Appendix for "Unifying Behavioral and Response Diversity for Open-ended Learning in Zero-sum Games"

Table of Contents

A Full Proof of Theorems .................................................. 1
A.1 Proof of Theorem 1 .................................................. 1
A.2 Proof of Theorem 2 .................................................. 2
A.3 Proof of Theorem 3 .................................................. 2
B PE Approximation via PSRO .............................................. 4
C Additional Experimental Results on Google Research Football ............................................. 4
D Environment Details
  D.1 Environment Details of Non-Transitive Mixture Model ............................................. 5
  D.2 Environment Details of Google Research Football ............................................. 5
E Hyperparameter Settings
  E.1 Hyperparameter Settings for Real-World Games ............................................. 6
  E.2 Hyperparameter Settings for Non-Transitive Mixture Model ............................................. 6
  E.3 Hyperparameter Settings for Google Research Football ............................................. 6
F Ablation Studies
  F.1 Real-World Games ............................................. 7
  F.2 Non-Transitive Mixture Model ............................................. 7
  F.3 Google Research Football ............................................. 9
G Simplified Optimization Method for Unified Diverse Response ............................................. 9

A Full Proof of Theorems

A.1 Proof of Theorem 1

To prove Theorem 1, we need the help of the following Lemma

Lemma 1. If \( P_{X,Y} = P_X P_{Y|X} \) and \( Q_{X,Y} = Q_X P_{Y|X} \) then

\[
D_f(P_{X,Y}||Q_{X,Y}) = D_f(P_X||Q_X). \tag{14}
\]

Proof. See Proposition 7.1 in [3].

Now we can prove our Theorem 1.

Proof. For games with only one step (normal-form games, functional-form games), there is only one fixed state. Therefore, the distribution of state-action is equivalent to the distribution of the action. Formally, for \( \rho_{\pi'_i, \pi_{E_{-i}}} \), we have:

\[
\rho_{\pi'_i, \pi_{E_{-i}}}(s, a) = (\pi'_i, \pi_{E_{-i}})(s, a) = \pi'_i(a_i|s)\pi_{-i}(a_{-i}|s), \tag{15}
\]

where the second equation comes from the assumption that policies are independent. Similarly, for \( \rho_{\pi_i, \pi_{E_{-i}}} \), we also have:

\[
\rho_{\pi_i, \pi_{E_{-i}}}(s, a) = (\pi_i, \pi_{E_{-i}})(s, a) = \pi_i(a_i|s)\pi_{-i}(a_{-i}|s). \tag{16}
\]
Therefore, with the help of Lemma 1, we have:

\[
D_f(\rho_{\pi_i', \pi_{-i}} \| \rho_{\pi_i, \pi_{-i}}) = E_{s_0 \sim \eta(s)}[D_f(\pi'_i | \pi_{-i})] = E_{s_0 \sim \eta(s)}[D_f(\pi'_i | s_0) || \pi_i(\cdot|s_0)].
\]  

(17)

\[\square\]

A.2 Proof of Theorem 2

Let us restate our Theorem 2:

**Theorem 2.** For a given empirical payoff matrix \( \mathbf{A} \in \mathbb{R}^{M \times N} \) and the reward vector \( \mathbf{a}_{M+1} \) for policy \( \pi_i^{M+1} \), the lower bound of \( \text{Div}_\text{occ} \) is given by:

\[
\text{Div}_{\text{rew}}(\pi_i^{M+1}) \geq \sigma_{\text{min}}^2(\mathbf{A})(1 - \mathbf{1}^T(\mathbf{A}^T)^{\dagger}\mathbf{a}_{M+1})^2 + \| (\mathbf{I} - \mathbf{A}^T(\mathbf{A}^T)^{\dagger})\mathbf{a}_{M+1} \|^2, \tag{18}
\]

where \((\mathbf{A}^T)^{\dagger}\) is the Moore–Penrose pseudoinverse of \(\mathbf{A}^T\), and \(\sigma_{\text{min}}(\mathbf{A})\) is the minimum singular value of \(\mathbf{A}\).

**Proof.**

\[
\min_{\mathbf{1}^T \beta = 1} \| \mathbf{A}^T \beta - \mathbf{a}_{M+1} \|^2 = \min_{\mathbf{1}^T \beta = 1, \beta \geq 0} \| \mathbf{A}^T \beta - \mathbf{A}^T(\mathbf{A}^T)^{\dagger}\mathbf{a}_{M+1} \|^2 + \| (\mathbf{I} - \mathbf{A}^T(\mathbf{A}^T)^{\dagger})\mathbf{a}_{M+1} \|^2
\]

\[
\geq \min_{\mathbf{1}^T \beta = 1, \beta \geq 0} \| \mathbf{A}^T \beta - \mathbf{A}^T(\mathbf{A}^T)^{\dagger}\mathbf{a}_{M+1} \|^2 + \| (\mathbf{I} - \mathbf{A}^T(\mathbf{A}^T)^{\dagger})\mathbf{a}_{M+1} \|^2
\]

\[
= \sigma_{\text{min}}^2(\mathbf{A}) \min_{\mathbf{1}^T \beta = 1} \| \beta - (\mathbf{A}^T)^{\dagger}\mathbf{a}_{M+1} \|^2 + \| (\mathbf{I} - \mathbf{A}^T(\mathbf{A}^T)^{\dagger})\mathbf{a}_{M+1} \|^2
\]

\[
= \sigma_{\text{min}}^2(\mathbf{A})(1 - \mathbf{1}^T(\mathbf{A}^T)^{\dagger}\mathbf{a}_{M+1})^2 + \| (\mathbf{I} - \mathbf{A}^T(\mathbf{A}^T)^{\dagger})\mathbf{a}_{M+1} \|^2,
\]

where the first equation comes from that we decompose \(\mathbf{a}_{M+1}\) into \(\mathbf{A}^T(\mathbf{A}^T)^{\dagger}\mathbf{a}_{M+1} + (\mathbf{I} - \mathbf{A}^T(\mathbf{A}^T)^{\dagger})\mathbf{a}_{M+1}\). The last equation comes from the analytic calculation of \(\min_{\mathbf{1}^T \beta = 1} \| \beta - (\mathbf{A}^T)^{\dagger}\mathbf{a}_{M+1} \|^2\) using Lagrangian. \(\square\)

A.3 Proof of Theorem 3

Now let us first restate the propositions.

**Proposition 1.** If \( N = 1 \) and the underlying game \( \phi_i(\cdot, \cdot) \) is a symmetric two-player zero-sum game, PE is equivalent to exploitability.

To prove this, let us prove the following Lemma 2.

**Lemma 2.** For any policy \( \pi \) in two-player symmetric zero-sum games:

\[
\phi_i(\pi, \pi) = 0. \tag{19}
\]

**Proof.** To begin with, due to the assumption that the game is symmetric, we get:

\[
\phi_i(\pi_i, \pi_{-i}) = \phi_{-i}(\pi_{-i}, \pi_i). \tag{20}
\]

Since the game is also zero-sum, we have:

\[
\phi_{-i}(\pi_{-i}, \pi_i) = -\phi_i(\pi_{-i}, \pi_i). \tag{21}
\]

By combining Equation 20 and Equation 21, for any \( \pi_i \) and \( \pi_{-i} \), we get:

\[
\phi_i(\pi_i, \pi_{-i}) + \phi_i(\pi_{-i}, \pi_i) = 0. \tag{22}
\]

Let \( \pi_i = \pi_{-i} = \pi \), we get what we need to prove:

\[
\phi_i(\pi, \pi) = 0. \tag{23}
\]

\(\square\)
Now we can begin proof of our proposition.

Proof. To prove this theorem, we need a further assumption that PSRO maintains only one population for two-player symmetric game, which is a quite common practice. Therefore, the joint Nash aggregated policy satisfies that $\pi_E = (\pi_i, \pi_{-i})$ satisfies $\pi_i = \pi_{-i}$. Therefore, with the help of Lemma 2:

$$\phi_i(\pi_i, \pi_{-i}) = \phi_{-i}(\pi_{-i}, \pi_i) = 0.$$  \hfill (24)

Furthermore, exploitability for symmetric zero-sum game can be written as:

$$\text{Expl}(\pi_E) = \sum_{i=1}^{2} \max_{\pi'_i} \phi_i(\pi'_i, \pi_{-i}) - \phi_i(\pi_i, \pi_{-i})$$  \hfill (25)

$$= \sum_{i=1}^{2} \max_{\pi'_i} \phi_i(\pi'_i, \pi_{-i})$$  \hfill (26)

$$= 2 \max_{\pi'_i} \phi_i(\pi'_i, \pi_{-i}),$$  \hfill (27)

where the last equation comes from the symmetry of the game and $\pi_i = \pi_{-i}$.

For PE, it is calculated as:

$$\text{PE}(\{\pi_i\}) = \min_{\pi_{-i}} \phi_i(\pi_i, \pi_{-i})$$  \hfill (28)

$$= - \max_{\pi_{-i}} \phi_i(\pi'_i, \pi_i)$$  \hfill (29)

$$= -\frac{1}{2} \text{Expl}(\pi_E).$$  \hfill (30)

The second to last equation comes from Equation 22, and the last equation is due to the assumption that $\pi_i = \pi_{-i}$.

**Proposition 2.** If there are two populations $\Psi_i$, $\Omega_i$ and $\Psi_i \subseteq \Omega_i$, then $\text{PE}(\Omega_i) \leq \text{PE}(\Psi_i)$, while the relationship for exploitability of the Nash aggregated policies of $\Psi_i$ and $\Omega_i$ may or may not hold.

Proof. We begin with proof of the monotonicity of PE. W.o.l.g. let us assume that $\Psi_i = \{\pi_i^k\}_{k=1}^M$, $\Omega_i = \{\pi_i^k\}_{k=1}^N$, where $M \leq N$. Then for the population effectivity of $\Omega_i$:

$$\text{PE}(\Omega_i) = \min_{\pi_{-i}} \max_{1 \leq i \leq N} \sum_{k=1}^{N} \alpha_k \phi_i(\pi_i^k, \pi_{-i})$$  \hfill (31)

where $\alpha = (\alpha_1, \ldots, \alpha_N)^T$. Let $\alpha_i = 0$ for $M + 1 \leq i \leq N$, then we get:

$$\min_{\pi_{-i}} \max_{1 \leq i \leq M} \sum_{k=1}^{N} \alpha_k \phi_i(\pi_i^k, \pi_{-i}) \geq \min_{\pi_{-i}} \max_{1 \leq i \leq M} \sum_{k=1}^{M} \alpha_k \phi_i(\pi_i^k, \pi_{-i})$$  \hfill (32)

$$= \min_{\pi_{-i}} \max_{1 \leq i \leq M} \sum_{k=1}^{M} \alpha_k' \phi_i(\pi_i^k, \pi_{-i})$$  \hfill (33)

$$= \text{PE}(\Psi_i).$$  \hfill (34)

Now we conclude that:

$$\text{PE}(\Omega_i) \geq \text{PE}(\Psi_i).$$  \hfill (35)

Regarding exploitability, the analysis comes from our Example 1. Suppose player 1 holds the population $\Psi_1 = \{ \left[ \begin{array}{c} \frac{1}{2} \\ \frac{1}{2} \end{array} \right] \}$ and $\Psi_2 = \Psi_1$. Apparently, the Nash aggregated joint policy is

$$\pi_E^\Psi = \left[ \begin{array}{c} \left[ \begin{array}{c} \frac{1}{2} \\ \frac{1}{2} \end{array} \right] \\ \left[ \begin{array}{c} \frac{1}{2} \\ \frac{1}{2} \end{array} \right] \end{array} \right],$$  \hfill (36)
since there is only one policy in each players’ population.

Now consider another two populations $Q_1 = \{\begin{bmatrix} 1 \\ \frac{2}{3} \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\}$ and $Q_2 = Q_1$. Then the Nash aggregated joint policy is given by:

$$\pi^Q_E = \left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right). \quad (37)$$

With simple derivations, the exploitability for $\pi^P_E$ and $\pi^Q_E$ is:

$$\text{Expl}(\pi^P_E) = 1. \quad (38)$$

$$\text{Expl}(\pi^Q_E) = 2. \quad (39)$$

Now we can conclude that for player 1 and player 2, even their population both get strictly enlarged ($P_1 \subseteq Q_1$ and $P_2 \subseteq Q_2$), they become more exploitable: $\text{Expl}(\pi^Q_E) \geq \text{Expl}(\pi^P_E)$. \hfill \square

**Proposition 3.** If the underlying game $\phi_i(\cdot, \cdot)$ is a matrix game, then computing PE is still solving a matrix game.

**Proof.** The proof follows some simple algebraic manipulations. Note that for matrix games, the reward function is given by:

$$\phi_i(\pi_i, \pi_{-i}) = \pi_i^T P \pi_{-i}, \quad (40)$$

where $P$ is the payoff matrix. Then for population effectivity:

$$\text{PE}(\{\pi^k_i\}_{k=1}^N) = \min_{\pi_{-i}} \max_{\alpha_i \geq 0} \sum_{k=1}^N \alpha_k \phi_i(\pi^k_i, \pi_{-i}) \quad (41)$$

$$= \min_{\pi_{-i}} \max_{\alpha_i \geq 0} \sum_{k=1}^N \alpha_k (\pi^k_i)^T P \pi_{-i} \quad (42)$$

$$= \min_{\pi_{-i}} \max_{\alpha_i \geq 0} \alpha^T (\pi^1_i, \cdots, \pi^N_i)^T P \pi_{-i}. \quad (43)$$

Therefore, solving PE is still a matrix game with payoff matrix $(\pi^1_i, \cdots, \pi^N_i)^T P$. \hfill \square

**B PE Approximation via PSRO**

We have already mentioned the tractability for PE of matrix games in Theorem 3. However, for more general games, solving exact PE is still very hard. Since PE is still computing an NE, we here propose using PSRO again as the approximate solver. The only difference is that population of player $i$ is already fixed by $P_i = \{\pi^k_i\}_{k=1}^N$. Therefore, during iterations of PSRO, only player $-i$ needs to enlarge its population. We now outline the algorithm $\text{PE}(n)$ in Algorithm 2. The intuition behind this algorithm is that the opponent is enlarging its population gradually and trying to exploit $P_i$. Therefore, the metric of PE is actually testing how exploitable a population is by gradually constructing a real adversarial! The opponent strength $n$ essentially represents how accurate each best response is.

**C Additional Experimental Results on Google Research Football**

In real world games, we expect our models are robust enough to defeat all the previous models in the model pool and show diverse behaviors to better exploit the opponents. To further evaluate the performance of all the models generated with different methods during the training process, we rank all the models with the Elo rating system [2], and the results are shown in Figure 4. It can be found
Algorithm 2 PE(n)

1: **Input:** Population $\mathcal{P}_i = \{\pi^k_i\}_{k=1}^N$, Opponent Strength $n$, Number of iterations $T$
2: $\mathcal{P}_{-i} \leftarrow$ Initialize opponent population with one random policy
3: $A_{\mathcal{P}_i \times \mathcal{P}_{-i}} \leftarrow$ Initialize empirical payoff matrix
4: for $t = 1$ to $T$ do
5: $\sigma_i, \sigma_{-i} \leftarrow$ Nash Equilibrium on $A_{\mathcal{P}_i \times \mathcal{P}_{-i}}$
6: $\pi^t_{-i}(\theta) \leftarrow$ Initialize a new opponent policy
7: $\theta^* \leftarrow$ Train $\pi^t_{-i}(\theta)$ against mixture of $\sum_j \sigma^j_{-i} \pi^j_i$ with $n$ gradient steps
8: $\mathcal{P}_{-i} \leftarrow \mathcal{P}_{-i} \cup \{\pi^t_{-i}(\theta^*)\}$
9: Compute missing entries in the evaluation matrix $A_{\mathcal{P}_i \times \mathcal{P}_{-i}}$
10: end for
11: **Output:** Nash value on $A_{\mathcal{P}_i \times \mathcal{P}_{-i}}$

that models generated by PSRO w. BD&RD outperform other methods and reaches an Elo score of around 1300. This implies that a combination of BD and RD will essentially contribute to the generation of diverse opponents during the training, so that the final models will be more robust and less exploitable since they are more likely to be offered strong diverse opponents and have the chance to learn to defeat them. Additionally, we also visualize the policies of our methods when playing against other baseline methods and verify our methods truly generate diverse behaviors (see https://sites.google.com/view/diverse-psro/).

Figure 4: The Elo scores of all the models generated by different methods. Shaded areas represent the standard deviation.

D Environment Details

D.1 Environment Details of Non-Transitive Mixture Model

In our experiments, we set $l = 4$ and use 9 Gaussian distributions in the plane. This environment involves both transitivity and non-transitivity because of the delicately designed $S$ in the reward function $\phi_i(\pi_i, \pi_{-i}) = \pi_i^\top S \pi_{-i} + 1^\top (\pi_i - \pi_{-i})$. $S$ is constructed by:

$$S[i][k] = \begin{cases} 
0 & \text{if } k = i \\
1 & \text{if } 0 < (k - i) \text{ mod } (2l + 1) \leq l \\
-1 & \text{otherwise}
\end{cases}$$

D.2 Environment Details of Google Research Football

*Google Research Football* (GRF) [4] is a physics-based 3D simulator where agents can be trained to play football. The engine implements a full football game under standard rules (such as goal kicks, side kicks, corner kicks, etc.), with 11 players on each team and 3000 steps duration for a full game. It offers several state wrappers (such as Pixels, SMM, Floats) and the players can be controlled...
with 19 discrete actions (such as move in 8 directions, high pass, long pass, steal, etc.). The rewards include both the scoring reward (±1 or −1) and the checkpoint reward, where the checkpoint reward means that the agent will be reward with +0.1 if it is the first time that the agent’s team possesses the ball in each of the checkpoint regions.

E Hyperparameter Settings

E.1 Hyperparameter Settings for Real-World Games

We report our hyperparameter setting for real-world metagames in Table 3.

Table 3: The hyperparameters of real-world metagames.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>0.5</td>
<td>Learning rate for agents</td>
</tr>
<tr>
<td>Improvement threshold</td>
<td>0.03</td>
<td>Convergence criteria</td>
</tr>
<tr>
<td>Metasolver</td>
<td>Fictitious play</td>
<td>Method to compute NE</td>
</tr>
<tr>
<td>Metasolver iterations</td>
<td>1000</td>
<td># of iterations of metasolver</td>
</tr>
<tr>
<td>Metasolver iterations for PE</td>
<td>2000</td>
<td># of iterations to compute PE</td>
</tr>
<tr>
<td># of threads in pipeline</td>
<td>1.0</td>
<td>Number of learners in Pipeline-PSRO</td>
</tr>
<tr>
<td># of seeds</td>
<td>5</td>
<td># of trials</td>
</tr>
<tr>
<td>λ₁</td>
<td>0.2</td>
<td>Weight for BD</td>
</tr>
<tr>
<td>λ₂</td>
<td>0.2</td>
<td>Weight for RD</td>
</tr>
</tbody>
</table>

E.2 Hyperparameter Settings for Non-Transitive Mixture Model

We report our hyperparameter setting for non-transitive mixture model in Table 4.

Table 4: The hyperparameters of non-transitive mixture model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>0.1</td>
<td>Learning rate for agents</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
<td>Gradient-based optimization</td>
</tr>
<tr>
<td>Betas</td>
<td>(0.9, 0.99)</td>
<td>Parameter for Adam</td>
</tr>
<tr>
<td>N_iters</td>
<td>5</td>
<td># of iterations using GD per BR</td>
</tr>
<tr>
<td>Σ</td>
<td>1/2I</td>
<td>Covariance of each Gaussian</td>
</tr>
<tr>
<td>r</td>
<td>4</td>
<td># of trials</td>
</tr>
<tr>
<td>e</td>
<td>5</td>
<td>Radius of each Gaussian</td>
</tr>
<tr>
<td>Metasolver</td>
<td>Fictitious play</td>
<td>Method to compute NE</td>
</tr>
<tr>
<td>Metasolver iterations</td>
<td>1000</td>
<td># of iterations of metasolver</td>
</tr>
<tr>
<td># of threads in pipeline</td>
<td>1.0</td>
<td>Number of learners in Pipeline-PSRO</td>
</tr>
<tr>
<td># of iteration</td>
<td>50</td>
<td># of training iterations for PSRO</td>
</tr>
<tr>
<td># of seeds</td>
<td>5</td>
<td># of trials</td>
</tr>
<tr>
<td>λ₁</td>
<td>1</td>
<td>Weight for BD</td>
</tr>
<tr>
<td>λ₂</td>
<td>1500</td>
<td>Weight for RD</td>
</tr>
<tr>
<td>Decrease rate of λ₁ and λ₂</td>
<td>1 − e^{−0.25t}</td>
<td>The weights will decrease as the iteration progresses, where t is the current iteration</td>
</tr>
<tr>
<td>λ₁ decrease rate of PE</td>
<td>30</td>
<td># of training iterations using PSRO for PE</td>
</tr>
</tbody>
</table>

E.3 Hyperparameter Settings for Google Research Football

States and Network Architecture. For GRF, We use a structured multi-head vector as the states input. The information of each head is listed in Table 5:

The network structure is shown in Figure 5. The shapes of the fully-connected layers for the input heads are: [32, 64, 64, 16, 16, 128 × 64, 128 × 64, 128 × 128, 128 × 128, 128 × 128, 64, 16, 64, 64], followed by three fully-connected layers (i.e. [512 × 256 × 128]) and finally output the policy and value.

Hyperparameter Settings for Reinforcement Learning Oracle. We use IMPALA [1] as the reinforcement learning algorithm to approximate the best response for each opponent selected by different methods during the training process. The hyperparameters are listed in Table 6.
Table 5: The states input for Google Research Football.

<table>
<thead>
<tr>
<th>Head index</th>
<th>Length</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9</td>
<td>Ball information (position, direction, rotation)</td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>Ball owner information (ball owned team id, ball owned player id)</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>Active player information (id, position, direction, area of the field)</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>Active player vs. ball (distance, 1/distance)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Active player vs. ball player (distance, 1/distance)</td>
</tr>
<tr>
<td>5</td>
<td>66</td>
<td>Active player vs. self-team players (position, distance, 1/distance, position cosine, direction cosine)</td>
</tr>
<tr>
<td>6</td>
<td>66</td>
<td>Active player vs. oppo-team players (position, distance, 1/distance, position cosine, direction cosine)</td>
</tr>
<tr>
<td>7</td>
<td>88</td>
<td>Self-team information (position, direction, tired factor, yellow card, active player, offside flag)</td>
</tr>
<tr>
<td>8</td>
<td>88</td>
<td>Oppo-team information (position, direction, tired factor, yellow card, active player, offside flag)</td>
</tr>
<tr>
<td>9</td>
<td>32</td>
<td>Goal keeper information (distance to self-player/oppo-player, nearest/farthest player information)</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>Game mode information (one-hot)</td>
</tr>
<tr>
<td>11</td>
<td>29</td>
<td>Legal action and sticky action information</td>
</tr>
<tr>
<td>12</td>
<td>76</td>
<td>History (one-hot) actions of last four steps</td>
</tr>
</tbody>
</table>

Figure 5: The shape of input states for each head and the general network structure.

Network Training Details. We carry out the experiments on six servers (CPU: AMD EPYC 7542 128-Core Processor, RAM: 500G), with each one corresponding to one of six methods (i.e. Self-play, PSRO, PSROrN, PSRO w. BD, PSRO w. RD, PSRO w.BD&RD). For each experiment, the approximated best response (i.e. checkpoint) is saved only when the win-rate against corresponding opponent is stable during two checks (check frequency = 1000 model steps, and \( \Delta_{\text{winrate}} < 0.05 \)) or the training model step reaches an upper bound (i.e. 50000 model steps). The \( \lambda_1 \) and \( \lambda_2 \) we used for both coefficients are 0.5. For the Google Research Football environment settings, we use both scoring reward and checkpoint reward for the training.

F Ablation Studies

We also conduct ablation study on the sensitivity of the diversity weights \( \lambda_1 \) and \( \lambda_2 \) in real-world games, non-transitive mixture model, and Google Research Football.

F.1 Real-World Games

We report the exploitability and PE by varying \( \lambda_1 \) in Figure 6a, 6b and \( \lambda_2 \) in Figure 7a, 7b. It can be found that too large weights can cause the slow convergence and too small weights prevent the algorithm from finding populations with smaller exploitability and larger PE.

F.2 Non-Transitive Mixture Model

We report the exploitability of the final population generated by our algorithm with different \( \lambda_1 \) in Table 7 and \( \lambda_2 \) in Table 8. In this game, we set both \( \lambda_1 \) and \( \lambda_2 \) to decrease following the rate \( 1 - \frac{0.7}{1+e^{-(t-25)/(-25)}} \), where \( t \) is the current iteration. We can find that in terms of exploitability, PSRO with only BD cannot help the population to achieve lower exploitability.
Table 6: The hyperparameters of the IMPALA algorithm.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>1024</td>
</tr>
<tr>
<td>Discount Factor ($\gamma$)</td>
<td>0.993</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.00019896</td>
</tr>
<tr>
<td>Number of Actors</td>
<td>100</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Unroll Length/n-step</td>
<td>1.0</td>
</tr>
<tr>
<td>Entropy Coefficient</td>
<td>0.0001</td>
</tr>
<tr>
<td>Value Function Coefficient</td>
<td>1.0</td>
</tr>
<tr>
<td>Grad Clip Norm</td>
<td>0.5</td>
</tr>
<tr>
<td>Rho (for V-Trace)</td>
<td>1.0</td>
</tr>
<tr>
<td>C (for V-Trace)</td>
<td>1.0</td>
</tr>
<tr>
<td>$\lambda_1$ (Weight for BD)</td>
<td>0.5</td>
</tr>
<tr>
<td>$\lambda_2$ (Weight for RD)</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Figure 6: Ablation study on $\lambda_1$. (a): Exploitability vs. training iterations. (b): Negative Population Effectivity vs. training iterations on the AlphaStar game. $\text{cdf}(k) = \int_{-\infty}^{k} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$ is the cumulative distribution function of the standard normal distribution.

Figure 7: Ablation study on $\lambda_2$. (a): Exploitability vs. training iterations. (b): Negative Population Effectivity vs. training iterations on the AlphaStar game. $\text{cdf}(k) = \int_{-\infty}^{k} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$ is the cumulative distribution function of the standard normal distribution.
We also conduct an ablation study on the weight of RD λ₂ where we fixed λ₁. Table 7 shows the exploitability for populations generated by PSRO only with BD with varied diversity weight λ₁.

<table>
<thead>
<tr>
<th>λ₁</th>
<th>7.5</th>
<th>15</th>
<th>75</th>
<th>750</th>
<th>1500</th>
<th>7500</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expl</td>
<td>14.57 ± 0.69</td>
<td>14.93 ± 1.87</td>
<td>14.64 ± 1.48</td>
<td>42.37 ± 10.12</td>
<td>33.39 ± 5.71</td>
<td>62.69 ± 10.90</td>
</tr>
</tbody>
</table>

Table 8: Exploitability ×10^2 for populations generated by PSRO only with RD with varied diversity weight λ₂.

<table>
<thead>
<tr>
<th>λ₂</th>
<th>0.5</th>
<th>1.0</th>
<th>5.0</th>
<th>10.0</th>
<th>50.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expl</td>
<td>16.23 ± 0.48</td>
<td>14.06 ± 1.20</td>
<td>14.77 ± 0.09</td>
<td>15.60 ± 1.11</td>
<td>31.29 ± 12.93</td>
</tr>
</tbody>
</table>

Table 9: The win-rate between the final policies of different methods after trained for 300000 model steps. (We set λ₁ = λ₂ = 0.5 as default values for PSRO w. RD and PSRO w. BD&RD)

<table>
<thead>
<tr>
<th>Method</th>
<th>Self-play</th>
<th>PSRO</th>
<th>PSRO₂</th>
<th>PSRO w. BD</th>
<th>PSRO w. RD</th>
<th>PSRO w. BD&amp;RD</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSRO w. RD (λ₂ = 1.0)</td>
<td>0.62 ± 0.01</td>
<td>0.49 ± 0.03</td>
<td>0.65 ± 0.02</td>
<td>0.47 ± 0.01</td>
<td>0.28 ± 0.04</td>
<td>0.33 ± 0.02</td>
</tr>
<tr>
<td>PSRO w. RD (λ₂ = 0.5)</td>
<td>0.68 ± 0.03</td>
<td>0.61 ± 0.02</td>
<td>0.74 ± 0.03</td>
<td>0.54 ± 0.02</td>
<td>-</td>
<td>0.43 ± 0.02</td>
</tr>
<tr>
<td>PSRO w. RD (λ₂ = 0.2)</td>
<td>0.63 ± 0.02</td>
<td>0.48 ± 0.02</td>
<td>0.68 ± 0.02</td>
<td>0.50 ± 0.01</td>
<td>0.45 ± 0.03</td>
<td>0.28 ± 0.03</td>
</tr>
<tr>
<td>PSRO w. BD&amp;RD</td>
<td>0.74 ± 0.02</td>
<td>0.78 ± 0.01</td>
<td>0.80 ± 0.05</td>
<td>0.69 ± 0.02</td>
<td>0.57 ± 0.02</td>
<td>-</td>
</tr>
</tbody>
</table>

Algorithm 3 Optimization for Matrix Games

1: **Input:** population 𝜋i for each i, meta-game A_{q1,xq-1}, weights λ₁ and λ₂, learning rate μ
2: σ₁, σ₂ ← Nash on A_{q1,xq-1}
3: πE ← Aggregate according to σ₁, σ₂
4: π_i(θ) ← Initialize a new random policy for player i
5: BR_{qual} ← Compute best response against mixture of opponents \( \sum_k σ^{-1}_i φ_i(\cdot, π^k_i) \)
6: while the reward \( p \) improvement does not meet the threshold do
7: BR_{occ} ← arg max_s \( D_f(s_j || π_i) \) for each pure strategy \( s_j \)
8: BR ← Choose BR = BR_{occ} with probability \( λ₁ \) else BR = BR_{qual}
9: \( \theta \leftarrow \mu θ + (1 - \mu)θ_{BR} \)
10: \( p ← Compute \ the \ payoff \ \text{after the update according to} \ \sum_k σ^{-1}_i φ_i(π_i(θ), π_{-i}) \)
11: end while
12: BR_{rew} ← arg max_s \( F(s_j) \) for each pure strategy \( s_j \) with probability \( λ₂ \) else BR_{qual}
13: \( \theta ← \mu θ + (1 - \mu)θ_{BR_{rew}} \)
14: **Output:** policy \( π_i(θ) \)

F.3 Google Research Football

We also conduct an ablation study on the weight of RD λ₂ (see Table 9) in the GRF environment, where we fixed λ₁ to be 0.5 and show the results with different λ₂.

G Simplified Optimization Method for Unified Diverse Response

In Algorithm 1, we have outlined using RL as the optimization oracle for approximate best response. However, computing best response in real-world metagames (matrix games) or non-transitive mixture model (differential games) can be simplified, since the f-divergence objective can be simplified according to Theorem 1 or the reward function \( φ_i \) is analytically accessible. Now we provide the simplified optimization methods separately for matrix games in Algorithm 3 and differential games in Algorithm 4.
Algorithm 4 Optimization for Differential Games

1: **Input:** population $\mathcal{P}_i$ for each $i$, meta-game $A_{\mathcal{P}_i \times \mathcal{P}_{-i}}$, weights $\lambda_1$ and $\lambda_2$, number of gradient updates $N_{\text{train}}$

2: $\sigma_i, \sigma_{-i} \leftarrow \text{Nash on } A_{\mathcal{P}_i \times \mathcal{P}_{-i}}$

3: $\pi_E = (\pi_i, \pi_{E_{-i}}) \leftarrow \text{Aggregate according to } \sigma_i, \sigma_{-i}$

4: $\pi'(\theta) \leftarrow \text{Initialize a new random policy for player } i$

5: **for** $j = 1$ **to** $N_{\text{train}}$ **do**

6: $p_j \leftarrow \text{Compute payoff against the mixture of opponents } p_j = \sum_k \sigma_{-i}^k \phi_i(\pi'_i, \pi'_k)$

7: $d_{\text{occ}}^j \leftarrow \text{Compute BD } d_{\text{occ}}^j = D_f(\pi'_i || \pi_i) \text{ as the } f\text{-divergence between } \pi_i \text{ and } \pi'_i$

8: $a_j \leftarrow \text{Compute new reward vector as } a_j = (\phi_i(\pi'_i, \pi'_k))_{k=1}^{\mathcal{P}_{-i}}$

9: $d_{\text{rew}}^j \leftarrow \text{Compute the lower bound of RD as } F(a_j) \text{ according to Theorem 2}$

10: $l_j \leftarrow -(p_j + \lambda_1 d_{\text{occ}}^j + \lambda_2 d_{\text{rew}}^j)$

11: **Update** $\theta$ to minimize $l_j$ by backpropagation

12: **end for**

13: **Output:** policy $\pi'_i(\theta)$

References


