSRO (EPSRO)

Input: inital restricted policy sets $\Pi^r = (\Pi_1^r, \Pi_2^r)$
while *not terminated* **do**
  for *all player* $i \in \{1, 2\}$ *in parallel* **do**
    for *all active policies* $\pi_i^j \in \Pi_i^r$ *in parallel* **do**
      $\pi_{i,\theta}^j, \sigma_{-i,\beta}^{<j} = \textsc{SolveURR}(\pi_{i,\theta}^j, \Pi_{-i}^{r,<j})$
      if $\pi_{i,\theta}^j$ *is the lowest active policy and meets stops cond.* **then**
        Set it to fixed and $\Pi_i^r := \Pi_i^r \cup \{\pi_i^j\}$
        Generate a new active policy
Output: current meta-strategies $\sigma = (\sigma_1, \sigma_2)$

4 Experiments

We compare EPSRO with five algorithms, including Self-play (Hernandez et al., 2019), PSRO (Lanctot et al., 2017), Rectified PSRO (PSRO-rN) (Balduzzi et al., 2019), Mixed-Oracles (Smith et al., 2020), and Pipeline PSRO (P-PSRO) (McAleer et al., 2020). The environments for the test are four classes of increasing difficulty games, i.e., matrix games, Poker games, card game (Apppendix F.2) and Multi-agent Gathering. We investigate the exploration and computational efficiency of algorithms in these experiments, also the performance on convergence. We use NashConv to evaluate the convergence quality and exploration efficiency, and *cardinality of payoff matrix* (Régin & Gomes, 2004) to assess the diversity, i.e., the policy differences in the policy set. The matrix games are designed for the comparison of exploration efficiency. Especially the non-transitive mixture game (Section 4.1), which vividly characterizes the exploration dynamics of algorithms. As for the computational efficiency, we analyze it from the number of samples and time consumption in poker games. Since the card game and multi-agent gathering tasks are difficult to traverse the game tree, we evaluate them with a normalized scoring. Extra results and the pseudo-code of score calculation are in Appendix F. The higher the score, the lower exploitability of the algorithm. We repeat all experiments with 5 different random seeds on a single machine with 64 CPUs, 256 G RAM, and 2 GeForce RTX 3090 GPUs.



(a) NashConv (b) Cardinality (c) NashConv (d) Cardinality

Figure 2: Comparison of NASHCONV and cardinality. (a) and (b) show the NASHCONV and cardinality on AlphaStar matrix game, respectively; (c) and (d) show the NASHCONV and cardinality on a high-dimensional symmetric game ($n = 120$), respectively. Although EPSRO has lower cardinality than some other algorithms, it outperforms all baselines on the NASHCONV. We argue that a lower cardinality but better exploitability indicates that the algorithm has higher exploration efficiency since it achieves better performance with fewer policies. More results in Appendix F.3.

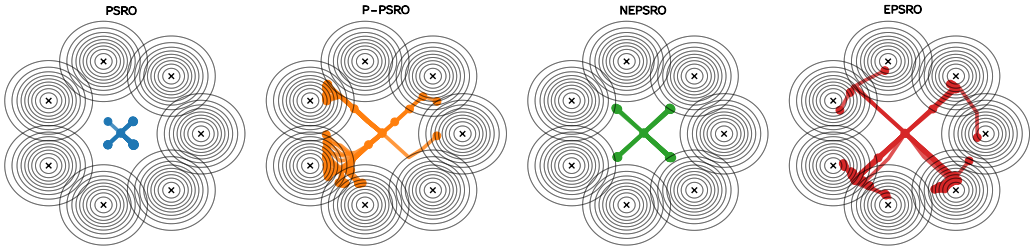
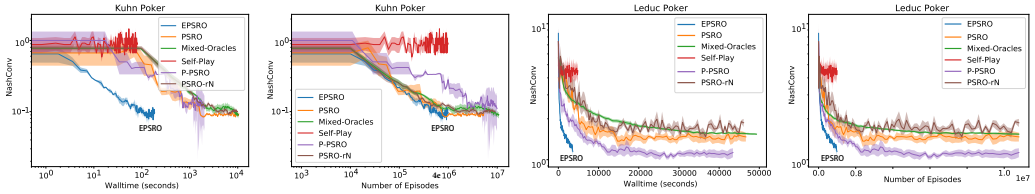


Figure 3: Exploration trajectories on *Non-transitive Mixture Games*. The more trajectories close to the centers of Gaussian, the higher the exploration efficiency of the algorithm. EPSRO outperforms all selected baselines since it explored all centers. More results in Appendix F.1.



(a) vs. Wall-time (b) vs. Samples (c) vs. Wall-time (d) vs. Samples

Figure 4: NashConv on poker games. The number of samples for training each best response is set to $1e4$ episodes, and the number of simulations (if needed) for each joint policy is set to $1e3$ episodes. EPSRO achieves similar NASHCONV as P-PSRO. For the computation efficiency, EPSRO achieves more than 50x speedup on wall-time; more than 2.5x sample efficiency than other algorithms.

4.1 COMPARISON OF EXPLORATION EFFICIENCY

The non-transitive game for the comparison of exploration efficiency is a zero-sum two-player game consisting of 7 equally-distanced Gaussian humps on the 2D plane. In this game, each player chooses a point in the 2D plane as its decision, which is transformed into a 7-dimensional vector π_i with each coordinate being the density in the corresponding Gaussian distribution. The payoff is given by $\phi_i(\pi_i, \pi_{-i}) = \pi_i^\top \mathbf{S}\pi_{-i} + \mathbf{1}^\top(\pi_i - \pi_{-i})$, where $\mathbf{S}$ is the payoff matrix. In this game, the optimal strategy should stay close to the center of the Gaussian and explore all the Gaussian distributions equally. We train best response policies in 50 epochs for each algorithm. As presented in Figure 3, EPSRO successfully explores all the centers and shows higher exploration efficiency than other baselines. Although the NEPSRO fails to achieve to any centers like PSRO, its explored policy space is larger than most algorithms.

4.2 HIGH-DIMENSIONAL MATRIX GAMES

Random Symmetric Matrix Game. McAleer et al. (2020) introduce the games to investigate the performance of PSRO-based methods in high-dimensional symmetric games (SymGame). In these experiments, we train a strategy π as a best response that plays against another strategy $\hat{\pi}$, it is updated by a learning rate r multiplied by the best response to that strategy: $\pi' = r\mathbf{BR}(\hat{\pi})+(1-r)\pi$. Figure 2 shows the results for $n = 120$. We report both NASHCONV and cardinality. The results show that EPSRO and NEPSRO achieve a faster convergence rate and the lowest NASHCONV than all of the other algorithms. Though they do not achieve the highest cardinality, we argue that there is a trade-off between convergence and exploration, and EPSRO performs a better balance between them. It is worth mentioning that the Mixed-Oracle fails to seek a meta-strategy that has a smaller distance to the Nash equilibrium, even worse than Self-Play. We argue that the policy distills of Mixed-Oracles may harm the exploration efficiency, especially in a high-dimensional policy space.

AlphaStar Empirical Game. The AlphaStar Matrix Game is derived from solving a complex real-world game StraCraftII (Czarnecki et al., 2020), which involves 888 reinforcement learning policies. We test the exploration efficiency and the convergence quality of our method for solving such empirical games. Similar to the results in the random symmetric matrix game, our algorithm performs a faster convergence rate and lower NASHCONV than other algorithms, while the Mixed-Oracle still fails to explore new policies to expand its policy sets (Figure 2).

4.3 POKER GAMES

Poker is a common benchmark in multi-agent decision tasks. In this paper, we introduce two simplified forms of poker games for experiments, i.e., Kuhn Poker and Leduc Poker (Zinkevich et al., 2007b; Bowling et al., 2015). These poker tasks model zero-sum two-player imperfect-information games, in which each player shows uncertainty about the game rules and the state of other players. Similar to Poker, where each player in these games chooses to raise/call/fold through rounds of betting. We investigate the sample efficiency and performance of EPSRO in this experiment. The training for each algorithm is set to learn 100 policies. The number of samples for training each policy is set to 10000 episodes, and the number of simulations for each joint policy is set to 1000 episodes. Figure 4 presents the results of NACHCONV w.r.t to wall-time and the number of samples. Since no simulations and parallel best response learning, EPSRO substantially achieves high training efficiency. Specifically, for the wall-time, EPSRO has more than 50x speedup than baselines; for the sample efficiency, EPSRO has more than 2.5x than other non-parallelized baselines.

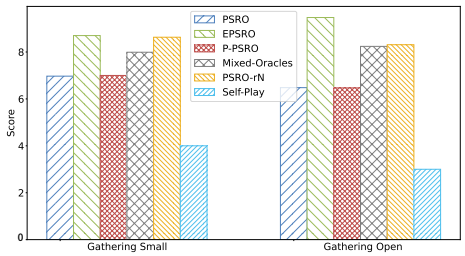


Figure 5: The average score of final policy set that plays against Π^{PSRO}.

4.4 MULTI-AGENT GATHERING

We further investigate the capability of EPSRO to handle more complex multi-agent tasks in multi-agent gathering environments (MAG)[1]. MAG is an endless environment whose horizon length could be infinite. In our experiments, we limit the horizon of each episode to 100. In MAG, the goal of each agent is to collect as many as "apples". The apples regrow at a rate dependent upon the configuration of the uncollected nearby apples. In this case, the more nearby apples, the higher the regrowth rate of the apples. Naturally, this presents a dilemma for the players: each wants to pick as many apples as possible. However, if they over-harvest the throughput of apples diminishes, potentially falling to zero. Figure 5 shows the confrontation results of each algorithm's final policy set Π^{TEST} to an evaluation policy set Π^{PSRO}. We calculate the score as $\sigma^{\text{TEST}} M^{\text{TEST}} \left[\sigma^{\text{PSRO}}\right]^T$, where M^{TEST} is the empirical payoff matrix and σ is a learned meta-strategy. The higher the score, the lower the exploitability and the better the performance of the corresponding algorithm. We report the learning curve in Figure 10 (Appendix F.4).

[1] https://github.com/HumanCompatibleAI/multi-agent

4.5 Ablation Study

To investigate the effect of warm-starting and pipeline, we further conduct ablation experiments on poker games. In our experiments, three kinds of variants of EPSRO are tested, i.e., $--$EPSRO for no warm-starting and parallelism, $-+$EPSRO for warm-starting but no parallelism, $+-$EPSRO for parallelism but no warm-starting. We run five trials for each algorithm and train 100 epochs to generate 100 policies. Figure 6 shows the comparison results on Leduc poker. Although the warm-starting variant outperforms $--$EPSRO, it fails to achieve a better performance than PSRO. We think there may be challenging to improve the training efficiency of no-regret optimization with warm-starting alone in the case of a nested reinforcement learning step. Also, the parallelism variant fails to outperform PSRO. As a comparison, EPSRO is equipped with warm-starting and parallelism, achieving a faster convergence rate and better performance than the other algorithms. Thus, we think warm-starting and parallelism can improve the efficiency of each other to some extent.

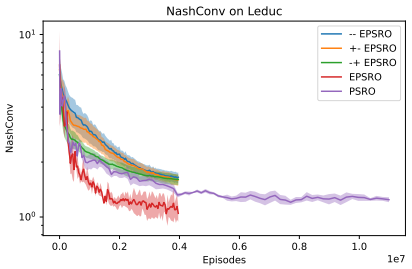


Figure 6: Ablation study on Leduc.

5 Related Work

There are many variants of PSRO that focus on improving exploration efficiency, while none of them concentrate on simulation elimination. For instance, DCH (Lanctot et al., 2017) parallelizes PSRO by training multiple RL policies, each against the meta-strategies below it in the hierarchy. A problem with this method is that the number of policies should be set beforehand. However, it is difficult to figure out how many policies does it require to solve a game in practice. McAleer et al. (2020) proposed a similar solution (P-PSRO), to solve this problem, which inherits the parallelism training but has no need to preset the number of policies. Another parallelism variant is Rectified PSRO (Balduzzi et al., 2019), but it has been proved not converge in all symmetric zero-sum games (McAleer et al., 2020). Also, the quality of best responses affects the exploration efficiency. Specifically, stronger the learned best responses, higher the efficiency (Perez-Nieves et al., 2021; Yang et al., 2021). However, it is difficult to discover an exact best response in practice. As a solution, the researchers try to improve the policy diversity. Among the existing work, DPP (Perez-Nieves et al., 2021) utilizes the expected cardinality to measure the diversity of policy set. Liu et al. (2021b) proposed a method that unify both behavior and reward distance to measure the diversity.

Concurrently to our work, McAleer et al. (2022) proposed a similar work named AODO. Their work differs from ours in the following ways: (1) We focus on improving the training efficiency of PSRO-based methods, providing monotone improvement and convergence analysis for EPSRO, while AODO focuses on making a guarantee to monotonic decrease the NashConv (Johanson et al., 2011); (2) EPSRO introduces parallelized best response training while AODO executes in singleton; (3) We proposed a warm-start technique to tackle the re-training problem while AODO starts from scratch at each epoch; (4) We demonstrate the experiments with 5 baselines on both high-dimensional matrix games, poker games, card games and non-trivial multi-agent gathering tasks while AODO considers only one baseline and fewer environments.

6 Conclusions

In this work, we propose a parallel algorithm EPSRO to improve the efficiency of PSRO. The empirical results show that EPSRO achieves higher efficiency than existing works. The improvements of EPSRO benefit from learning best responses against the whole opponent restricted policy set and cooperating with parallelism. However, EPSRO is limited to handling the two-player cases because there will be a divergence in selecting meta-strategies for more players involved. In future work, we would like to seek a method to solve this problem and generalize EPSRO to multi-player cases. Moreover, since the warm-starting used in our current implementation heavily relies on the approximate NE from the last epoch, it may be challenging to achieve an ideal performance when the approximate NE is low quality. Thus, a warm-starting that can be flexible in choosing any init regret would be more valuable.

REFERENCES

Haris Aziz. Multiagent systems: algorithmic, game-theoretic, and logical foundations by y. shoham and k. leyton-brown cambridge university press, 2008. *ACM Sigact News*, 41(1):34–37, 2010.

David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning*, pp. 434–443. PMLR, 2019.

Michael Bowling. Convergence and no-regret in multiagent learning. *Advances in neural information processing systems*, 17, 2004.

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

Noam Brown and Tuomas Sandholm. Strategy-based warm starting for regret minimization in games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Shicong Cen, Fan Chen, and Yuejie Chi. Independent natural policy gradient methods for potential games: Finite-time global convergence with entropy regularization. *arXiv preprint arXiv:2204.05466*, 2022.

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.

Constantinos Daskalakis, Alan Deckelbaum, and Anthony Kim. Near-optimal no-regret algorithms for zero-sum games. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pp. 235–254. SIAM, 2011.

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

Drew Fudenberg and Jean Tirole. *Game theory*. MIT press, 1991.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Thomas Dueholm Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. On range of skill. In *AAAI*, pp. 277–282, 2008.

Daniel Hernandez, Kevin Denamganaï, Yuan Gao, Peter York, Sam Devlin, Spyridon Samothrakis, and James Alfred Walker. A generalized framework for self-play training. In *2019 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE, 2019.

Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In *Twenty-second international joint conference on artificial intelligence*, 2011.

Marc Lanctot, Vinicius Zambaldi, Audrūnas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 4193–4206, 2017.

Yaodong Yang Le Cong Dinh, Zheng Tian, Nicolas Perez Nieves, Oliver Slumbers, David Henry Mguni, Haitham Bou Ammar, and Jun Wang. Online double oracle. 2021.

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

Siqi Liu, Luke Marris, Daniel Hennes, Josh Merel, Nicolas Heess, and Thore Graepel. Neupl: Neural population learning. In *International Conference on Learning Representations*, 2021a.

Xiangyu Liu, Hangtian Jia, Ying Wen, Yaodong Yang, Yujing Hu, Yingfeng Chen, Changjie Fan, and Zhipeng Hu. Towards unifying behavioral and response diversity for open-ended learning in zero-sum games. *Advances in Neural Information Processing Systems*, 34, 2021b.

Stephen McAleer, JB Lanier, Roy Fox, and Pierre Baldi. Pipeline psro: A scalable approach for finding approximate nash equilibria in large games. *Advances in Neural Information Processing Systems*, 33:20238–20248, 2020.

Stephen McAleer, John Banister Lanier, Kevin A Wang, Pierre Baldi, and Roy Fox. Xdo: A double oracle algorithm for extensive-form games. *Advances in Neural Information Processing Systems*, 34:23128–23139, 2021.

Stephen McAleer, Kevin Wang, Marc Lanctot, John Lanier, Pierre Baldi, and Roy Fox. Anytime optimal psro for two-player zero-sum games, 2022.

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)*, pp. 536–543, 2003.

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.

Nicolas Perez-Nieves, Yaodong Yang, Oliver Slumbers, David H Mguni, Ying Wen, and Jun Wang. Modelling behavioural diversity for learning in open-ended games. In *International Conference on Machine Learning*, pp. 8514–8524. PMLR, 2021.

Jean-Charles Régin and Carla P Gomes. The cardinality matrix constraint. In *International Conference on Principles and Practice of Constraint Programming*, pp. 572–587. Springer, 2004.

Max Smith, Thomas Anthony, and Michael Wellman. Iterative empirical game solving via single policy best response. In *International Conference on Learning Representations*, 2020.

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

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*, pp. 259–278. SIAM, 2020.

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.

Yongzhao Wang, Qiurui Ma, and Michael P Wellman. Evaluating strategy exploration in empirical game-theoretic analysis. *arXiv preprint arXiv:2105.10423*, 2021.

Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 5571–5580. PMLR, 2018.

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*, pp. 1575–1583, 2020.

Yaodong Yang, Jun Luo, Ying Wen, Oliver Slumbers, Daniel Graves, Haitham Bou Ammar, Jun Wang, and Matthew E Taylor. Diverse auto-curriculum is critical for successful real-world multiagent learning systems. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 51–56, 2021.

Martin Zinkevich, Michael Bowling, and Neil Burch. A new algorithm for generating equilibria in massive zero-sum games. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 22, pp. 788. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007a.

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, 2007b.

A ALGORITHMS

Algorithm 5 shows the pseudo-code of meta-strategy optimization. Although the update of EPSRO's meta-strategies does not exploit the loss directly like MWU, it follows MWU by exploiting the computed gradients. We further explain it in Lemma B.4.

Algorithm 5: DETERMINISTIC META STRATEGY OPTIMIZATION

Input: initialize the $\sigma_{-i,\beta}$ with Algorithm 6
Input: initialize an episode reword buffer $\mathcal{B}$;window size L; K the length of meta-strategy $\sigma_{-i,\beta}$
Input: loss vector buffer $\mathcal{L}_i, \mathcal{L}_{-i} \in \mathbb{R}^K$; counters: $N_1 = 0, \ldots, N_K = 0$
while *not terminated* **do**
  for *many episodes* **do**
    Sample $\pi_{-i}^k \sim \sigma_{-i,\beta}$ to play against $\pi_{i,\theta}$
    Collect episode return r^k into $\mathcal{B}$ and $N_k := N_k + 1$
    if *the size of* $\mathcal{B}$ *equals to* L **then**
      Calculate average observed returns: $\forall k = 1, \ldots, K, \bar{r}^k = \frac{\sum_{l=1}^{L} \mathbf{1}_{j=k} r_l^j}{N_k}$
      Compute gradients for $\beta_k = \arg\max_{\beta_k} \bar{r}^k$ and update $\sigma_{-i,\beta}$ with Lemma B.7
      Collect loss vectors $l_{-i}^k = -\bar{r}^k$, $l_i^k = \bar{r}^k$ into $\mathcal{L}_i$, $\mathcal{L}_{-i}$, respectively
      Reset buffer $\mathcal{B}$ and counters
Output: meta-strategy σ_{-i} for player $-i$, loss vector buffers $\mathcal{L}_i$, $\mathcal{L}_{-i}$

We introduce the warm-starting for opponent meta-strategy as follows. For each player i, the algorithm randomly initializes a vector β with the length equals to the size of Π_{-i}^r. At each epoch $e+1$, we slightly run a simulation procedure of (π_i, π_{-i}) to estimate the utility $u_{-i}(\pi_i, \pi_{-i})$ before applying the Lemma 3.7 to optimize β. Then, by cooperating with the average loss vector from last epoch e, we could start the optimization to compute a feasible β^* that makes σ_{-i} satisfy Theorem 3.6.

Algorithm 6: META-STRATEGY WARM-START

Input: newly learned policy support π_{-i}; best response π_i from last epoch; error threshold $\tau \geq 0$
Input: initialize σ_{-i} with $\beta \in \mathbb{R}^{|\Pi_{-i}^r|}$; loss vectors from last epoch $[l_{-i}^1, \ldots, l_{-i}^T]$
Compute average loss vector $\bar{l}_{-i} = \left(l_{-i}^1 + \cdots + l_{-i}^T\right)/T$
Estimate $u_{-i}(\pi_i, \pi_{-i})$ by running simulation for (π_i, π_{-i})
Compute the square error of utility as
$\xi_2 = \| (x - \sigma_{-i})^\top \bar{l}_{-i} - \sigma_{-i}(k) u_{-i}(\pi_i, \pi_{-i}) \|_2 - \lambda H(\sigma_{-i})$
Update β as $\beta := \beta - \nabla_\beta \xi_2$
if $\xi_2 \leq \tau$ **then**
  Stop optimization and continue
else
  Go back to step 3
Output: meta-strategy σ_{-i}

Algorithm 6 relies on the computation of ξ_2, we introduce its definition in Lemma B.7.

B PROOFS

A PROOF OF THEOREM 3.3

Theorem B.1 (*Monotonic Policy Space Expanding*)**.** *For any given epoch* e *and* $e+1$*, let* (π_i^e, σ_{-i}^e) *and* $(\pi_i^{e+1}, \sigma_{-i}^{e+1})$ *be Nash equilibrium of* $\boldsymbol{URR}_i^e$ *and* $\boldsymbol{URR}_i^{e+1}$*, respectively, where* $\pi_i^e, \pi_i^{e+1} \in \Pi_i$*,* $\sigma_{-i}^e \in \Delta_{\Pi_{-i}^r}^e$ *and* $\sigma_{-i}^{e+1} \in \Delta_{\Pi_{-i}^r}^{e+1}$*. The utilities of* π_i^e *against opponent strategies* σ_{-i}^e *and* σ_{-i}^{e+1}

satisfies

$$u_i(\pi_i^e, \sigma_{-\mathbf{i}}^{\mathbf{e}}) - u_i(\pi_i^e, \sigma_{-\mathbf{i}}^{\mathbf{e+1}}) \geq 0, \tag{3}$$

where $\Delta^e_{\Pi^r_{-i}}$ *indicates* $\Delta(\Pi^{r,e}_{-i})$. *Especially,* $u_i(\pi_i^e, \sigma_{-i}^e) - u_i(\pi_i^e, \sigma_{-i}^{e+1}) > 0$ *indicates there is a strictly policy space expanding at* $e+1$*, i.e.,* $\pi_{-i}^{e+1} \in \Pi_{-i}^{r,e+1} \setminus \Pi_{-i}^{r,e}$.

Proof. Note that the proof is non-trivial since $\Pi_{-i}^{r,e} \subset \Pi_{-i}^{r,e+1}$, so we cannot directly derive Theorem B.1 with the Nash equilibrium (π_i^e, σ_{-i}^e). Instead, we should combine equilibriums of epoch e and $e+1$. Additionally, the equilibrium considered is mixed-strategy Nash equilibrium. Considering the property of Nash equilibrium, $\forall \pi_i \in \Delta(\Pi_i), u_i(\pi_i, \sigma_{-i}^{e+1}) \leq u_i(\pi_i^{e+1}, \sigma_{-i}^{e+1})$, then we have

$$u_i(\pi_i^e, \sigma_{-i}^{e+1}) \leq u_i(\pi_i^{e+1}, \sigma_{-i}^{e+1}). \tag{4}$$

Analogously, $\forall \sigma_{-i} \in \Delta^{e+1}_{\Pi^r_{-i}}, u_i(\pi_i^{e+1}, \sigma_{-i}^{e+1}) \leq u_i(\pi_i^{e+1}, \sigma_{-i})$, then we have

$$u_i(\pi_i^{e+1}, \sigma_{-i}^{e+1}) \leq u_i(\pi_i^{e+1}, \sigma_{-i}^e). \tag{5}$$

Combining Eq. (4) and (5), we can derive

$$\begin{aligned} u_i(\pi_i^e, \sigma_{-i}^e) - u_i(\pi_i^e, \sigma_{-i}^{e+1}) & \\ \geq u_i(\pi_i^e, \sigma_{-i}^e) - u_i(\pi_i^{e+1}, \sigma_{-i}^{e+1}) & \\ \geq u_i(\pi_i^e, \sigma_{-i}^e) - u_i(\pi_i^{e+1}, \sigma_{-i}^e). & \end{aligned} \tag{6}$$

Since (π_i^e, σ_{-i}^e) is a Nash equilibrium, then there is $u_i(\pi_i^e, \sigma_{-i}^e) - u_i(\pi_i^{e+1}, \sigma_{-i}^e) \geq 0$. Thus $u_i(\pi_i^e, \sigma_{-i}^e) - u_i(\pi_i^e, \sigma_{-i}^{e+1}) \geq 0$. □

Discussion of Exploration Efficiency

We further discuss the exploration efficiency from the theoretical perspective as follows.

Proposition B.2. *EPSRO has higher exploration efficiency than PSRO.*

Proof. Before the proof, we note that $\sigma_i \in \Delta(\Pi_i^r)$ and $\pi_i \in \Delta(\Pi_i)$ for each player $i \in \{1, 2\}$. The NashConv (NC) of PSRO is

$$\begin{aligned} \text{NC}^{psro} &= \max_{\pi_i \in \Delta(\Pi_i)} u_i(\pi_i, \sigma_{-i}) + \max_{\pi_{-i} \in \Delta(\Pi_{-i})} u_{-i}(\sigma_i, \pi_{-i}) \\ &= \max_{\pi_i \in \Delta(\Pi_i)} u_i(\pi_i, \sigma_{-i}) - \max_{\pi_{-i} \in \Delta(\Pi_{-i})} u_i(\sigma_i, \pi_{-i}) \\ &= u_i(\pi_i^*, \sigma_{-i}) - u_i(\sigma_i, \pi_{-i}^*), \end{aligned}$$

and $u_i(\pi_i^*, \sigma_{-i}) \geq u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma_i, \pi_{-i}^*)$.

At each epoch, EPSRO directly learns a tuple of strategies as an equilibrium for each player, i.e., (π_i^*, σ_{-i}^*) and (σ_i^*, π_{-i}^*) are two NEs of (Π_i, Π_{-i}^r), (Π_i^r, Π_{-i}), respectively. Thus, we can apply the same rule as PSRO to derive EPSRO's NashConv which satisfies the following inequality:

$$\begin{aligned} \text{NC}^{epsro} &= u_i(\pi_i^*, \sigma_{-i}^*) - u_i(\sigma_i^*, \pi_{-i}^*) \\ &\leq u_i(\pi_i^*, \sigma_{-i}) - u_i(\sigma_i^*, \pi_{-i}^*) \\ &\leq u_i(\pi_i^*, \sigma_{-i}) - u_i(\sigma_i, \pi_{-i}^*) = \text{NC}^{psro}. \end{aligned}$$

Though $\text{NC}^{epsro} \leq \text{NC}^{psro}$, the equation only holds when (π_i^*, σ_{-i}) and (σ_i, π_{-i}^*) are equilibrium because it requires $\Pi_i^r = \Pi_i$ for each player. Thus, $\text{NC}^{epsro} < \text{NC}^{psro}$ at each epoch before convergence, i.e. EPSRO has higher exploration than PSRO. □

MODELING THE LEARNING AS MWU

In this paper, we argue that the updates of meta-strategies (Algorithm 5) follow the rule of Multiplicative Weights Update (MWU) algorithms. Before the explanation of equivalence, we introduce the definition of MWU as follows.

Definition B.3 (*Multiplicative Weights Update Freund & Schapire (1999)*)**.** Given a game with utility matrix U for the row player, let $c_1, c_2, \dots$ be a sequence of mixed strategies played by the column player. The row player is said to follow MWU if π_{t+1} is updated as follows:

$$\pi_{t+1}(i) = \pi_t(i)\frac{\exp\left(\mu_t {a^i}^\top U c_t\right)}{\sum_{i=1}^n \pi_t(i)\exp\left(\mu_t {a^i}^\top U c_t\right)}, \forall i \in [n] \tag{7}$$

where $\mu_t > 0$ is a parameter, $\pi_0 = [1/n, \dots, 1/n]$ and n is the number of pure strategies (a.k.a. experts).

Then we prove the equivalence in the following Lemma.

Lemma B.4. *Let $\beta = [\beta_1, \dots, \beta_n]$ be the weights vector of the meta strategy played by player $-i$. The player is said to follow MWU if $\sigma_{-i,\beta'}$ is updated as follows*

$$\sigma_{-i,\beta'}(j) = \frac{\exp\left(\beta_j - g_j\right)}{\sum_{j=1}^n \exp\left(\beta_j - g_j\right)}, \tag{8}$$

where g_j is the estimated gradients to item β_j.

Proof. Since the learning for each opponent $-i$ follows the same rule, we mitigate the index $-i$ here.

$$\begin{aligned}
\sigma_{\beta'}(j) &= \frac{\exp\left(\beta_j - g_j\right)}{\sum_{i=1}^n \exp\left(\beta_i - g_i\right)} = \exp\beta_j \frac{\exp -g_j}{\sum_{i=1}^n \exp\beta_i \cdot \exp -g_i} \\
&= \frac{\exp\beta_j}{\sum_{i=1}^n \exp\beta_i} \cdot \frac{\exp -g_j}{\sum_{i=1}^n \frac{\exp\beta_i}{\sum_{k=1}^n \exp\beta_k} \exp -g_i} \\
&= \sigma_\beta(j) \cdot \frac{\exp -g_j}{\sum_{i=1}^n \sigma_\beta(i) \exp -g_i}
\end{aligned}$$

where $g_j = \eta \nabla_{\beta_j} u_{-i}(\pi_i, \sigma_{-i,\beta})$, η is the learning rate. □

A PROOF OF THEOREM 3.6

As introduced in the main content, we consider a warm-start technique to reduce the re-training cost of meta-strategies and save training steps. To find a feasible meta-strategy $\bar{\sigma}'_{-i}$ for warm-starting, we need to ensure that $\bar{\sigma}'_{-i}$ follows two conditions proposed by Noam Brown et al. (2016) Brown & Sandholm (2016), i.e. (1) cannot violate the bound of regret of the latest epoch $e+1$ and (2) cannot violate the Nash equilibrium of the latest epoch $e+1$. In our paper, we choose the average strategies $\bar{\sigma}'_{-i}$ that satisfies $u_i^{e+1}(\bar{\pi}_i, \bar{\sigma}'_{-i}) = u_i^e(\bar{\pi}_i, \bar{\sigma}_{-i})$. Note that the utility function of epoch $e+1$ is different from the one of epoch e since the dimension of opponent strategy is changed with a new introduced best response π_{-i}^{e+1}, i.e. $u_i^e(\bar{\pi}_i, \cdot) \in \mathbb{R}^k$ while $u_i^{e+1}(\bar{\pi}_i, \cdot) \in \mathbb{R}^{k+1}$.

Theorem B.5. *Suppose a substitute policy of $\bar{\sigma}_{-i} \in \Delta^e_{\Pi^r_{-i}}$ is $\bar{\sigma}'_{-i} \in \Delta^{e+1}_{\Pi^r_{-i}}$, and it satisfies $u_i^{e+1}(\bar{\pi}_i, \bar{\sigma}'_{-i}) = u_i^e(\bar{\pi}_i, \bar{\sigma}_{-i})$, we have*

$$\max_{\sigma_{-i} \in \Delta^{e+1}_{\Pi^r_{-i}}} \frac{1}{T}\sum_{t=1}^T \left(u_{-i}^{e+1}(\pi_i^t, \sigma_{-i}) - u_{-i}^{e+1}(\pi_i^t, \bar{\sigma}'_{-i})\right) \le \epsilon_{-i}^T, \tag{9}$$

where ϵ_{-i}^T is the regret bound of epoch $e+1$, $\pi_i^t \in \Delta(\Pi_i)$.

Proof. As introduced in Brown & Sandholm (2016), the regret bound ϵ_{-i}^T in the case of extensive games is

$$\epsilon_{-i}^T = \frac{\sum_{I \in \mathcal{I}_{-i}} \sqrt{p_i^\sigma(I)} \Delta(I) \sqrt{|A(I)|}}{\sqrt{T}},$$

where I is the information set, $\mathcal{I}_{-i}$ indicates the information set for player $-i$, p_i^σ is the joint probability of reaching information set I, $\Delta(I)$ is the range of payoffs reachable by player $-i$, $A(I)$ is the set of policies. As EPSRO frames the game solving in a normal-form game at any epoch k, then we know the size of information set I is always 1. $p_i^\sigma = 1$, $\Delta(I) = U_{max}^k - U_{min}^k$ where U^k is the payoff matrix, $|A(I)| = k$ at epoch k. Considering the horizon of no-regret optimization, we have

$$\max_{\sigma_{-i} \in \Delta_{\Pi_{-i}^r}^{e+1}} \frac{1}{T} \sum_{t=1}^{T} \left(u_{-i}^{e+1}(\pi_i^t, \sigma_{-i}) - u_{-i}^{e+1}(\pi_i^t, \sigma_{-i}^t) \right) \leq \epsilon_{-i}^T, \text{ where } \epsilon_{-i}^T = \frac{\sqrt{k}(U_{max}^k - U_{min}^k)}{\sqrt{T}}. \tag{10}$$

As a feasible substitute strategy for the sequence of σ_{-i}^t, Brown & Sandholm (2016) proved that it should satisfies the following two constraints: *(1) a chosen substitute $\bar{\sigma} = (\bar{\pi}_i^{'}, \bar{\sigma}_{-i}^{'})$ should satisfy $u_i^{e+1}(\bar{\sigma}) + u_{-i}^{e+1}(\bar{\sigma}) \leq 0$; (2) we must ensure that no information set violates the bound on regret guaranteed in (10).* As $u_i^{e+1}(\bar{\pi}_i, \bar{\sigma}_{-i}^{'}) = u_i^e(\bar{\pi}_i, \bar{\sigma}_{-i})$, the substitute strategies hold the condition (1) as they are a NE at epoch e. As for the condition (2), Brown & Sandholm (2016) claimed that *(3) the growth of substitute regret has the same bound as the growth rate of normal regret ((Brown & Sandholm, 2016, Lemma. 2))*; (4) *we can use a substitute strategy regret to prove convergence to a Nash Equilibrium just as we could use normal regret ((Brown & Sandholm, 2016, Lemma. 3)).*

Thus, if we use a substitute strategy $(\bar{\pi}_i, \bar{\sigma}_{-i}^{'})$ that warm-start to T iterations as Eq. 9, and play more $T' \geq 0$ iterations according to Algorithm 5, then we still have:

$$\max_{\sigma_{-i} \in \Delta_{\Pi_{-i}^r}^{e+1}} \left(T \left(u_{-i}^{e+1}(\bar{\pi}_i, \sigma_{-i}) - u_{-i}^{e+1}(\bar{\pi}_i, \bar{\sigma}_{-i}^{'}) \right) + \sum_{t'=1}^{T'} \left(u_{-i}^{e+1}(\pi_i^{t'}, \sigma_{-i}) - u_{-i}^{e+1}(\pi_i^{t'}, \sigma_{-i}^{t'}) \right) \right) \tag{11}$$

$$\leq (T + T')\epsilon_{-i}^{T+T'}.$$

Thus, $(\bar{\pi}_i, \bar{\sigma}_{-i}^{'})$ satisfies the conditions (1) and (2) as a warm-starting strategy. □

We can further quantify the saved number of iteration as follows when warm-starting is introduced.

Corollary B.6. *Assuming EPSRO achieves $\epsilon_i^k = \frac{\sqrt{k}R_k}{\sqrt{T}}$ at epoch k, the substitute strategy for epoch $k+1$ which follows Theorem B.8 will save training iteration as*

$$T^{'} \leq \left| \frac{R_{k+1}}{R_k} \right|^2 \cdot \frac{k+1}{k} \cdot T,$$

where T is the training iteration for at epoch k, $R_{k+1} = U_{max}^{k+1} - U_{min}^{k+1}$, $R_k = U_{max}^k - U_{min}^k$.

Proof. Suppose the warm-starting initialize a strategy to save T' a epoch $k+1$, then we have

$$\frac{R_k^2 \cdot k}{T} \leq \frac{R_{k+1}^2 \cdot k + 1}{T'}. \tag{12}$$

Thus we have $T^{'} \leq \left| \frac{R_{k+1}}{R_k} \right|^2 \cdot \frac{k+1}{k} \cdot T$. Note there always have $R_{k+1} \geq R_k$. □

A PROOF OF LEMMA 3.7

As for the exact bound of regret, we will give a proof in Theorem B.5. Note that the utility function of epoch $e+1$ is different from the one of epoch e since the dimension of opponent strategy is changed with a new introduced best response π_{-i}^{e+1}, i.e. $u_{-i}^{e}(\bar{\pi}_i, \cdot) \in \mathbb{R}^k$ while $u_{-i}^{e+1}(\bar{\pi}_i, \cdot) \in \mathbb{R}^{k+1}$. Thus, before the optimization of β, we need to slightly run a simulation to estimate the item $u_{-i}(\bar{\pi}_i, \pi_{-i}^{e+1})$ in u_{-i}^{e+1}.

Lemma B.7. *Let $k = |\Pi_{-i}^{r,e+1}|$, $\bar{\sigma}_{-i}^{'}$ be parameterized by $\beta_{-i} = [\beta_{-i,1}, \beta_{-i,2}, \ldots, \beta_{-i,k}]$, $\bar{\sigma}_{-i}^{'}(k)$ the k-th item of $\bar{\sigma}_{-i}^{'}$, $x = [\bar{\sigma}_{-i}^{'}(1), \ldots, \bar{\sigma}_{-i}^{'}(k-1)]^\top$, $\bar{l}_{-i}^{e}$ is $-i$'s average loss vector to $\bar{\pi}_i$ at epoch e. Then a feasible initial of β_{-i}^{e+1} could satisfy*

$$\beta_{-i}^{e+1} = \arg\min_{\beta_{-i}} \parallel (x - \bar{\sigma}_{-i})^\top \bar{l}_{-i}^{e} - \bar{\sigma}_{-i}^{'}(k) u_{-i}(\bar{\pi}_i, \pi_{-i}^{e+1}) \parallel_2 - \lambda H(\bar{\sigma}_{-i}^{'}), \tag{13}$$

where $\lambda > 0$, $H(\bar{\sigma}_{-i}^{'})$ is the entropy of $\bar{\sigma}_{-i}^{'}$.

Proof. Considering the error between $\xi = u_{-i}^{e}(\bar{\pi}_i, \bar{\sigma}_{-i}) - u_{-i}^{e+1}(\bar{\pi}_i, \bar{\sigma}_{-i}^{'})$. It can be rewritten as

$$\begin{aligned}
\xi &= -\bar{\sigma}_{-i}^{\top} \bar{l}_{-i}^{e} - \left(-x^\top \bar{l}_{-i}^{e} + \bar{\sigma}_{-i}^{'}(k) u_{-i}(\bar{\pi}_i, \pi_{-i}^{e+1}) \right) \\
&= (x - \bar{\sigma}_{-i})^\top \bar{l}_{-i}^{e} - \bar{\sigma}_{-i}^{'}(k) u_{-i}(\bar{\pi}_i, \pi_{-i}^{e+1}).
\end{aligned}$$

Thus, we can optimize β_{-i} by minimizing the error ξ as $\beta_{-i}^{e+1} = \arg\min_{\beta_{-i}} \parallel \xi \parallel_2$. □

We compute the initial of σ_{-i}^{e+1} as Lemma B.7. Note that there is another solution as $\bar{\sigma}_{-i}^{'}(k) = 0, \bar{\sigma}_{-i} = x$. In that case, the newly introduced policy π_{-i}^{e+1} will have no chance to be sampled, causing biased or even wrong solution. Thus, we consider adding a regularization in Eq. 13 that maximizes the entropy of $\bar{\sigma}_{-i}^{'}$, i.e. $H(\bar{\sigma}_{-i}^{'}) = -\bar{\sigma}_{-i}^{'\top} \log \bar{\sigma}_{-i}^{'}$ and $\beta_{-i}^{e+1} = \arg\min_{\beta_{-i}} \parallel \xi \parallel_2 - \lambda H(\bar{\sigma}_{-i}^{'})$, where $\lambda > 0$.

A PROOF OF THEOREM 3.8

Theorem B.8 (*Regret Bound of EPSRO*)**.** *Let $l_1, l_2, \ldots, l_T$ be a sequence of loss vectors player by an adversary, and $\langle \cdot, \cdot \rangle$ be the dot product, then Algorithm 5 is a no-regret algorithm with*

$$\frac{1}{T}\left(\sum_{t=1}^{T} \langle \sigma_t, l_t \rangle - \min_{\sigma \in \Delta(\Pi_t^r)} \sum_{t=1}^{T} \langle \sigma, l_t \rangle \right) \leq \frac{\sqrt{\log\left[(k+1)k/2\right]}}{\sqrt{2T}}, \text{ where } k \text{ is the size of } \Pi^r.$$

Proof. Note that the loss vector is the opposite of utility at each iteration, i.e., $l_t = -u_t$. The proof follows the approach of Le Cong Dinh et al. (2021). Denote $|T_1|, \ldots, |T_k|$ be the learning horizon of each epoch. During each $|T_j|$, the restricted policy set is denoted as unchanged $\Pi^{r,i}$. In the case of finite k, we have:

$$\sum_{i=1}^{k} |T_i| = T. \tag{14}$$

For each training epoch i, following the regret bound of MWU, we have:

$$\sum_{t=|T^i|+1}^{|T^{i+1}|} \langle \sigma_t, l_t \rangle - \min_{\sigma \in \Delta(\Pi_i^r)} \sum_{t=|T^i|+1}^{|T^{i+1}|} \langle \sigma, l_t \rangle \leq \sqrt{\frac{|T_i| \log |\Pi^{r,i}|}{2}}, \text{ where } |T^i| = \sum_{j=1}^{i} |T_j|. \tag{15}$$

Considering all time horizon, for $i = 1, \ldots, k$, we have:

$$\sum_{i=1}^{k} \sqrt{\frac{|T_i| \log |\Pi^{r,i}|}{2}} \geq \sum_{t=1}^{T} \langle \sigma_t, l_t \rangle - \sum_{i=1}^{k} \min_{\sigma \in \Delta(\Pi^{r,i})} \sum_{t=|T^i|+1}^{|T^{i+1}|} \langle \sigma, l_t \rangle \tag{16}$$

$$\geq \sum_{t=1}^{T} \langle \sigma_t, l_t \rangle - \min_{\sigma \in \Delta(\Pi^{r})} \sum_{i=1}^{k} \sum_{t=|T^i|+1}^{|T^{i+1}|} \langle \sigma, l_t \rangle \tag{17}$$

$$= \sum_{t=1}^{T} \langle \sigma_t, l_t \rangle - \min_{\sigma \in \Delta(\Pi^{r})} \sum_{t=1}^{T} \langle \sigma, l_t \rangle. \tag{18}$$

Note that $i = |\Pi^{r,i}|$, then $\sum_{i=1}^{k} \sqrt{|T_i| \log |\Pi^{r,i}|} \leq \sum_{i=1}^{k} \sqrt{T \log(i)}$. As the sum of logarithms is concave, so we have $\sum_{i=1}^{k} \sqrt{T \log(i)} \leq \sqrt{T \log\left(\sum_{i=1}^{k} i\right)} = \sqrt{T \log\left[(k+1)k/2\right]}$ then we have

$$\frac{\sqrt{\log\left[(k+1)k/2\right]}}{\sqrt{2T}} \geq \frac{1}{T}\left(\sum_{t=1}^{T} \langle \sigma_t, l_t \rangle - \min_{\sigma \in \Delta(\Pi^{r})} \sum_{t=1}^{T} \langle \sigma, l_t \rangle\right).$$

□

In addition, we provide a comparison of regret bound on existing solvers in Appendix G.1.

A Proof of Theorem 3.9

Before the proof of convergence rate of EPSRO, we want to clarify that the reinforcement learning procedure for the best response π_i should be an equivalence of MWU. Fortunately, many reinforcement learning algorithms can be treated like that as their policy function is modeled as a Boltzman distribution Cen et al. (2022). Thus, EPSRO can implement such a procedure with algorithms like Soft Actor Critic Haarnoja et al. (2018), Soft Q-learning Haarnoja et al. (2017), DQN with Boltzman sampling Yang et al. (2018) and others. With this condition, we can apply the Theorem B.5 to π_i and then derive the convergence rate as follows.

Theorem B.9 (*Convergence Rate of EPSRO*)**.** *Let k, N denote the size of restricted policy sets Π^r_{-i} and Π_i. Then the learning of Algorithm 3 will converge to the Nash equilibrium with the rate:*

$$\epsilon_T = \sqrt{\frac{\log\left[(k+1)k/2\right]}{2T}} + \sqrt{\frac{\log\left[(N+1)N/2\right]}{2T}}.$$

Proof. Using the regret bound of Theorem B.8 we have:

$$\max_{\pi_i} \frac{1}{T}\left(\sum_{t=1}^{T} u_i(\pi_i, \sigma^t_{-i}) - u_i(\pi^t_i, \sigma^t_{-i})\right) \leq \epsilon_i,$$

$$\max_{\sigma_{-i}} \frac{1}{T}\left(\sum_{t=1}^{T} u_{-i}(\pi^t_i, \sigma_{-i}) - u_{-i}(\pi^t_i, \sigma^t_{-i})\right) \leq \epsilon_{-i},$$

where $\epsilon_i = \sqrt{\frac{\log\left[(N+1)N/2\right]}{2T}}$ and $\epsilon_{-i} = \sqrt{\frac{\log\left[(k+1)k/2\right]}{2T}}$. From the above inequalities and $u_{-i}(\cdot,\cdot) = -u_i(\cdot,\cdot)$ in zero-sum games, we can derive that

$$u_i(\bar{\pi}_i, \bar{\sigma}_i) \geq \min_{\sigma_{-i} \in \Delta(\Pi^r_{-i})} u_i(\bar{\pi}_i, \sigma_{-i}) \geq \frac{1}{T}\sum_{t=1}^{T} u_i(\pi_t, \sigma_t) - \epsilon_{-i}$$

$$\geq \max_{\pi_i} u_i(\pi_i, \bar{\sigma}_{-i}) - \epsilon_i - \epsilon_{-i}.$$

By symmetry, we have

$$u_i(\bar{\pi}_i, \bar{\sigma}_i) \leq \max_{\pi_i} u_i(\pi_i, \bar{\sigma}_{-i}) \leq \frac{1}{T}\sum_{t=1}^{T} u_i(\pi^t_i, \sigma^t_{-i}) + \epsilon_i$$

$$\leq \min_{\sigma_{-i}} u_i(\bar{\pi}_i, \sigma_{-i}) + \epsilon_{-i} + \epsilon_i.$$

Thus, with $\epsilon_T = \epsilon_i + \epsilon_{-i}$, we have

$$\max_{\pi_i} u_i(\pi_i, \bar{\sigma}_{-i}) - \epsilon_T \leq u_i(\bar{\pi}_i, \bar{\sigma}_{-i}) \leq \min_{\sigma_{-i}} u_i(\bar{\pi}_i, \sigma_{-i}) + \epsilon_T.$$

By definition, we have $(\bar{\pi}_i, \bar{\sigma}_{-i})$ is ϵ_T-Nash equilibrium. □

C Implementation and Hyper-parameter Selection

Learning Meta-strategies and Best-response. We introduce the algorithm for learning meta-strategies in Algorithm 5. We train best response policies with off-policy reinforcement learning algorithm, DQN. For all involved PSRO baselines in our paper, the selected implementation for best response learning is DQN.

Parameter Selection. We keep the consistency on the implementation of policy support in each PSRO-based method in this paper. Expressly, the network is set to 4 Dense layers, 256 units each. The learning rate for reinforcement learning policy is set to 0.01. For the hyper-parameters of meta-strategy, the window size L is set to 100 episodes, the learning rate is 0.01 for Kuhn Poker, and 0.005 for other environments. The number of parallel workers is set to 4 for EPSRO and P-PSRO in all experiments.

D Baselines

Self-Play. Self-play is an open-ended learning algorithm for multi-agent reinforcement learning Hernandez et al. (2019). In the training process, self-play generates a sequence of policies and keeps training policies against the newest opponents. This algorithm outperforms in some classic games, such as Go and Chess. However, self-play fails in nontransitive games.

Policy Space Response Oracles (PSRO). PSRO algorithm is well described above in previous sections. It provides an iterative solution to solve the approximation of Nash equilibrium for large games Lanctot et al. (2017). PSRO iteratively trains new policies against a meta-strategy of opponent population and expends policy populations with the current well-trained policy.

Rectified PSRO (PSRO-rN). PSRO-rN is a variant of PSRO that aims to solve non-transitive zero-sum games Balduzzi et al. (2019), such as rock-paper-scissors. This algorithm involves rectified Nash response to construct adaptive sequences of objectives for non-transitive games. Policies in PSRO-rN only train against others that they already beat.

Mixed Oracles. Mixed Oracles is another variant of PSRO that aims to improve computational efficiency by reducing training costs Smith et al. (2020). At each iteration, it utilizes knowledge of former iterations, thus only needing to train current policies against the newest opponent.

Pipeline-PSRO (P-PSRO). To further accelerate the training process of PSRO, P-PSRO is proposed to parallelize the training process McAleer et al. (2020). Compared to other parallel algorithms, such as DHC, which fail to converge in some cases, P-PSRO maintains a parallel pipeline of learning workers with convergence guarantees.

E Environment Details

We introduce more details about the random symmetric games and mulit-agent gathering environments here.

Random Symmetric Games. McAleer et al. McAleer et al. (2020) introduce the games to investigate the performance of PSRO-based methods in high-dimensional symmetric games (SymGame). In this experiment, we generated random symmetric zero-sum matrices with different dimension n. For a given matrix, elements in the upper triangle are distributed uniformly: $\forall i < j \leq n, a_{i,j} \sim \text{UNIFORM}(-1, 1)$ and for the lower triangle, the elements are set to be the negative of its diagonal

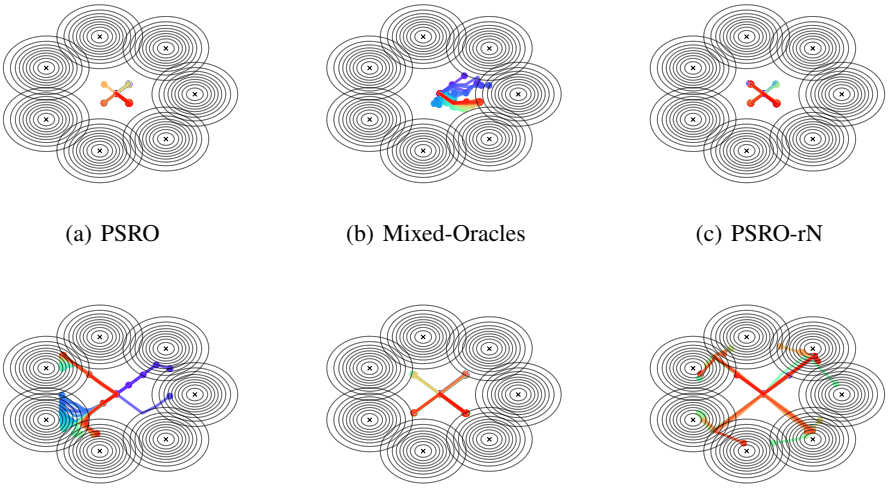
counterpart: $\forall j < i \leq n, a_{i,j} = -a_{j,i}$. The diagonal elements are equal to zero: $a_i, i = 0$. The matrix defines the utility of two pure strategies to the row player. A strategy $\pi \in \Delta^n$ is a distribution over the n pure strategies of the game given by the rows (or equivalently, columns) of the matrix.

Multi-agent Gathering. We introduce two multi-agent gathering environments in this paper, the *Gathering Small* and the *Gathering Open*. For each environment, the agent number is set to 2, and the difference between them is that the *Gathering Open* has a much bigger map than *Gathering Small*. Therefore, the agents need to explore a higher dimensional state space in the *Gathering Open* than the smaller one.

F Additional Experimental Results

F.1 Non-transitive Mixture Game

We test six PSRO-based algorithms in the *non-transitive mixture game* to investigate the exploration efficiency. We also introduce the results of NashConv to compare the performance. We find that EPSRO outperforms all other algorithms in this game, and performs the highest exploration efficiency.



(a) PSRO (b) Mixed-Oracles (c) PSRO-rN

(d) P-PSRO (e) NEPSRO (f) EPSRO

Figure 7: Exploration trajectories on *Non-transitive Mixture Games*. The more trajectories close to the centers of Gaussian, the higher the exploration efficiency of the algorithm. EPSRO outperform all selected baselines since it explored all centers.

F.2 Goofspiel

Goofspiel is a more complex task than Leduc poker with 13 cards and turn-based moves. Considering the high cost of traversing a game tree to compute the exploitability, we evaluated the algorithm with the same approach as in Multi-agent gathering tasks, i.e., calculating the normalized score to the final policy population of naive PSRO. For the comparison, we choose Pipeline PSRO since it outperforms other baselines in all environments. We explain the meaning of score as: 0 for a tie, < 0 for a loss and > 0 for a win. In particular, -1 means that an algorithm is

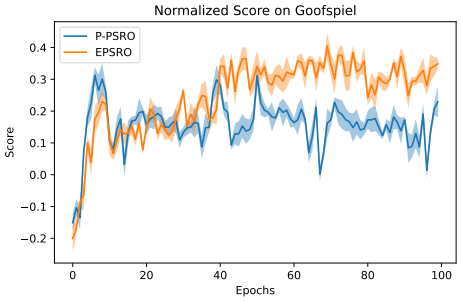
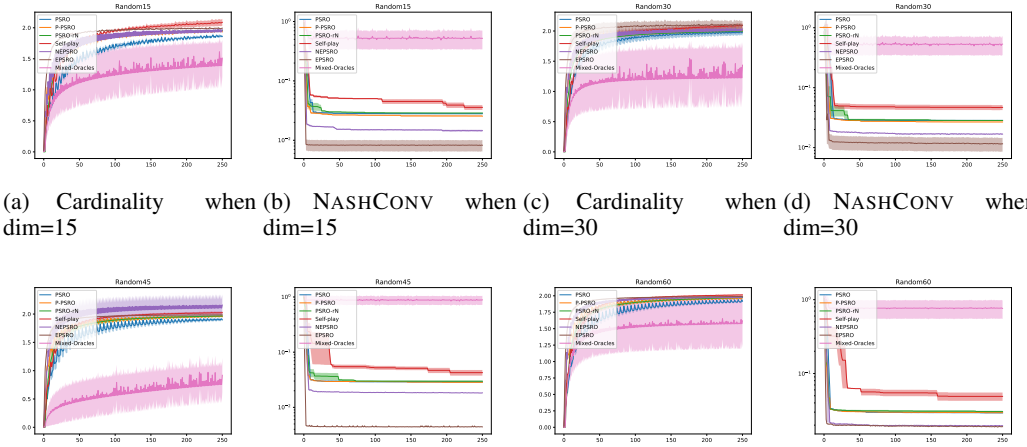


Figure 8: The average score of growing population on Goofspiel.

dominated by naive PSRO, and 1 means that an algorithm dominates naive PSRO. We evaluate the score of the grouping population (size from 1 to 100) and report the results in Figure 8. The results show that the EPSRO achieves better performance than P-PSRO and lower exploitability. For the scores of the final population, P-PSRO and EPSRO achieve 0.244 and 0.332, respectively.

F.3 RANDOM SYMMETRIC MATRIX GAME

We compare the NASHCONV over iteration of EPSRO with PSRO, P-PSRO, Rectified PSRO, Self-Play, Mixed-Oracles and naive EPSRO without pipeline training (NEPSRO). We run 5 experiments for each set of dimension. The dimensions of size including 15, 30, 45, 60 and 120. The learning rates is set to 0.5, and 4 parallel threads for parallel algorithms. We find that EPSRO performs better than all other algorithms in every dimension setting.



(a) Cardinality when dim=15 (b) NASHCONV when dim=15 (c) Cardinality when dim=30 (d) NASHCONV when dim=30

(e) Cardinality when dim=45 (f) NASHCONV when dim=45 (g) Cardinality when dim=60 (h) NASHCONV when dim=60

Figure 9: Comparison on NASHCONV and cardinality for random symmetric matrix games with different dimensions.

F.4 MULTI-AGENT GATHERING

Multi-agent Gathering environments (MAG) have complex state space than the other games, so that it is highly computation expensive to traverse the game tree to compute the NASHCONV. Instead, we evaluate all algorithms with a fixed policy set. In our experiments, the fixed policy set is generated with PSRO, i.e. Π^{PSRO}. We list the pseudo-code for evaluation in Algorithm 7.

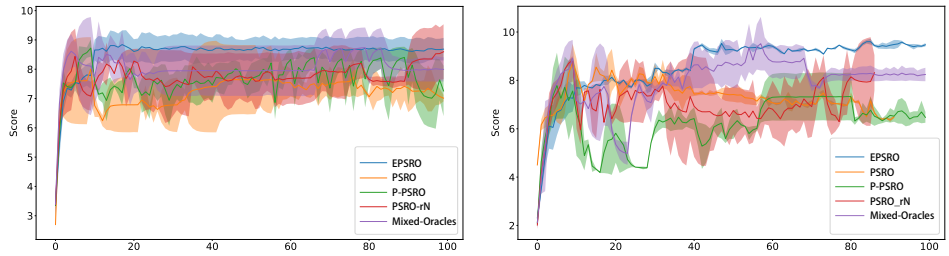
Algorithm 7: EMPIRICAL EVALUATION ON MAG

Input: a policy set Π^{TEST} of evaluated algorithm; Π^{PSRO}; an empty matrix $M \in \mathbb{R}^{|\Pi^{\text{TEST}}| \times |\Pi^{\text{PSRO}}|}$

for *each policy* π_i^{TEST} *in* Π^{TEST} **do**
  for *each policy* π_j^{PSRO} *in* Π^{PSRO} **do**
    Run 50 episodes to evaluate $M_{i,j} = u_i(\pi_i^{\text{TEST}}, \pi_j^{\text{PSRO}})$
  Compute score of $\sigma_{1:i}^{\text{TEST}}$ as $\text{SCORE}(\sigma_{1:i}^{\text{TEST}}) = \sigma_{1:i}^{\text{TEST}} M_{1:i} \left[\sigma^{\text{PSRO}}\right]^T$

Output: a list of score $\text{SCORE}(\Pi^{\text{TEST}}) = \{\text{SCORE}(\sigma_{1:i}^{\text{TEST}}) \mid i = 1, \ldots, |\Pi^{\text{TEST}}|\}$ for Π^{TEST}

$\sigma_{1:i}^{\text{TEST}}$ in Algorithm 7 indicates a meta-strategy composed of $\pi_1, \ldots, \pi_i$, and $M_{1:i}$ is a sub matrix with row 1 to i. We present $\text{SCORE}(\Pi^{\text{TEST}})$ of each algorithm in Figure 5. The curve of Self-Play is not included because we keep only two policies (one for the opponent, another for training) in our implementation.



(a) Gathering Small (b) Gathering Open

Figure 10: The score of algorithms on two multi-agent gathering environments. The horizon axis indicates the number of training iterations. As reported in this Figure, EPSRO performs better than other algorithms.

In the training stage, we set the number of simulations for each joint policy as 100, and 10000 episodes to optimize each policy. Except for EPSRO, the training time of each algorithm on the Gathering Small is about 24 hours, and 26 hours for the Gathering Open, while the training time for EPSRO is about 8 hours.

G Convergence Proof of Average Substitute Strategies

Theorem G.1. *((Brown & Sandholm, 2016, Theorem 1)). In a two-player zero-sum game, if* $\frac{R_i^T}{T} \leq \epsilon_i$ *for both player* $i \in \{1, 2\}$*, then* $\bar{\sigma}$ *is a* $(\epsilon_1 + \epsilon_2)$*-equilibrium.*

Proof. Follow the proof approach of Waugh et al. (2009). From (5), we have that

$$\max_{\sigma \in \Delta} \frac{1}{T} \left(\sum_{t=1}^{T} u_i(\sigma_i', \sigma_{-i}^t) - u_i(\sigma_i^t, \sigma_{-i}^t) \right) \leq \epsilon_i. \tag{19}$$

Since σ_i' is the same on every iteration, this becomes

$$\max_{\sigma' \in \Delta} u_i(\sigma_i', \bar{\sigma}_{-i}^T) - \frac{1}{T} \sum_{t=1}^{T} u_i(\sigma_i^t, \sigma_{-i}^t) \leq \epsilon_i. \tag{20}$$

Since $u_i(\sigma) = u_2(\sigma)$, if we sum Equation 20 for both players

$$\max_{\sigma_1' \in \Delta} u_1(\sigma_1', \bar{\sigma}_2^T) + \max_{\sigma_2' \in \Delta} u_2(\bar{\sigma}_1^T, \sigma_2') \leq \epsilon_1 + \epsilon_2, \tag{21}$$

$$\max_{\sigma_1' \in \Delta} u_1(\sigma_1', \bar{\sigma}_2^T) - \min_{\sigma_2' \in \Delta} u_1(\bar{\sigma}_1^T, \sigma_2') \leq \epsilon_1 + \epsilon_2. \tag{22}$$

Since $u_1(\bar{\sigma}_1^T, \bar{\sigma}_2^T) \geq \min_{\sigma_2' \in \Delta} u_1(\bar{\sigma}_1^T, \sigma_2')$, so we have $\max_{\sigma_1' \in \Delta} u_1(\sigma_1', \bar{\sigma}_2^T) - u_1(\bar{\sigma}_1^T, \bar{\sigma}_2^T) \leq \epsilon_1 + \epsilon_2$. By symmetry, this is also true for player 2. Therefore, $\langle \bar{\sigma}_1^T, \bar{\sigma}_2^T \rangle$ is a $(\epsilon_1 + \epsilon_2)$-equilibrium. □

When warm starting, we can calculate this value because we set $\bar{\sigma}^T = \sigma$. However we cannot calculate $\sum_{t=1}^{T} u_i(\sigma^t)$ because we did not define individual strategies played on each iteration. Fortunately, it turns out we can substitute another value we refer to as $Tu_i(\bar{\sigma})$, chosen from a range of acceptable options. To see this we first observe that the value of $\sum_{t=1}^{T} u_i(\sigma^t)$ is not relevant to the proof of Theorem G.1. Specifically, in Eq. 21, we see it cancels out. Thus, if we choose $(\bar{\pi}_i, \bar{\sigma}_{-i})$ such that satisfies it. Since $\max_{\pi} u_i(\pi, \bar{\sigma}_{-i}^T) \geq u_i(\bar{\pi}_i, \bar{\sigma}_{-i})$ and $\max_{\sigma_{-i}} u_{-i}(\bar{\pi}_i, \sigma_{-i}) \geq u_{-i}(\bar{\pi}_i, \bar{\sigma}_{-i})$. Thus $(\bar{\pi}_i, \bar{\sigma}_{-i})$ is a feasible warm-starting strategy.

G.1 Comparison of Regret Bound

We compare EPSRO with existing solvers in the Table from the perspective of time complexity or regret bound.

Method	**Time Complexity ($\tilde{\mathcal{O}}$) / Regret Bound ($\mathcal{O}$)**
Linear Programming van den Brand (2020)	$\tilde{\mathcal{O}}(n \exp(-T/n^2.38))$
Fictitious Play Leslie & Collins (2006)	$\tilde{\mathcal{O}}(T^{-1/(n+m-2)})$
Double Oracle McMahan et al. (2003)	$\tilde{\mathcal{O}}(n \exp(-T/n^3.88))$
Multipli. Weight UpdateFreund & Schapire (1999)	$\mathcal{O}(\sqrt{\log n/T})$
Policy Response Oracles Lanctot et al. (2017)	$\times$
Online Double Oracle Le Cong Dinh et al. (2021)	$\mathcal{O}(\sqrt{k \log k/T})$
EPSRO	$\boldsymbol{\mathcal{O}(\sqrt{\log[(k^2+k)/2]/T})}$