

# Policy Space Response Oracle (PSRO) with Diversity

梁晏成

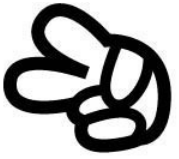


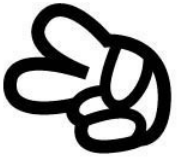


# Outline

---

- Terminology
- PSRO Framework
  - Parallel Training (DCH, P2SRO)
- PSRO from a game-theoretical view
  - N-player General Game: CE
  - Extensive Form Game: XDO
- PSRO with Diversity
  - DPP
  - Behavioral and Response Diversity

# Game Theory Terminology

- Normal-Form Game (Matrix Game)

			
			
		$(-1, 1)$	
			

Pure strategy for player i:  $a_i \in A_i$

for opponent:  $a_{-i} \in A_{-i}$

The whole action space  $A = A_1 \times \cdots \times A_n$

Mixed strategy:  $\pi_i = \sum_{a_i \in A_i} p(a_i) a_i$

$$\pi_i \in \Sigma_i = \Delta(A_i)$$

Independent-player:  $\Sigma = \Delta(A_1) \times \cdots \times \Delta(A_n)$

Correlated-player:  $\Sigma = \Delta(A_1 \times \cdots \times A_n)$

Payoff  $u_i(\pi_i, \pi_{-i}) = u_i(\pi)$

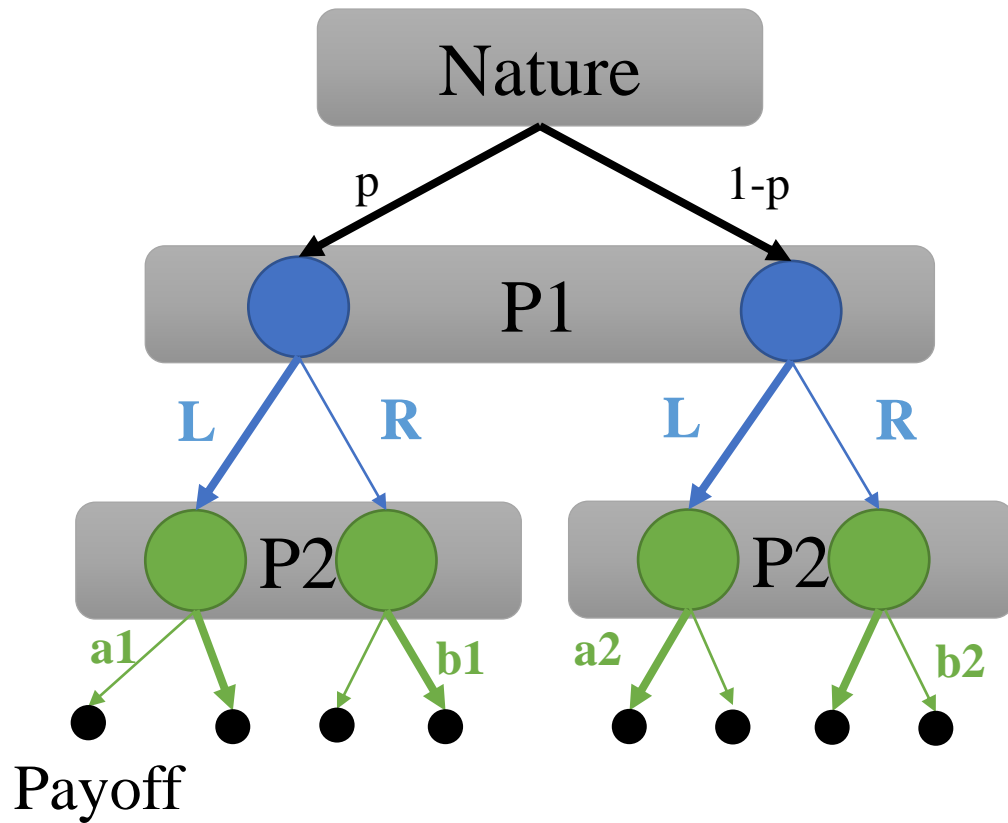
Best response :  $BR_i(\pi) = \arg \max_{\pi'_i} u_i(\pi'_i, \pi_{-i})$

NE:  $\pi_i \in BR_i(\pi)$

Exploitability:  $\sum_{i=1}^n u_i(BR_i(\pi), \pi_{-i}) - u_i(\pi)$

# Game Theory Terminology

- Extensive-Form Game



Node (history)	$s$	State
Action Space	$A(s)$	Action
Information Set	$I(s)$	Observation
(note: $A(I(s)) = A(s)$ , so $I(s) = I(s') \Rightarrow A(s) = A(s')$ )		

Player at some node  $P(s) = P(I(s))$

Pure strategy space  $A_i = \prod_{P(I)=i} A(I)$

Mixed strategy space  $\Delta(A_i)$

Behavior strategy space  $\prod_{P(I)=i} \Delta(A(I))$

# PSRO Framework

- A population of policies  $\Pi^T = \{\pi^1, \dots, \pi^T\}$
- An empirical game (restricted game) defined by (maybe implicit) a payoff tensor  $M^{\Pi^T}$
- Compute a meta strategy  $\sigma(\sigma_i = \sum_{j=1}^T \sigma_i^j \pi_i^j)$  on  $\Pi^T$  by a meta-solver
- Compute oracle  $\pi_i^{T+1} = BR_i(\sigma)$
- Now  $\Pi^{T+1} = \Pi^T \cup \{\pi^{T+1}\}$
- Example:
  - Self-play:  $\sigma_i = (0, \dots, 0, 1)$
  - Iterated BR:  $\sigma_i = (0, \dots, 0, 1, 0)$
  - Fictitious self-play (FSP):  $\sigma_i = \frac{1}{T} (1, \dots, 1, 0)$
  - Double Oracle (DO):  $\sigma = NE(M^{\Pi^T})$

Almost all use FSP to compute a *NE*

Almost all boost diversity by  
$$\pi_i^{T+1} = \arg \max_{\pi_i} u_i(\pi_i, \sigma_{-i}) + \lambda Div$$

# About Double Oracle

---

- Double Oracle for two-player game
  - Tabular (finite action space) Case:
    - Oracle policy  $\pi_i^T$  becomes pure strategy  $s_i^T$
    - Converge to  $\epsilon - NE$  if we use  $\epsilon - BR$  and  $\epsilon - NE$  meta solver (like LP)
      - Note: LP is only available for two-player case
  - General Case:
    - Usually use FSP to compute  $\epsilon - NE$  for the empirical game, theoretically converges only when FSP converges (two-player zero-sum, potential game)

*(Auxiliary)*

## **Double Oracle Framework:**

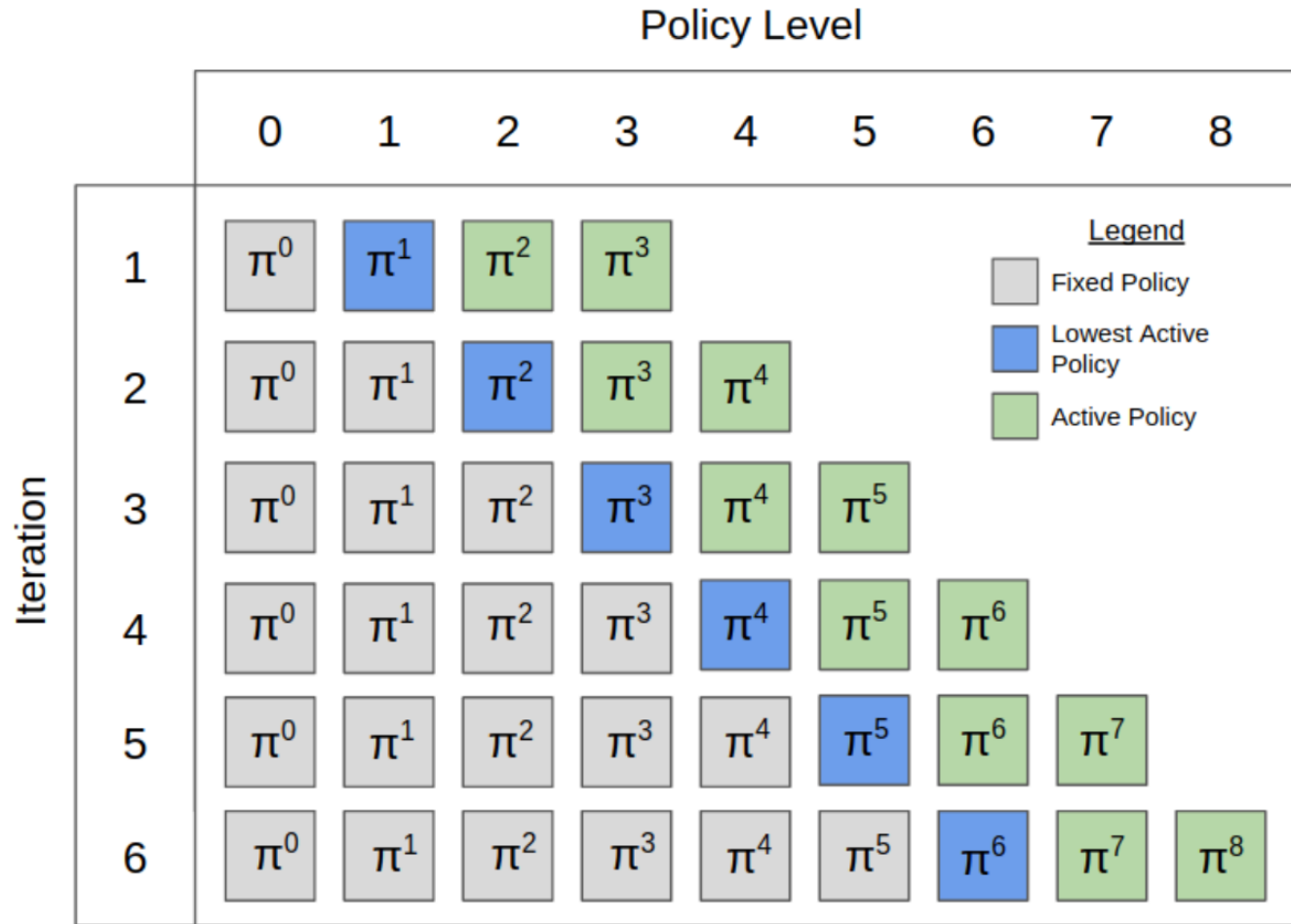
- 1) Maintain  $\Pi = (\pi^1, \dots, \pi^T)$
- 2) Compute  $\sigma = NE(M^\Pi)$  for the empirical game
- 3) Get oracle  $\pi^{T+1} = BR(\sigma)$  and  $\Pi \leftarrow \Pi + \pi^{T+1}$

# Scalable PSRO-based Algorithm

---

- Deep Cognitive Hierarchies [1]
  - Fully Parallel
  - Use an evolutionary method for the meta-solver
  - *Omitted*
- Pipeline PSRO [2]

# Pipeline Policy Space Response Oracles



- Training framework
- Maintain Two classes of policies:  $\Pi = \Pi^f \cup \Pi^a$ 
  - Idea: pretrain RL agents for the tedious learning at the beginning
- At every episode, if  $\pi^j \in \Pi^a$  :
  - $\pi_i^j$  is trained against  $\hat{\pi}_{-i}^j \sim \sigma_{-i}^j$  using RL algorithm (SAC here)
  - If  $j$  is the lowest active policy and  $\pi^j$  plateaus (performance does not improve past a given threshold in a given amount of time), fix  $j$  and activate new  $\pi^{t+1}$ ,
  - Periodically refresh the meta strategy  $\sigma^{j+1}$  to be a *NE* of  $\Pi^j$ , computed by fictitious play



# Pipeline Policy Space Response Oracles

---

**Algorithm 1** Pipeline Policy-Space Response Oracles

---

**Input:** Initial policy sets for all players  $\Pi^f$

Compute expected utilities for empirical payoff matrix  $U^\Pi$  for each joint  $\pi \in \Pi$

Compute meta-Nash equilibrium  $\sigma^{*,j}$  over fixed policies ( $\Pi^f$ )

**for** many episodes **do**

**for all**  $\pi^j \in \Pi^a$  in parallel **do**

**for** player  $i \in \{1, 2\}$  **do**

            Sample  $\pi_{-i} \sim \sigma_{-i}^{*,j}$

            Train  $\pi_i^j$  against  $\pi_{-i}$

**end for**

**if**  $\pi^j$  plateaus and  $\pi^j$  is the lowest active policy **then**

$\Pi^f = \Pi^f \cup \{\pi^j\}$

        Initialize new active policy at a higher level than all existing active policies

        Compute missing entries in  $U^\Pi$  from  $\Pi$

        Compute meta Nash equilibrium for each active policy

**end if**

    Periodically compute meta Nash equilibrium for each active policy

**end for**

**end for**

Output current meta Nash equilibrium on whole population  $\sigma^*$

---

- Implementation details of computing *NE*: (*I check it from the code, while it is not mentioned in the paper*)
- They explicitly compute the empirical game payoff matrix  $M^\Pi$  by simulated several ( $10^2 \sim 10^3$ ) games for two strategies
- Then they do 2000 iterations of FSP based on the normal-form game defined by the payoff matrix

# Pipeline Policy Space Response Oracles

- Evaluation:

Note:

**DCH:** fully parallel

**Naïve PSRO:** all parallel works play against the same  $\sigma$  of the lowest active policy

**Rectified PSRO:**  $BR$  against a combination of policies beat by a support of  $\sigma$

**Sequential PSRO:** no parallel

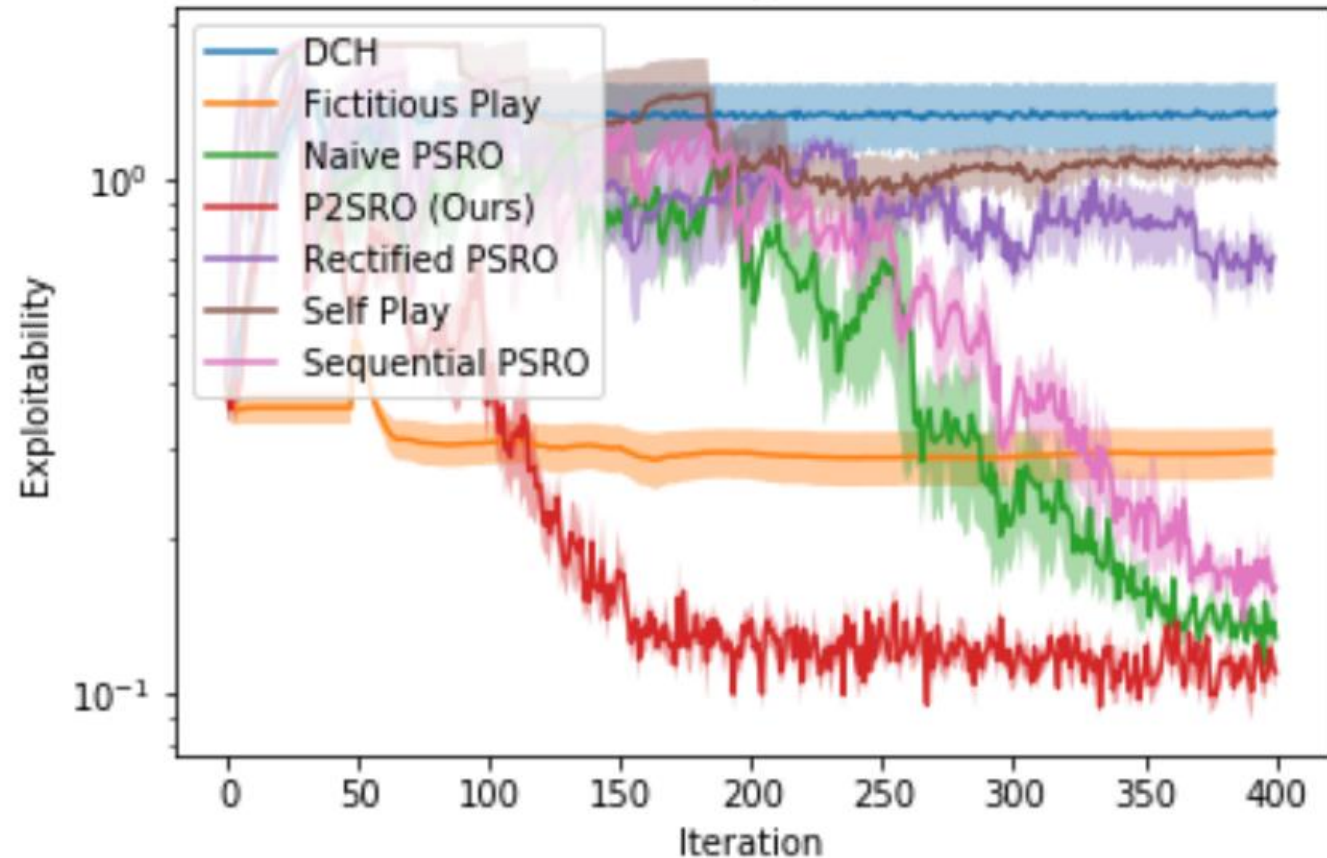
**Insight:** Naïve PSRO  $\approx$  Sequential PSRO

## 1) Random Symmetric Normal-Form Game

Tabular case,  $BR$  can be computed directly.

Every time the policy moves to the  $BR$  with a learning rate

Dimension: 60, Learning Rate: 0.1, Workers: 4

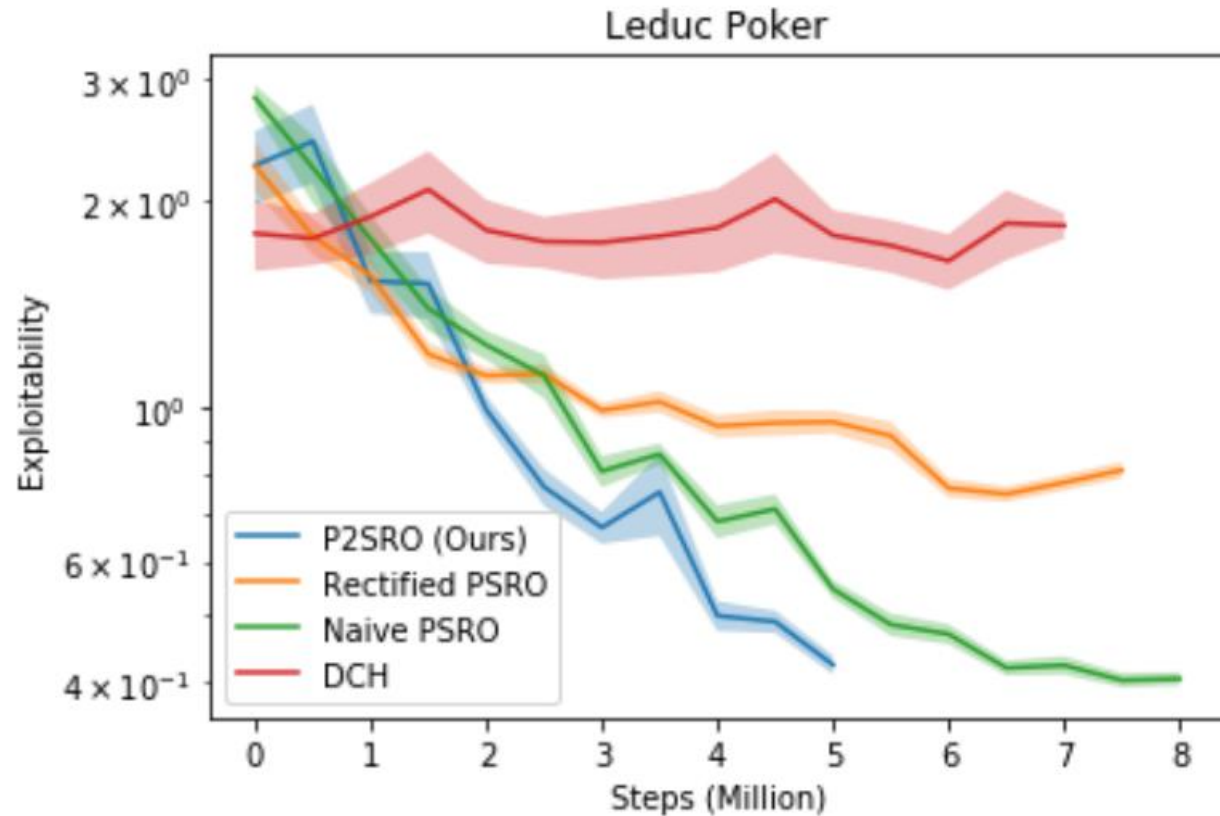


# Pipeline Policy Space Response Oracles

- Evaluation:

## 2) Leduc Poker

Why not compare to CFR?  
(neither does DCH)



(a) Leduc poker

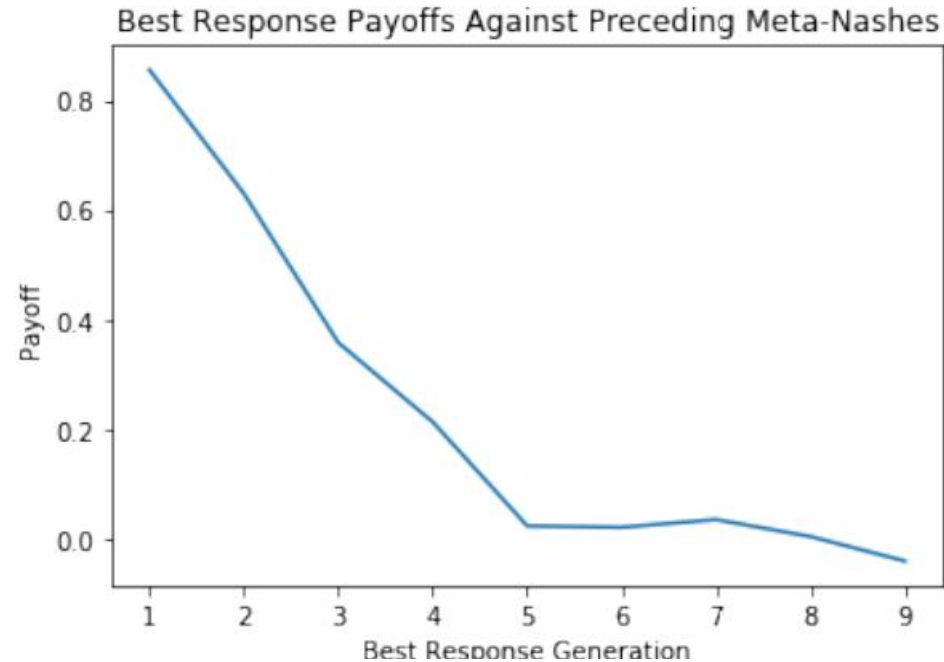
# Pipeline Policy Space Response Oracles

- Evaluation:

Barrage Stratego is a game like Kriegspiel (军旗)

							F		
				9		2	10		
2	3				B			S	
	S				2		3		
				2				10	
							9		
							B	F	

## 3) Barrage Stratego



Name	P2SRO Win Rate vs. Bot
Asmodeus	81%
Celsius	70%
Vixen	69%
Celsius1.1	65%
<b>All Bots Average</b>	<b>71%</b>

After 82000 episodes

# Theoretically Extended PSRO (esp. DO)

- PSRO with Correlated Equilibrium Meta-Solvers [3]
  - **Only for the tabular case**
  - Extend PSRO to n-player general-sum game **by replace *NE* meta-solver with *CE* meta-solver**
  - See some challenges for games that are not two-player zero-sum
- Extensive Double Oracle [4]
  - Mix the policy space **at each information** set instead of at the root
  - For the tabular case, it suffices to compute oracle for linear times with respect to **the number of information set (linear iteration numbers)**

# Multi-Agent Training beyond Zero-Sum with Correlated Equilibrium Meta-Solvers

- Note: here only consider the **tabular case, normal-form** game
- The policy is now a joint p.d. on the whole action space  $A$ ; let  $\sigma(a_i, a_{-i})$  be the probability of one certain joint action
- Suppose a mechanism samples according to  $\sigma$  and send each one  $a_i$
- Definition of Correlated Equilibrium  $CE$ :
  - $\forall i, a_i, a_i \in \arg \max_{a'_i} \sum_{a_{-i} \in A_{-i}} \sigma(a_{-i} | a_i) u_i(a'_i, a_{-i})$ 
    - Meaning that under the agreement of  $\pi$ , when an agent is told to do  $a_i$ , it will
- Definition Of coarse CE  $CCE$ :
  - $\forall i, u_i(\pi) = \max_{a'_i} \sum_{a_{-i} \in A_{-i}} \sigma(a_{-i}) u_i(a'_i, a_{-i})$ 
    - Meaning that under the agreement of  $\pi$ , the agent first decides whether follows the agreement, and then automatically listens to the mechanism

# Multi-Agent Training beyond Zero-Sum with Correlated Equilibrium Meta-Solvers

- Now we define  $\epsilon - CE$ :
  - $\forall i, a_i, a'_i \neq a_i, \sum_{a_{-i} \in A_{-i}} \sigma(\textcolor{red}{a}_{-i}, \textcolor{red}{a}_i) u_i(a'_i, a_{-i}) \leq u_i(\sigma) + \epsilon$
  - Note here the  $\neq$  is important! So we can make  $\epsilon < 0$  later
- Point 1: use  $CE$  instead of  $NE$ 
  - Justification: for  $n > 2$  or general-sum game, independent strategy does not guarantee a lower-bound utility, so why not use a joint mechanism?
  - Truth:  $CE$  has a nice property: any convex combination of  $\epsilon - CE$  is still a  $\epsilon - CE$ , which is essential for the later  $CE$  meta-solver; but this does not hold for  $NE$
- Point 2: use  $NFC(C)E$  rather than  $EFC(C)E$ 
  - Justification: we only communicate once at the root;  $NFCCE$  provides the highest welfare
  - Truth:  $EFC(C)E$  is hard to solve

# Multi-Agent Training beyond Zero-Sum with Correlated Equilibrium Meta-Solvers

- $CE$  constraints in matrix form
- Build an advantage matrix  $Y_i: [|A_i|(|A_i| - 1), |A|]$  by
  - $Y_i(a_i, a'_i, a_i, a_{-i}) = u_i(a'_i, a_{-i}) - u_i(a_i, a_{-i})$

*Align with  $\sigma(a_i, a_{-i})$*
- Now we get a matrix form restriction  $Y_i \sigma \leq \epsilon, \forall i$ 
  - This also shows the convexity of  $CE$



# Multi-Agent Training beyond Zero-Sum with Correlated Equilibrium Meta-Solvers

- $\epsilon - MG(C)CE$

- $\max 1 - \sigma^\top \sigma \text{ s.t.}$

- $\forall i, Y_i \sigma \leq \epsilon$

- $\sigma \geq 0, e^\top \sigma = 1$

- Quadratic Programming

- First use Lagrange Duality

- Then use gradient descent

Different CE:

MECE:  $\max H(\sigma)$

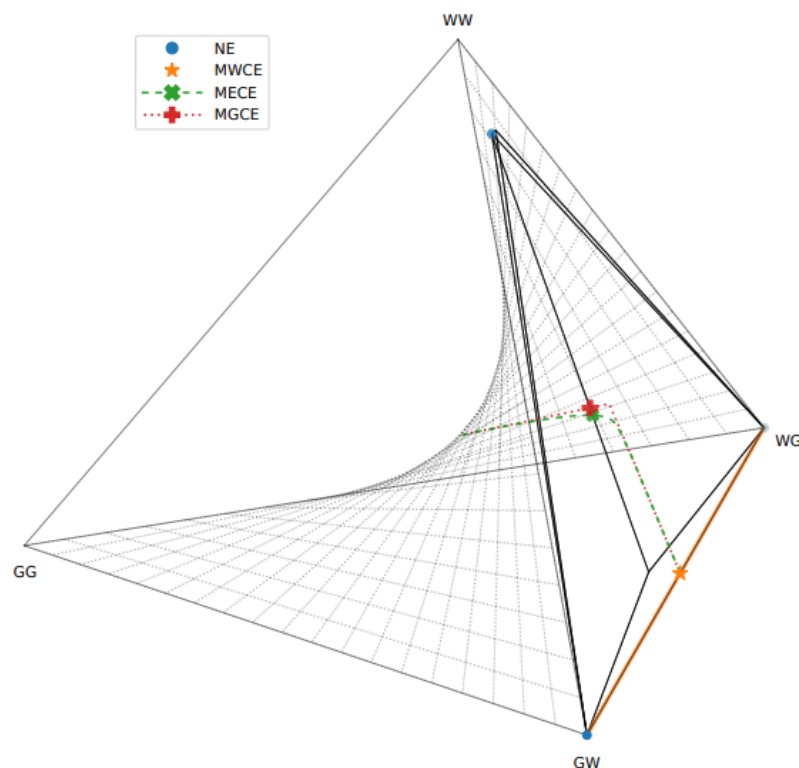
solvable when the solution  $\pi$  is full-support

MWCE:  $\max \sum_{i \in N} u_i(\sigma)$

solution is not unique

$MG(C)CE$  property: invariant under a linear transformation on payoff matrix

# • Equilibrium Selecting



(a) CE Convex Polytope

	G	W
G	-10, -10	1, 0
W	0, 1	0, 0

(b) Payoff Table

	G	W
G	0.033	0.334
W	0.334	0.299

(c) MECE

(0, 0)

factorizable

	G	W
G	$\frac{1}{121}$	$\frac{10}{121}$
W	$\frac{10}{121}$	$\frac{100}{121}$

(d) Mixed NE

(0, 0)

	G	W
G	0	1
W	0	0

(e) Row NE

(1, 0)

	G	W
G	0	0
W	1	0

(f) Col NE

(0, 1)

unique

	G	W
G	0.033	0.327
W	0.327	0.313

(g) MGCE

(0, 0)

	G	W
G	0	0.34
W	0.34	0.32

(h) F-MGCE

(0.34, 0.34)

	G	W
G	0	0.5
W	0.5	0

(i) MWCE

(0.5, 0.5)

unique  
full support

not unique

Figure 1. The solution landscape for the traffic lights game. The solid polytope shows the space of CE joint strategies, and the dotted surface shows factorizable joint strategies. NEs are where the surface and polytope intersect. There are three unsatisfying NEs: mixed spends most of its time waiting and does not avoid crashing, the others favour only the row or column player. One MWCE provides a better solution (note that Row NE and Col NE, and any mixture of the two are also MWCE solutions). The center of the tetrahedron is the uniform distribution and the MECE and MGCE attempt to be near this point. The dashed lines correspond to the family of solutions permitted by MGCE and MECE when varying the approximation parameter  $\epsilon$ . Both have  $(GW, WG) = (0.5, 0.5)$  as the  $\min \epsilon$  solution. Player payoffs are given in parenthesis.

# Multi-Agent Training beyond Zero-Sum with Correlated Equilibrium Meta-Solvers

- $\max \quad 1 - \sigma^\top \sigma \text{ s.t.}$ 
  - $\forall i, Y_i \sigma \leq \epsilon$
  - $\pi \geq 0, e^\top \sigma = 1$

$\epsilon$  – *MGCE* Family

Table 1. Family of MG(C)CE solutions.

MG(C)CE	$\epsilon$	Properties
$\max(Ab)\epsilon$ -MG(C)CE	$\max(Ab)$	Uniform, highest entropy, lowest payoff
$\frac{1}{2} \max(Ab)\epsilon$ -MG(C)CE	$\frac{1}{2} \max(Ab)$	Between uniform and (C)CE
full $\epsilon$ -MG(C)CE	$\leq \max(Ab)$	Minimum $\epsilon$ such that MG(C)CE is full-support
MG(C)CE	0	Weak (C)CE, NE in two-player constant sum
$\min \epsilon$ -MG(C)CE	$\leq 0$	Strictest (C)CE, lowest entropy, highest payoff

# Multi-Agent Training beyond Zero-Sum with Correlated Equilibrium Meta-Solvers

## Joint-PSRO

- $BR_i^{T+1}(a_i) = \arg \max_{a'_i} \sum_{a_{-i} \in \Pi_{-i}^T} \sigma(a_{-i} \mid a_i) u_i(a'_i, a_{-i})$  Oracle
- For all  $a_i \in \Pi_i^T$

- $\max \quad 1 - \sigma^\top \sigma \text{ s.t.}$ 
  - $\forall i, Y_i^{\Pi^T} \sigma \leq \epsilon$
  - $\pi \geq 0, e^\top \sigma = 1$

*MGCE* meta-solver

# Multi-Agent Training beyond Zero-Sum with Correlated Equilibrium Meta-Solvers

---

## Algorithm 1 Two-Player PSRO

---

```

1:  $\Pi_1^0, \Pi_2^0 \leftarrow \{\pi_1^0\}, \{\pi_2^0\}$ 
2:  $G^0 \leftarrow \text{ER}(\Pi^0)$ 
3:  $\sigma_1^0, \sigma_2^0 \leftarrow \text{MS}(G^0)$ 
4: for  $t \leftarrow \{1, \dots\}$  do
5:    $\pi_1^t, \Delta_1^t \leftarrow \text{BR}(\Pi_2^{t-1}, \sigma_2^{t-1})$ 
6:    $\pi_2^t, \Delta_2^t \leftarrow \text{BR}(\Pi_1^{t-1}, \sigma_1^{t-1})$ 
7:    $\Pi_1^t, \Pi_2^t \leftarrow \Pi_1^{t-1} \cup \{\pi_1^t\}, \Pi_2^{t-1} \cup \{\pi_2^t\}$ 
8:    $G^t \leftarrow \text{ER}(\Pi^t)$ 
9:    $\sigma_1^t, \sigma_2^t \leftarrow \text{MS}(G^t)$ 
10:  if  $\Delta_1^t + \Delta_2^t = 0$  then
11:    break
return  $(\Pi_1^{0:t}, \Pi_2^{0:t}), (\sigma_1^t, \sigma_2^t)$ 

```

---



---

## Algorithm 2 JPSRO

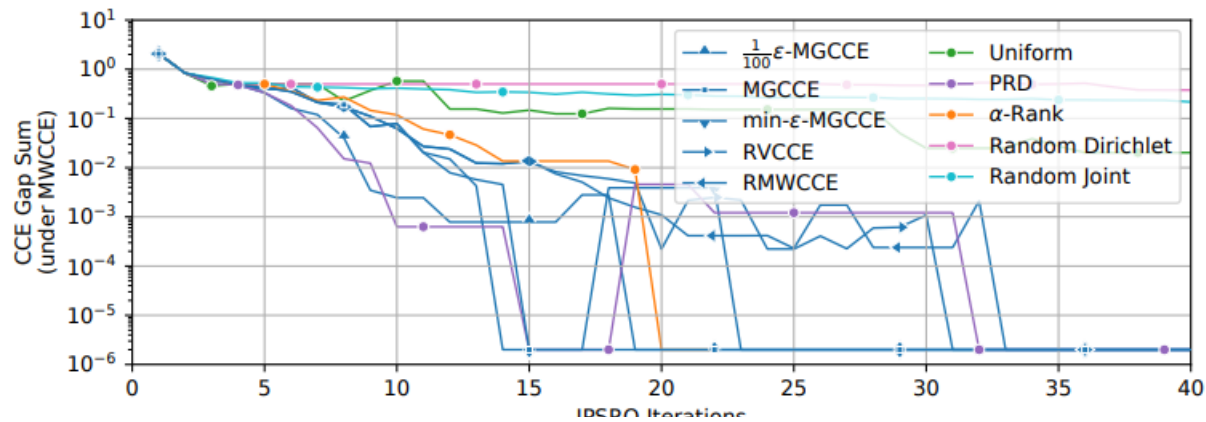
---

```

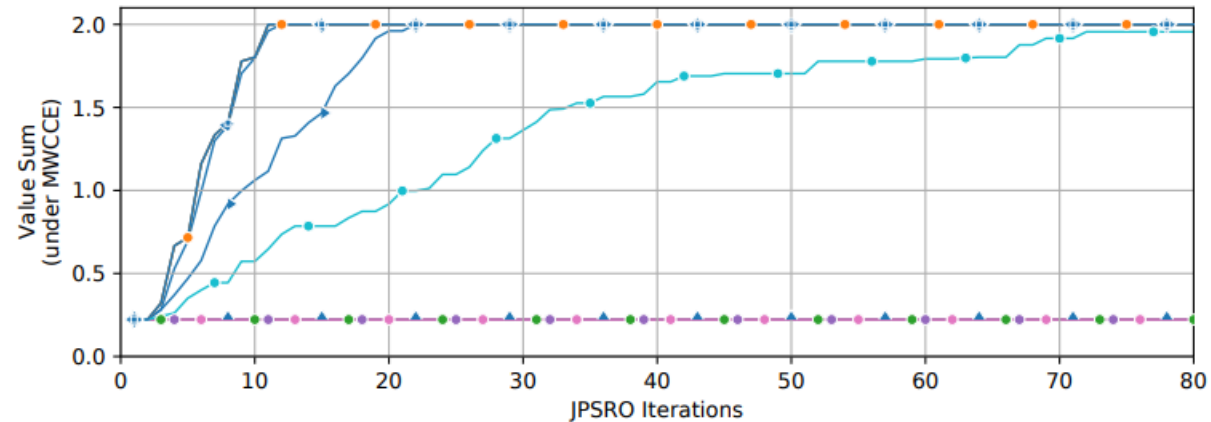
1:  $\Pi_1^0, \dots, \Pi_n^0 \leftarrow \{\pi_1^0\}, \dots, \{\pi_n^0\}$ 
2:  $G^0 \leftarrow \text{ER}(\Pi^0)$ 
3:  $\sigma^0 \leftarrow \text{MS}(G^0)$ 
4: for  $t \leftarrow \{1, \dots\}$  do
5:   for  $p \leftarrow \{1, \dots, n\}$  do
6:      $\{^1\pi_p^t, \dots\}, \{^1\Delta_p^t, \dots\} \leftarrow \text{BR}_p(\Pi^{0:t-1}, \sigma^{t-1})$ 
7:      $\Pi_p^{0:t} \leftarrow \Pi_p^{0:t-1} \cup \{^1\pi_p^t, \dots\}$ 
8:    $G^{0:t} \leftarrow \text{ER}(\Pi^{0:t})$ 
9:    $\sigma^t \leftarrow \text{MS}(G^{0:t})$ 
10:  if  $\sum_{p,c} {}^c\Delta_p^t = 0$  then
11:    break
return  $\Pi^{0:t}, \sigma^t$ 

```

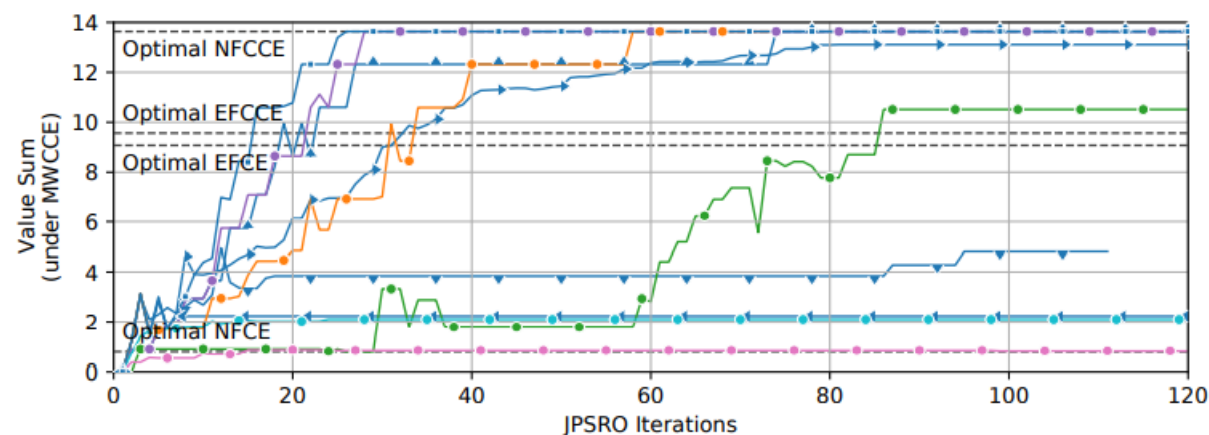
---



(a) CCE Gap on three-player Kuhn Poker. Several MS converge to within numerical accuracy (data is clipped) of a CCE.



(b) Value sum on three-item Trade Comm. The approximate CCE MS was not sufficient to converge in this game, however all valid CCE MSs were able to converge to the optimal value sum.



(c) Value sum on Sheriff. The optimal maximum welfare of other solution concepts are included to highlight the appeal of using NFCCE.

The final policy is a *MWCCE* on the whole policy space

- 1) *CCE* meta-solver converges quickly and successful find good solutions in all three games
- 2)  $\alpha$ -rank looks good

In other experiment results: *CCE* finds more unique polices

# XDO: A Double Oracle Algorithm for Extensive-Form Games

In one sentence:

instead of add policy at the root, add new action at every information set

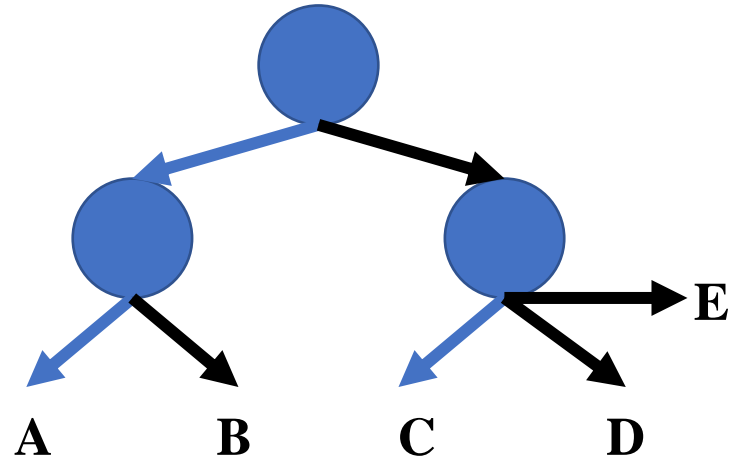
- First tabular case. Note for tabular case pure strategy oracle suffices.

# XDO: A Double Oracle Algorithm for Extensive-Form Games

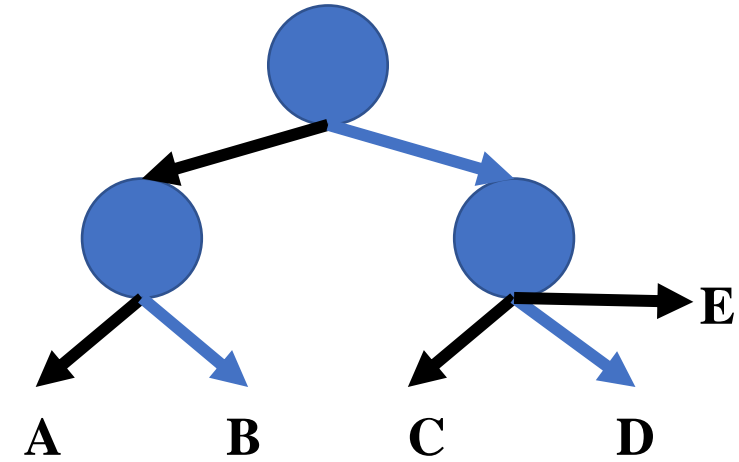
- In DO, the  $NE$  of an empirical game is computed.
- In XDO, instead of using the empirical game, another restricted game is constructed: the restricted game inherit all nodes of the original game, but at each information set, only those actions suggested by some strategy  $s_i^t$  is included.



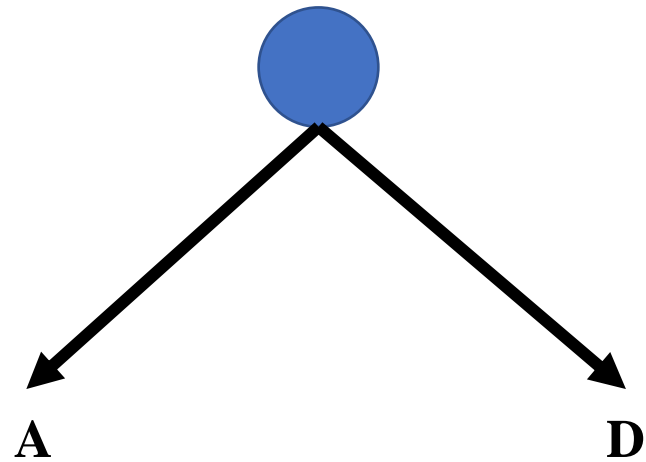
Here for clarity, assume there is only one player



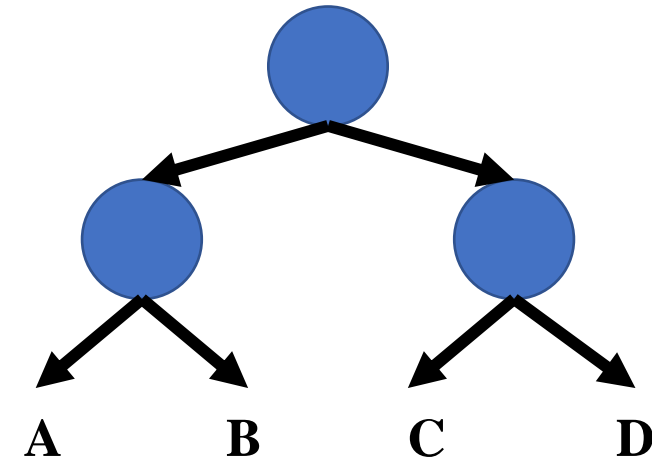
A population of two



DO case: empirical game



XDO restricted game is much larger



# XDO: A Double Oracle Algorithm for Extensive-Form Games

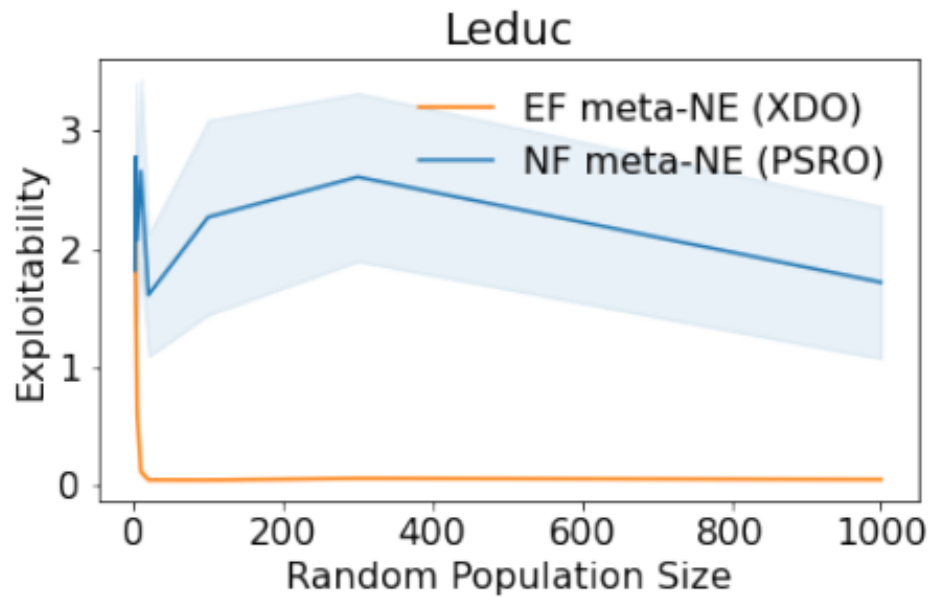
- In DO, the  $NE$  of an empirical game is computed, and we may explicitly compute the payoff matrix  $M^\Pi$
- In XDO, we re-define the action space of each information set, and then we can use CFR to compute an approximate  $NE$  on the restricted game
  - When the  $NE$  support of the original game is large, direct CFR may be better
- DO terminates in  $\sum_{i \in N} \prod_{P(I)=i} A(I)$  iterations
- XDO terminates in  $|\{I\}|$  iterations

# XDO: A Double Oracle Algorithm for Extensive-Form Games

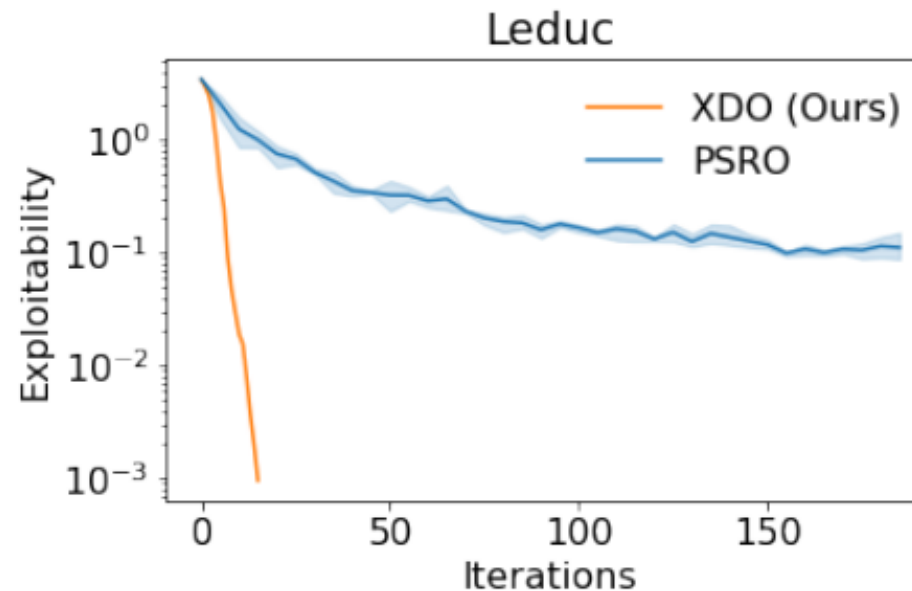
- Now general case: Neural-XDO
- Construct a restricted game as follows: at iteration  $T$ , on each information set  $I$ , choose a policy from  $\Pi^T$  as an action (i.e., choose some  $1 \leq k \leq T$ ), and then act according to  $\pi^k(I)$
- Use NFSP to compute an  $\epsilon - NE$  for the restricted game
  - The action space grows linear with the number of iterations, but NXDO converges in only a few iterations
  - NXDO-VA is the variant that still computes the explicit valid action space

# XDO: A Double Oracle Algorithm for Extensive-Form Games

- Evaluation 1) Compared to PSRO



Much lower exploitability than PSRO  
given the same policy population

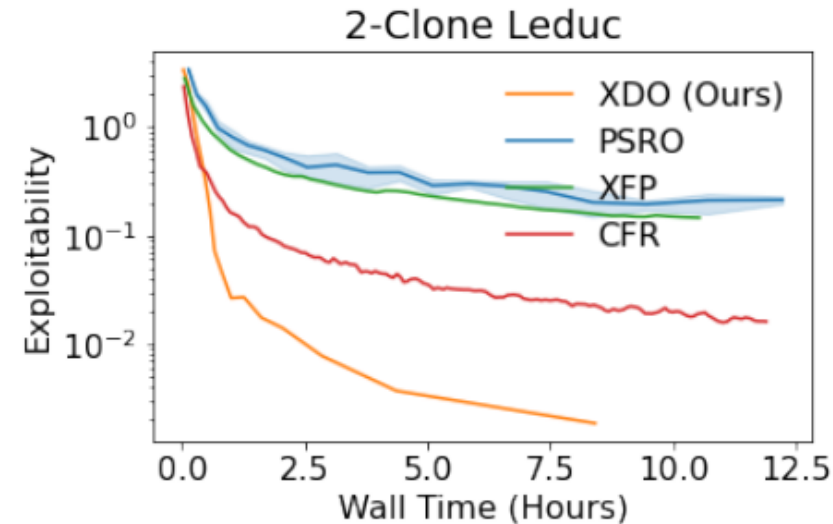
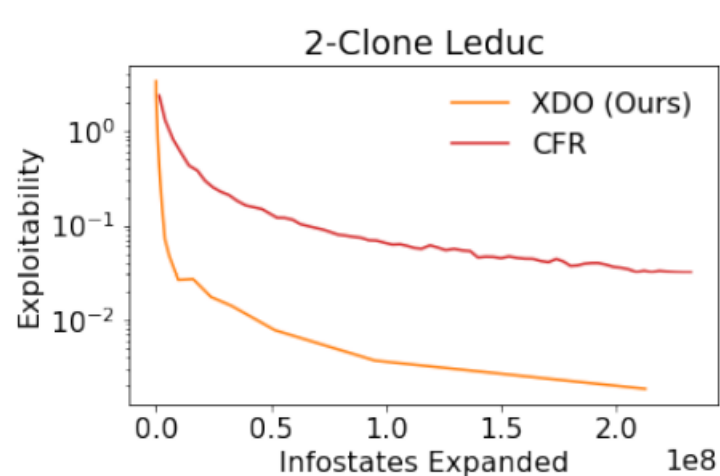


Leduc poker

Much lower exploitability than PSRO  
after the same number of iterations

# XDO: A Double Oracle Algorithm for Extensive-Form Games

- Evaluation 2) Designed m-Leduc Poker: every action is copied for m times.

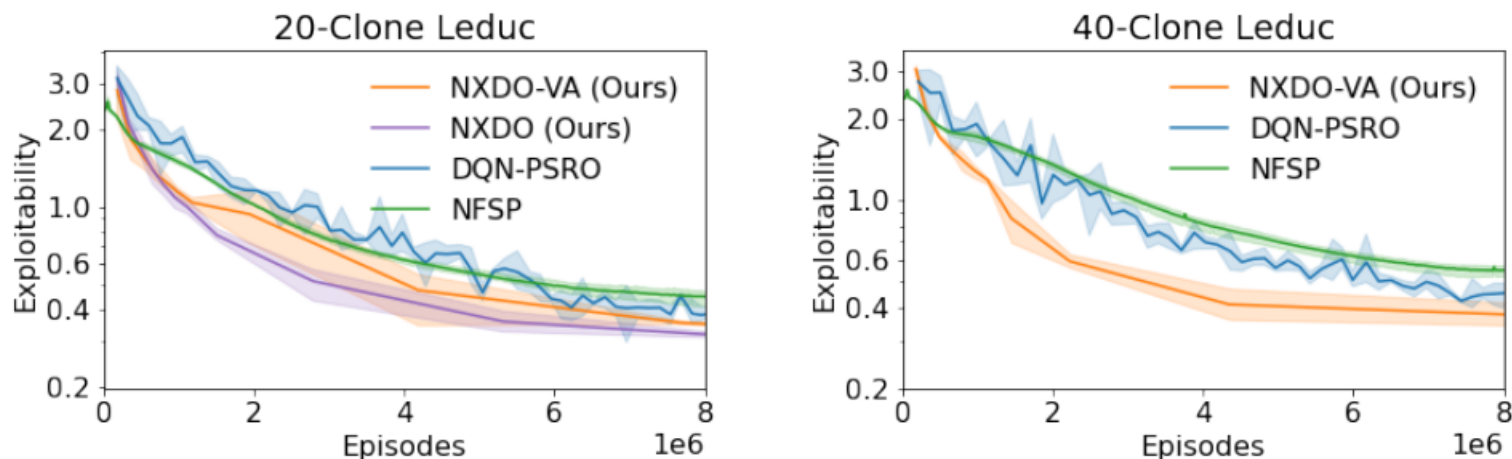


2-Leduc poker

Much quicker convergence rate than CFR, but a little bit tricky here, since we assign a prior on the choice of the best response, say, always choose the first best pure strategy response

# XDO: A Double Oracle Algorithm for Extensive-Form Games

- Evaluation 3) Even Larger Case



NFSP seems to be the bottle neck

	Average Reward			Average Win Rate		
	NXDO	PSRO	NFSP	NXDO	PSRO	NFSP
NXDO	-	<b>0.62</b>	-0.18	-	<b>0.51</b>	<b>0.60</b>
PSRO	-0.62	-	-0.46	0.49	-	0.58
NFSP	<b>0.18</b>	0.46	-	0.40	0.42	-

Table 1. No Limit Poker

# PSRO with Diversity Measurement

- PSRO with DPP diversity [5]
  - Define the population diversity as the expected cardinality of a G-DPP
- PSRO with behavior and response diversity [6]
  - Mix the policy space at each information set instead of at the root
  - For the tabular case, only need to call oracle in linear times with respect to the number of information set

# Modelling Behavioural Diversity for Learning in Open-Ended Games

- Previous Diversity Measures
- $BR_i(\pi) = \arg \max_{\pi'_i} u_i(\pi'_i, \pi_{-i}) + \tau \mathcal{H}(\pi'_i)$ 
  - Use **no population information**
  - Some PBT training loss includes population divergence, but will not be covered here
- Effective diversity:  $ED(\Pi) = \sigma^\top [M]_+ \sigma$  (rectified- $NE$ )
  - Every time add a  $BR$  against every policy  $\pi_i^j$ 's defeated opponents
  - Counter examples that it **does not converge**



# Modelling Behavioural Diversity for Learning in Open-Ended Games

- *Definition. DPP Process*
- For a kernel (positive semi-definite)  $L \in \mathbb{R}^{T \times T}$
- We sample a set  $Y \subseteq \{1,2, \dots, m\}$  , with probability proportional to  $\det(L_Y)$
- G-DPP. Let the kernel be  $L = M^\top M$  where  $M$  is the payoff matrix of the empirical game
  - The  $j$ -th row of  $M$  should be  $\pi_i^j$ 's payoff against all other  $\pi_{-i}^{1 \dots T}$
- Then  $\Pr(Y) \propto \det(L_Y) = volume^2(\{m_Y\})$

$\Pr(\{1,2,4\}) \propto \det$ 

■	■	□	■
□	□	□	□
■	■	□	■

$$L = M^\top M$$


 $=$ 



# Modelling Behavioural Diversity for Learning in Open-Ended Games

- *Definition. G-DPP based Diversity*
- Let  $L = M^\top M$ ,  $Div(\Pi) = \mathbb{E}_{Y \sim \mathbb{P}_L}[|Y|] = Tr(I - (L + I)^{-1})$ 
  - Well-defined for duplicated policies
  - Maximized when  $m_i$  are orthonormal (assume unit length of  $m_i$ )
  - Better than matrix rank: [0.99 Rocker, 0.01 Scissor] and [0.98 Rocker, 0.02 Scissor] should be similar and not diverse

# Modelling Behavioural Diversity for Learning in Open-Ended Games

- PSRO with G-DPP Diversity
- At iteration  $T$ , let the population of opponent policies be  $\pi_{-i}^1 \dots \pi_{-i}^T$ , the payoff vector of a policy  $\pi_i$  is simply its score against the opponent population
- Define Diversity  $Div(\{\pi_i^1, \dots, \pi_i^m\}) = \mathbb{E}_{Y \sim \mathbb{P}_L}[|Y|]$  where  $L = M^\top M$  and row  $m_j$  is  $\pi_i^j$ 's payoff vector
- Now for any meta-solver  $\sigma_{-i}$  on  $\Pi_{-i}$ , the oracle is computed by:
- $O_i(\sigma) = \arg \max_{\pi_i'} \mathbb{E}_{\pi_{-i} \sim \sigma_{-i}} [u_i(\pi_i', \pi_{-i})] + \tau Div(\Pi_i \cup \{\pi_i'\})$

---

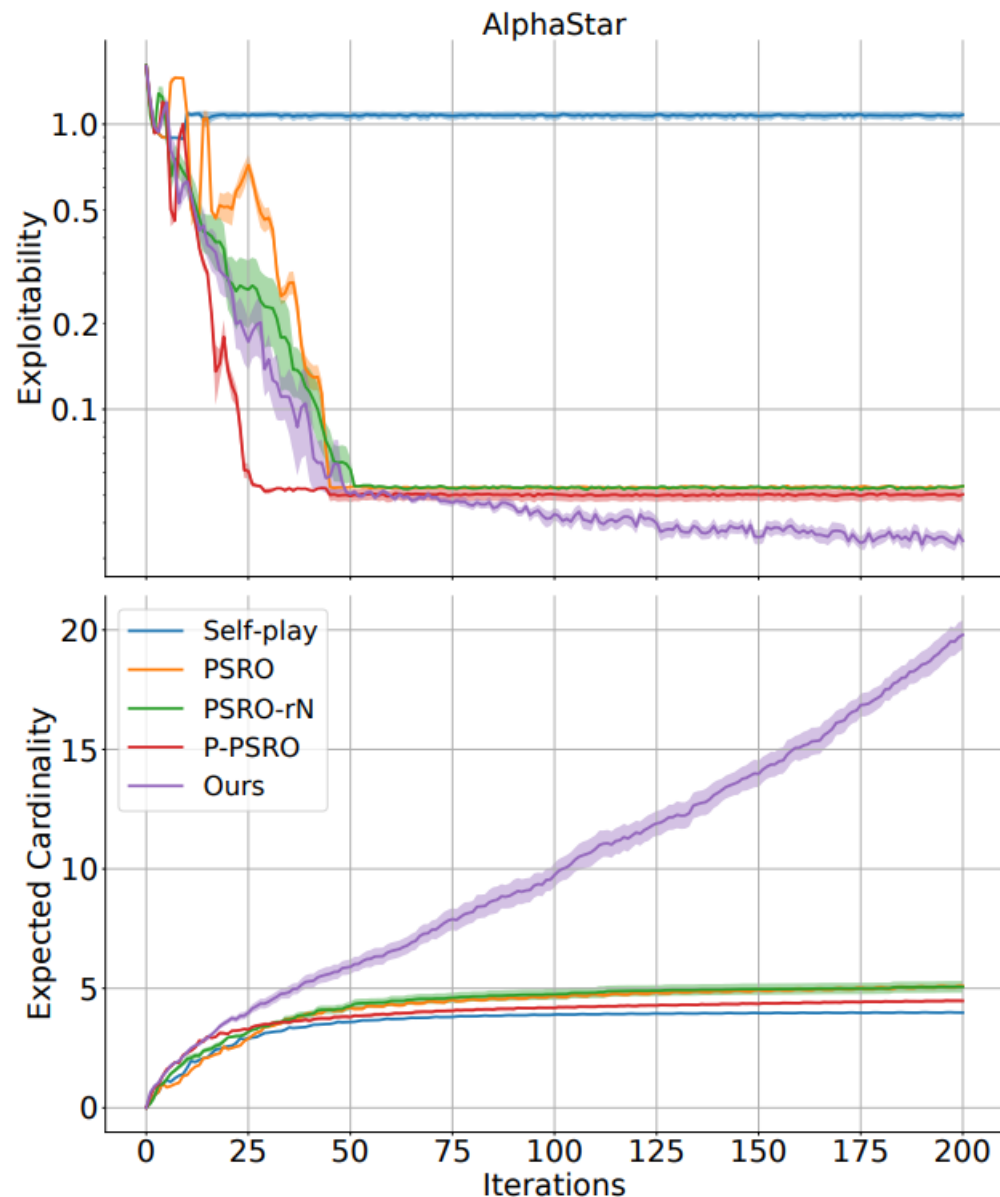
**Algorithm 2** Diverse Best Response Oracle

---

```
1: Inputs:  
2: Player Populations  $\mathbb{S}_t = \prod_{i \in \mathcal{N}} \mathbb{S}_t^i$ , with  $S_t^i \in \mathbb{S}_t^i$  parametrised by  $\theta_{S_t^i}$   
3: Metapolicies  $\pi_t = \prod_{i \in \mathcal{N}} \pi_t^i$   
4: Learning rate  $\mu$   
5: Diversity probability  $\lambda$   
6:  
7: function oracle( $\mathbb{S}_t, \pi_t, \mu, \lambda$ )  
8:   Compute  $\mathbf{BR}_{qual} = \mathbf{BR}^i(\mathbb{S}_t^{-i}, \pi_t^{-i})$   
9:   for each pure strategy  $P_j$  do:  
10:     Update meta-payoff  $\mathbf{M}_j = \mathbf{M}(\mathbb{S}_t^i \cup \{P_j\})$   
11:   Compute  $\mathbf{BR}_{div} = \arg \max_{P_j} \left( \text{Tr} \left( \mathbf{I} - (\mathbf{M}_j \mathbf{M}_j^\top + \mathbf{I})^{-1} \right) \right)$ .  
12:   Choose  $\mathbf{BR} = \mathbf{BR}_{div}$  with probability  $\lambda$  else  $\mathbf{BR} = \mathbf{BR}_{qual}$   
13:   Update  $\theta_{S_t^i} = \mu \theta_{S_t^i} + (1 - \mu) \theta_{\mathbf{BR}}$   
14:   Return:  $S_t^i$ 
```

---

the results over the **AlphaStar** game that contained the meta-payoffs for 888 RL policies



### Algorithm 3 Diverse Gradient Ascent Oracle

```

1: Inputs:
2: Player Populations  $\mathbb{S}_t = \prod_{i \in \mathcal{N}} \mathbb{S}_t^i$ , with  $S_t^i \in \mathbb{S}_t^i$  parametrised by  $\theta_{S_t^i}$ 
3: Metapolicies  $\pi_t = \prod_{i \in \mathcal{N}} \pi_t^i$ 
4: Number of training updates  $N_{train}$ 
5: Diversity weight  $\lambda$ 
6:
7: function oracle( $\mathbb{S}_t, \pi_t, N_{train}, \lambda$ )
8:   Randomly initialise a new  $S^{train}$ 
9:   for  $j = 1, \dots, N_{train}$ :
10:     Compute payoff  $p_j$  of  $S^{train}$ 
11:     Compute meta-payoff  $\mathbf{M}_j = \mathbf{M}(\mathbb{S}_t^i \cup \{S^{train}\})$ 
12:     Compute diversity  $d_j = \text{Tr}(\mathbf{I} - (\mathbf{M}_j \mathbf{M}_j^\top + \mathbf{I})^{-1})$ 
13:     Compute loss  $l_j = -(1 - \lambda)p_j - \lambda d_j$ 
14:     Update  $\theta_{S^{train}}$  to minimise  $l_j$  using a gradient based optimization method
15:   Return:  $S^{train}$ 

```

### Non-transitive Mixture Model

$$u_1(\pi_1, \pi_2) = \pi_1^\top V \pi_2 + k(|\pi_1|_1 - |\pi_2|_1)$$

$V$ : every gaussian beats three neighbors in clockwise order

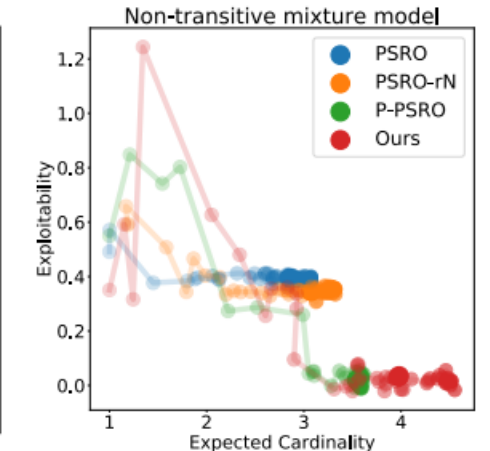
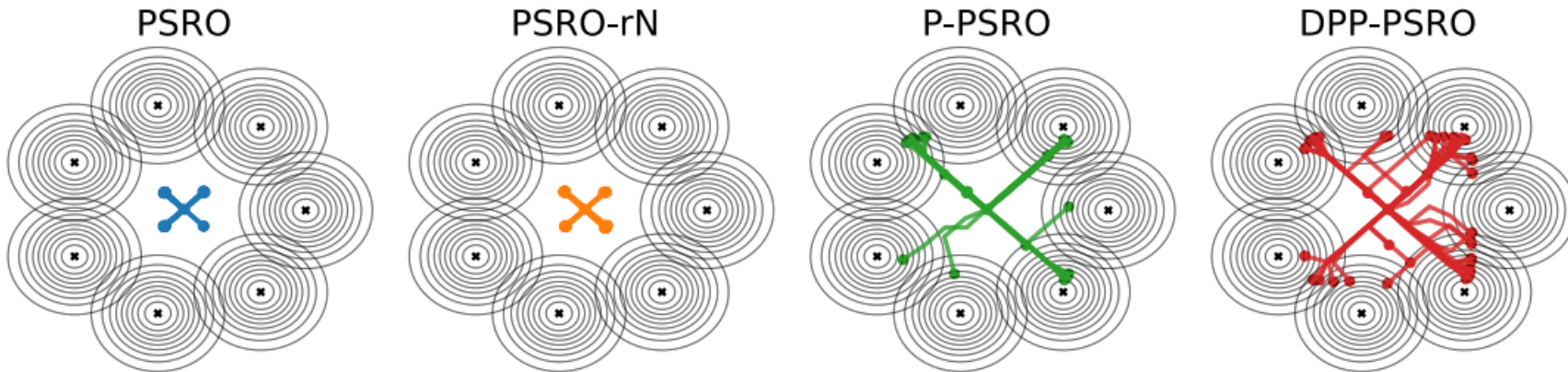


Figure 3: Non transitive mixture model. Exploration trajectories during training and Performance vs. Diversity comparisons

---

**Algorithm 4** Diverse Zero-order Oracle

---

1: **Inputs:**

2: Player Populations  $\mathbb{S}_t = \prod_{i \in \mathcal{N}} \mathbb{S}_t^i$ , with  $S_t^i \in \mathbb{S}_t^i$  parametrised by  $\theta_{S_t^i}$

3: Metapolicies  $\pi_t = \prod_{i \in \mathcal{N}} \pi_t^i$

4: Learning rate  $\mu$

5: Noise parameter  $\sigma$

6: Diversity weight  $\lambda$

7:

8: **function** oracle( $\mathbb{S}_t, \pi_t, \mu, \sigma, \lambda$ )

9:     Sample a single  $S_t^i \in \mathbb{S}_t^i$  with probability  $\pi_t^i$

For RL, replace  $S_t^i$  with a BR against  $\pi_t^{-i}$

10:    **for** perturbation  $j \in 1, 2, \dots$  **do:**

11:        $S_j = \text{RandomPerturbation}(S_t^i, \mu, \sigma)$

12:       Update meta-payoff  $\mathbf{M}_j = \mathbf{M}(\mathbb{S}_t^i \cup \{S_j\})$

13:       Compute payoff  $p_j$  of  $S_j$

14:       Compute diversity  $d_j = \text{Tr}(\mathbf{I} - (\mathbf{M}_j \mathbf{M}_j^\top + \mathbf{I})^{-1})$ .

15:     **Return:**  $\arg \max_{S_j} (1 - \lambda)p_j + \lambda d_j$

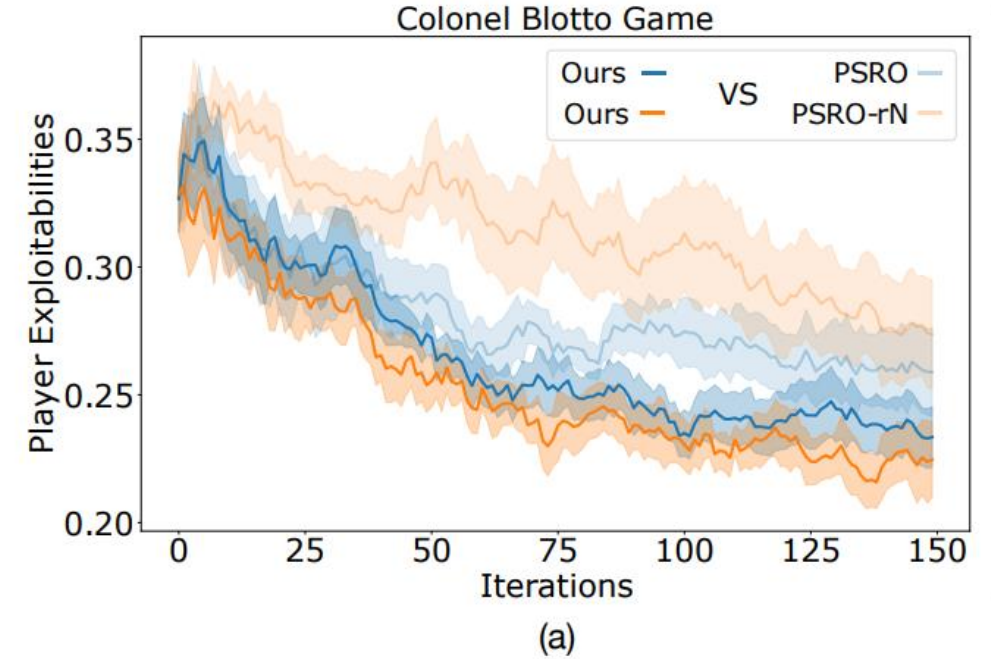
16: **function** RandomPerturbation( $S_t^i, \mu, \sigma$ )

17:     Generate noise  $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

18:     Add noise  $\theta_{S_t^i} = \mu \theta_{S_t^i} + (1 - \mu) \varepsilon$

19:     **Return**  $S_t^i$

---



**Colonel Blotto.** 10 coins on 3 areas, winner takes the area.

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

- General idea: introduce two diversity measure into oracle object function:

- $O_i(\sigma) = \arg \max_{\pi_i} u_i(\pi_i, \sigma_{-i}) + \lambda_1 Div_{occ}(\pi_i) + \lambda_2 Div_{rew}(\pi_i)$

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

- Behavioral Diversity
- Let  $\rho_\pi(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s \mid \pi)$  and  $\rho_\pi(s, a) = \rho_\pi(s) \pi(a \mid s)$  be the *Joint Occupancy Measure* (later one will be used)
- Define Behavioral Diversity as (assume meta strategy  $\sigma$ )
- $Div_{occ}(\pi_i) = D_f(\rho_{\pi_i, \sigma_{-i}} \parallel \rho_\sigma)$
- In practice, using a RND like method. Train a function  $f(s, a; \theta)$  to fit a random NN function  $f(s, a; \hat{\theta})$ , with  $(s, a)$  sampled based on  $\sigma$ . Then use  $r^{int}(s, a) = |f(s, a; \theta) - f(s, a; \hat{\theta})|$  as an intrinsic reward



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

- Reward Diversity based on the payoff matrix  $M$  (i.e.,  $M\Pi^T$ )
  - The idea is to enlarge the gamescape  $\{\text{convex combination of } m_i^T\}$
- $Div_{rew}(\pi'_i) = \min_{\beta \geq 0, e^\top \beta = 1} ||M^\top \beta - m^{T+1}||_2^2$  where  $m_{T+1} = \left\{u_i(\pi'_i, \pi_{-i}^j)\right\}_{j=1}^T$ , with a lower bound:
- $Div_{rew}(\pi'_i) \geq F(\pi'_i) = \frac{\lambda_{min}^2(M)(1 - e^\top (M^\top)^* m^{T+1})^2}{T} + ||(I - M^\top (M^\top)^*)m^{T+1}||_2^2$ 
  - $(M^\top)^*$  is the Moore–Penrose pseudoinverse
- For larger game, we have  $\frac{\partial F}{\partial \theta} = \sum_{j=1}^T \frac{\partial F}{\partial m_j^{T+1}} \frac{\partial m_j^{T+1}}{\partial \theta}$ . In practice we sample  $j$  according to  $|\frac{\partial F}{\partial m_j^{T+1}}|$  and apply gradient ascent/descent

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

- Finally,  $O_i(\sigma) = \arg \max_{\pi_i} u_i(\pi_i, \sigma_{-i}) + \lambda_1 Div_{occ}(\pi_i) + \lambda_2 Div_{rew}(\pi_i)$ 
  - When optimizing  $Div_{rew}$ , the  $\theta$  initialization is sensitive, so first use the first two terms to get a good initialization

---

## Algorithm 1 Gradient-based Optimization for Unified Diverse Response

---

- 1: **Input:** population  $\mathfrak{P}_i$  for each  $i$ , meta-game  $\mathbf{A}_{\mathfrak{P}_i \times \mathfrak{P}_{-i}}$ , state-action dataset  $\{(s, \mathbf{a})\}$ , weights  $\lambda_1$  and  $\lambda_2$
- 2:  $\sigma_i, \sigma_{-i} \leftarrow$  Nash on  $\mathbf{A}_{\mathfrak{P}_i \times \mathfrak{P}_{-i}}$
- 3:  $\pi_E \leftarrow$  Aggregate according to  $\sigma_i, \sigma_{-i}$
- 4:  $r_i^{int}(s, \mathbf{a}) \leftarrow$  Train fixed reward from distribution  $(s, \mathbf{a}) \sim \rho_{\pi_E}$  by EBM or prediction errors.
- 5:  $\theta^* \leftarrow$  Train  $\pi'_i(\theta)$  against fixed opponent policies  $\pi_{E_{-i}}$  by single-agent RL algorithm with  $r_i(s, \mathbf{a}) = r_i^{ext}(s, \mathbf{a}) + \lambda_1 r_i^{int}(s, \mathbf{a})$ ,  $r_i^{ext}$  is the original reward function.
- 6:  $\frac{\partial F}{\partial \mathbf{a}_{n+1}} \leftarrow$  Simulate the reward row vector  $\mathbf{a}_{n+1}$  using new  $\pi'_i(\theta)$  and compute  $\frac{\partial F}{\partial \mathbf{a}_{n+1}}$  analytically.
- 7:  $\hat{\theta} \leftarrow$  Train  $\pi'_i(\theta^*)$  against a new mixture distribution  $\sigma_{-i} + \lambda_2 \frac{\partial F}{\partial \mathbf{a}_{n+1}}$  of opponent policies.
- 8: **Output:** policy  $\pi'_i(\hat{\theta})$

$Div_{rew}$  is sensitive because **this term also depends on  $\theta$**

Actually  $Div_{rew}$  is **maximizing** a convex function, inducing non-stability

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

- Evaluation

- Exploitability

- Population Effectivity:  $PE(\Pi_i^T) = \min_{\pi_{-i}} \max_{e^\top p=1, p \geq 0} \sum_{j=1}^T p_j u_j(\pi_i^j, \pi_{-i})$

- $PE$  is better for diversity measure while Exploitability measures a single joint policy

- Compute PE is similar
  - to a unilateral PSRO

---

**Algorithm 2**  $PE(n)$ 

---

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

---

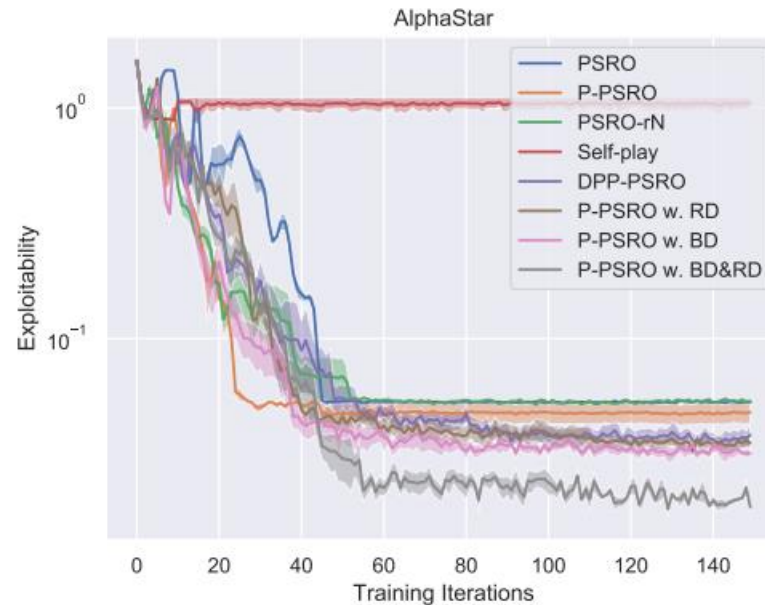
- 1) AlphaStar meta-game with 888 policies.

---

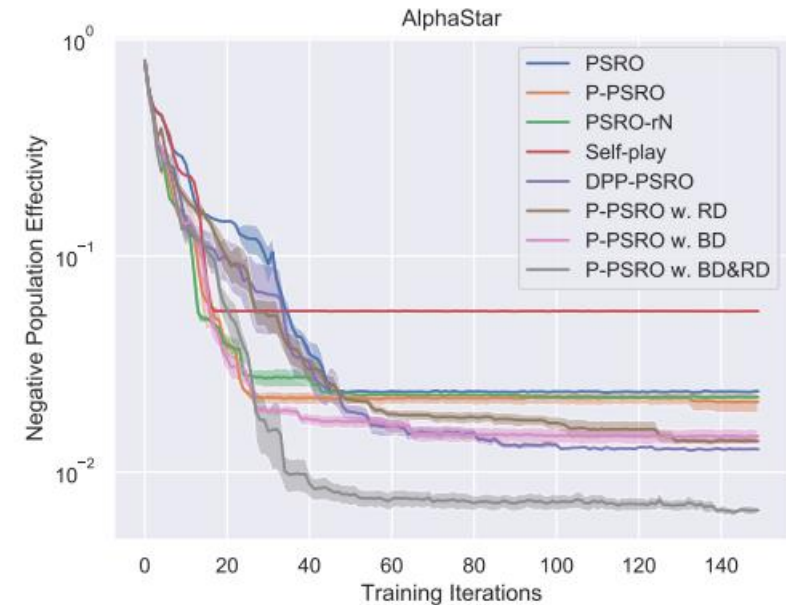
**Algorithm 3** Optimization for Matrix Games

---

- 1: **Input:** population  $\mathfrak{P}_i$  for each  $i$ , meta-game  $\mathbf{A}_{\mathfrak{P}_i \times \mathfrak{P}_{-i}}$ , weights  $\lambda_1$  and  $\lambda_2$ , learning rate  $\mu$
  - 2:  $\sigma_i, \sigma_{-i} \leftarrow$  Nash on  $\mathbf{A}_{\mathfrak{P}_i \times \mathfrak{P}_{-i}}$
  - 3:  $\pi_E \leftarrow$  Aggregate according to  $\sigma_i, \sigma_{-i}$
  - 4:  $\pi'_i(\theta) \leftarrow$  Initialize a new random policy for player  $i$
  - 5:  $\mathbf{BR}_{qual} \leftarrow$  Compute best response against mixture of opponents  $\sum_k \sigma_{-i}^k \phi_i(\cdot, \pi_{-i}^k)$
  - 6: **while** the reward  $p$  improvement does not meet the threshold **do**
  - 7:    $\mathbf{BR}_{occ} \leftarrow \arg \max_{s_j} D_f(s_j || \pi_i)$  for each pure strategy  $s_j$
  - 8:    $\mathbf{BR} \leftarrow$  Choose  $\mathbf{BR} = \mathbf{BR}_{occ}$  with probability  $\lambda_1$  else  $\mathbf{BR} = \mathbf{BR}_{qual}$
  - 9:    $\theta \leftarrow \mu\theta + (1 - \mu)\theta_{\mathbf{BR}}$
  - 10:    $p \leftarrow$  Compute the payoff  $p$  after the update according to  $\sum_k \sigma_{-i}^k \phi_i(\pi'_i(\theta), \pi_{-i}^k)$
  - 11: **end while**
  - 12:  $\mathbf{BR}_{rew} \leftarrow \arg \max_{s_j} F(s_j)$  for each pure strategy  $s_j$  with probability  $\lambda_2$  else  $\mathbf{BR}_{qual}$
  - 13:  $\theta \leftarrow \mu\theta + (1 - \mu)\theta_{\mathbf{BR}_{rew}}$
  - 14: **Output:** policy  $\pi'_i(\hat{\theta})$
- 



(a)



(b)



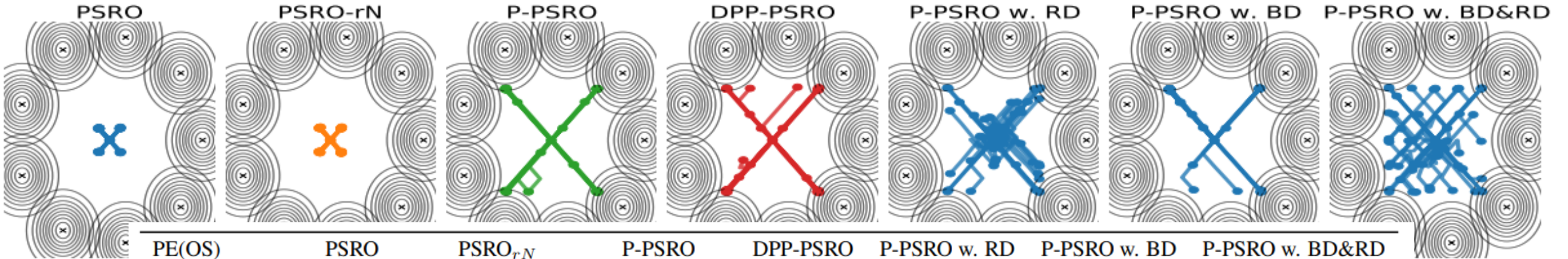
## • 2) Non-transitive Mixture

RD is extremely important!  
Much more than BD.

During training  $\lambda_2$  is set to be **1500** while  $\lambda_1$  is set to 1

### Algorithm 4 Optimization for Differential Games

- 1: **Input:** population  $\mathfrak{P}_i$  for each  $i$ , meta-game  $\mathbf{A}_{\mathfrak{P}_i \times \mathfrak{P}_{-i}}$ , weights  $\lambda_1$  and  $\lambda_2$ , number of gradient updates  $N_{train}$
- 2:  $\sigma_i, \sigma_{-i} \leftarrow$  Nash on  $\mathbf{A}_{\mathfrak{P}_i \times \mathfrak{P}_{-i}}$
- 3:  $\pi_E = (\pi_i, \pi_{E-i}) \leftarrow$  Aggregate according to  $\sigma_i, \sigma_{-i}$
- 4:  $\pi'_i(\theta) \leftarrow$  Initialize a new random policy for player  $i$
- 5: **for**  $j = 1$  **to**  $N_{train}$  **do**
- 6:    $p_j \leftarrow$  Compute payoff against the mixture of opponents  $p_j = \sum_k \sigma_{-i}^k \phi_i(\pi'_i, \pi_{-i}^k)$
- 7:    $d_j^{occ} \leftarrow$  Compute BD  $d_j^{occ} = D_f(\pi'_i || \pi_i)$  as the  $f$ -divergence between  $\pi_i$  and  $\pi'_i$
- 8:    $\mathbf{a}_j \leftarrow$  Compute new reward vector as  $\mathbf{a}_j = (\phi_i(\pi'_i, \pi_{-i}^k))_{k=1}^{|\mathfrak{P}_{-i}|}$
- 9:    $d_j^{rew} \leftarrow$  Compute the lower bound of RD as  $F(\mathbf{a}_j)$  according to Theorem 2
- 10:    $l_j \leftarrow -(p_j + \lambda_1 d_j^{occ} + \lambda_2 d_j^{rew})$
- 11:   Update  $\theta$  to minimize  $l_j$  by backpropagation
- 12: **end for**
- 13: **Output:** policy  $\pi'_i(\theta)$



PE(OS)	PSRO	PSRO <sub>rN</sub>	P-PSRO	DPP-PSRO	P-PSRO w. RD	P-PSRO w. BD	P-PSRO w. BD&RD
PE(5)	$-2.11 \pm 0.13$	$-2.11 \pm 0.14$	$40.20 \pm 0.09$	$40.49 \pm 0.07$	$40.42 \pm 0.08$	$40.19 \pm 0.10$	<b><math>40.54 \pm 0.12</math></b>
PE(10)	$-13.18 \pm 0.28$	$-13.18 \pm 0.28$	$29.14 \pm 0.19$	$29.45 \pm 0.13$	$29.55 \pm 0.13$	$29.05 \pm 0.21$	<b><math>29.63 \pm 0.26</math></b>
PE(15)	$-31.17 \pm 0.37$	$-31.17 \pm 0.37$	$11.03 \pm 0.26$	$11.49 \pm 0.21$	<b><math>11.63 \pm 0.15</math></b>	$10.97 \pm 0.29$	$11.57 \pm 0.33$
PE(20)	$-49.12 \pm 0.23$	$-49.12 \pm 0.24$	$-6.78 \pm 0.14$	$-6.41 \pm 0.10$	$-6.52 \pm 0.10$	$-7.03 \pm 0.21$	<b><math>-6.37 \pm 0.24</math></b>
PE(25)	$-54.59 \pm 0.02$	$-54.59 \pm 0.01$	$-12.51 \pm 0.05$	$-12.28 \pm 0.04$	$-12.42 \pm 0.03$	$-12.58 \pm 0.02$	<b><math>-12.18 \pm 0.04</math></b>
Expl	$54.66 \pm 0.06$	$54.90 \pm 0.10$	<b><math>13.21 \pm 0.29</math></b>	$13.24 \pm 0.33$	$13.77 \pm 0.40$	$41.132 \pm 1.06$	$13.26 \pm 0.24$

- 3) Google Research Football

