

# TRAINING A POPULATION OF REINFORCEMENT LEARNING AGENTS

Dr. Yaodong Yang

[www.yangyaodong.com](http://www.yangyaodong.com)

01/2022

The logo for King's College London, featuring the text "KING'S" in a large serif font, "College" in a smaller italicized serif font, and "LONDON" in a bold serif font, all in white on a red background.

KING'S  
*College*  
LONDON



# Intelligence is learning from mistakes!

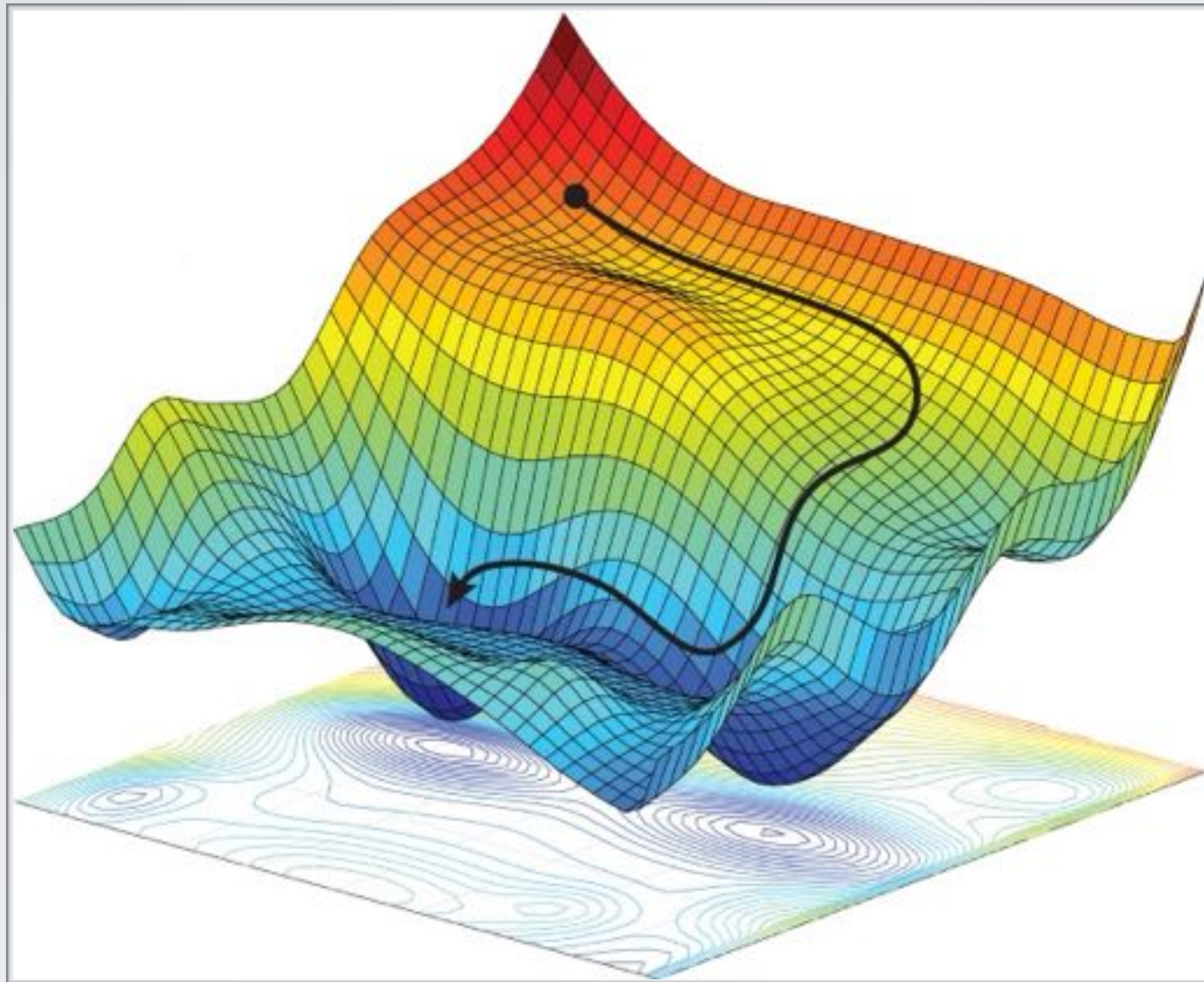


“... if a machine is expected to be infallible, it cannot also be intelligent. There are several mathematical theorems which say almost exactly that. But these theorems say nothing about how much intelligence may be displayed if a machine makes no pretence at infallibility...”

— Alan Turing, 1947

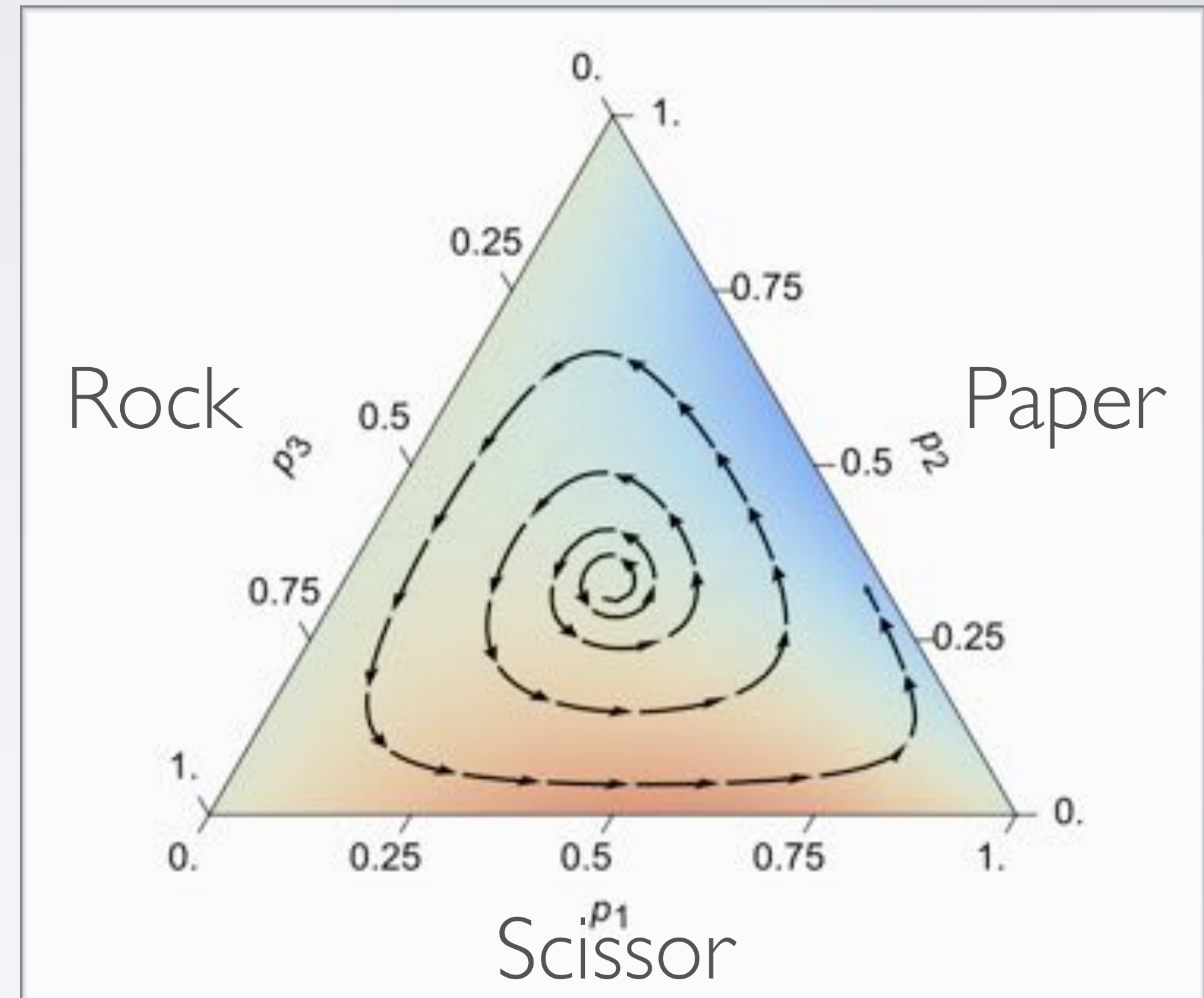
# Multi-Agent Intelligence: learning from mistakes from multiple agents' interactions

Normal machine learning problems:



$$\|\nabla f(x)\|_2 = 0$$

Multi-agent Learning problems:



agents learn to reach some equilibrium

# Contents

- **Formulation & Challenges of Training A Population of RL Agents**
- **Training A Population of RL Agents on Fully-Cooperative Games**
- **Training A Population of RL Agents on Zero-Sum Games**
- **Some System Level Thinkings**
- **Conclusions**

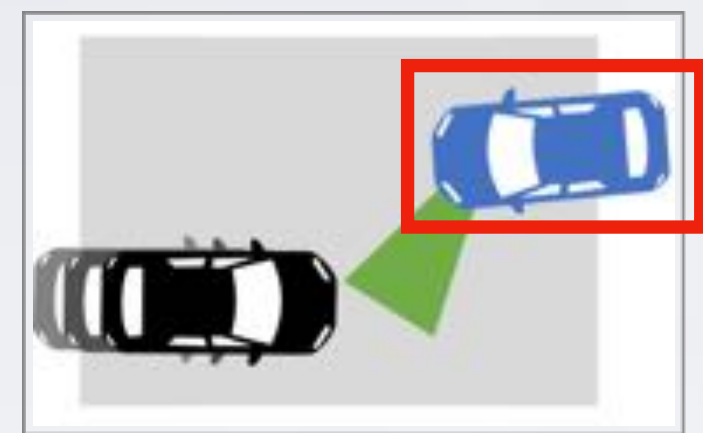
# Contents

- **Formulation & Challenges of Training A Population of RL Agents**
- Training A Population of RL Agents on Fully-Cooperative Games
- Training A Population of RL Agents on Zero-Sum Games
- Some System Level Thinkings
- Conclusions

# A Naive Example of Multi-Agent Learning

Traffic intersection is naturally a multi-agent system. From each driver's perspective, in order to perform the optimal action, he must take into account others' behaviours.

what you see



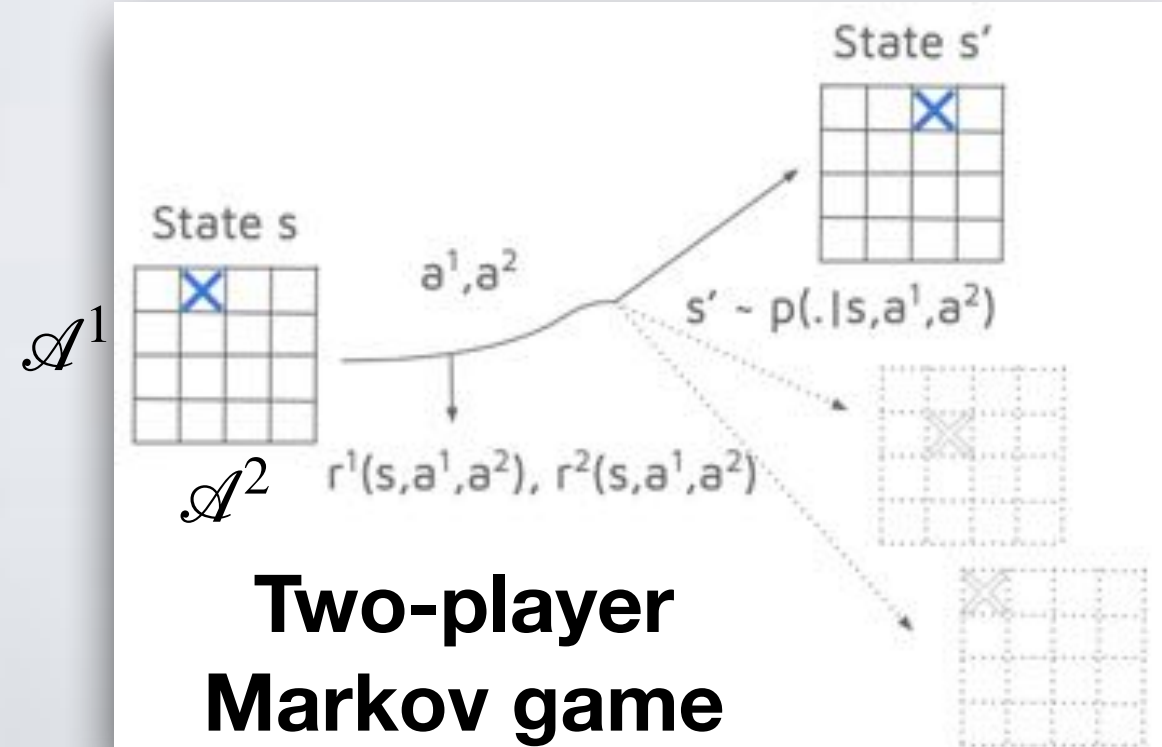
scenario

Yield  
Rush

	Yield	Rush
Yield	(0, 0)	(1, 2)
Rush	(2, 1)	(0, 0)

normal-form game

what the computer sees

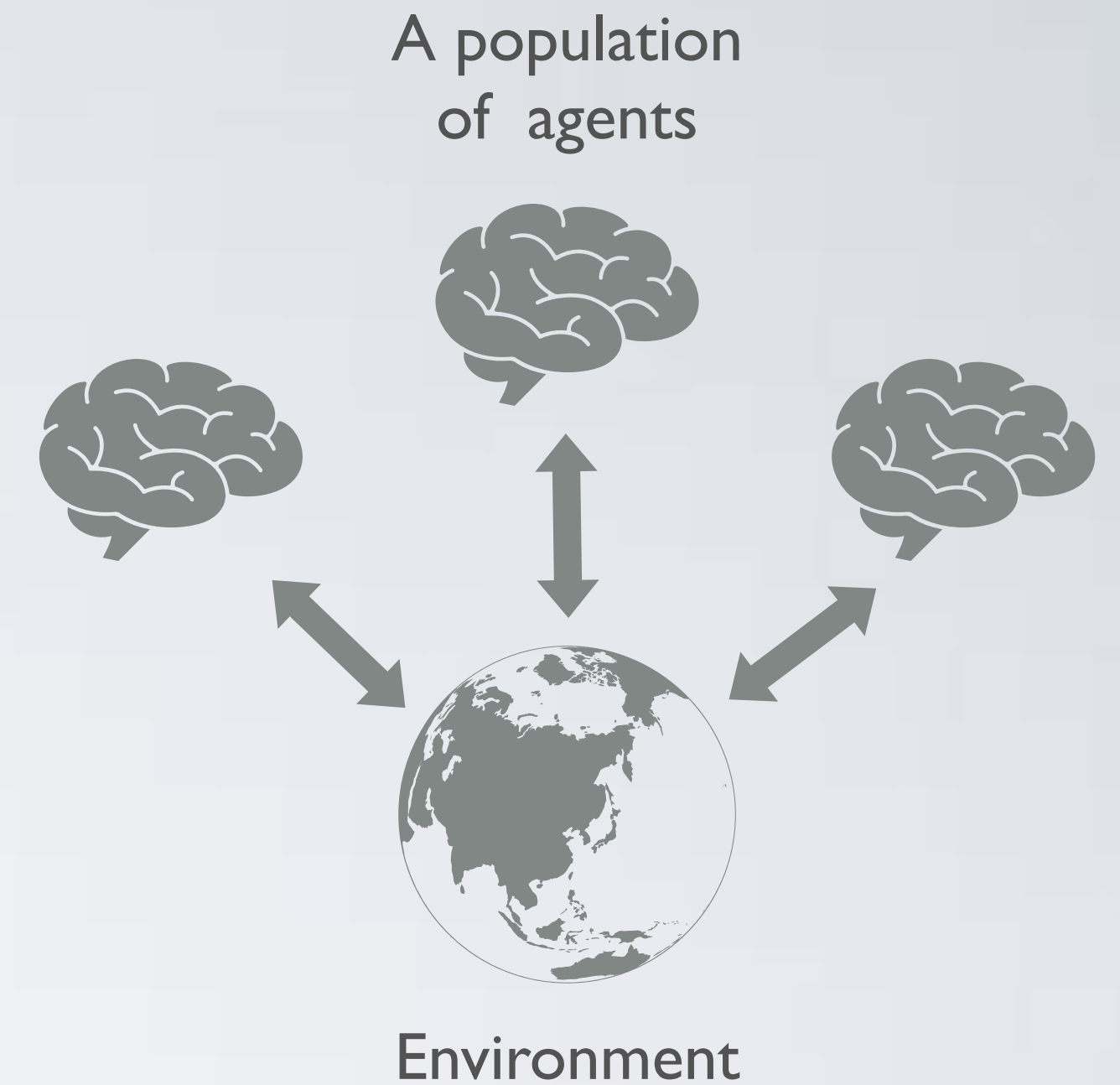


- When the drivers are rational, they will reach the outcome of a Nash Equilibrium. It is the outcome of interaction. Knowing it can predict future.
- Real-world decision making has cooperation & competition. For each agent, how to infer the belief of the other agents and make the optimal action is critical.
- The concept of using traffic light is in fact a **correlated equilibrium** in game theory.
- Many-agent system is when agents  $\gg 2$ . It is a very challenging problem to compute equilibrium, thus making decisions.

# Multi-Agent Reinforcement Learning

- Modelled by a Stochastic Game  $(\mathcal{S}, \mathcal{A}^{\{1,\dots,n\}}, \mathcal{R}^{\{1,\dots,n\}}, \mathcal{T}, \mathcal{P}_0, \gamma)$

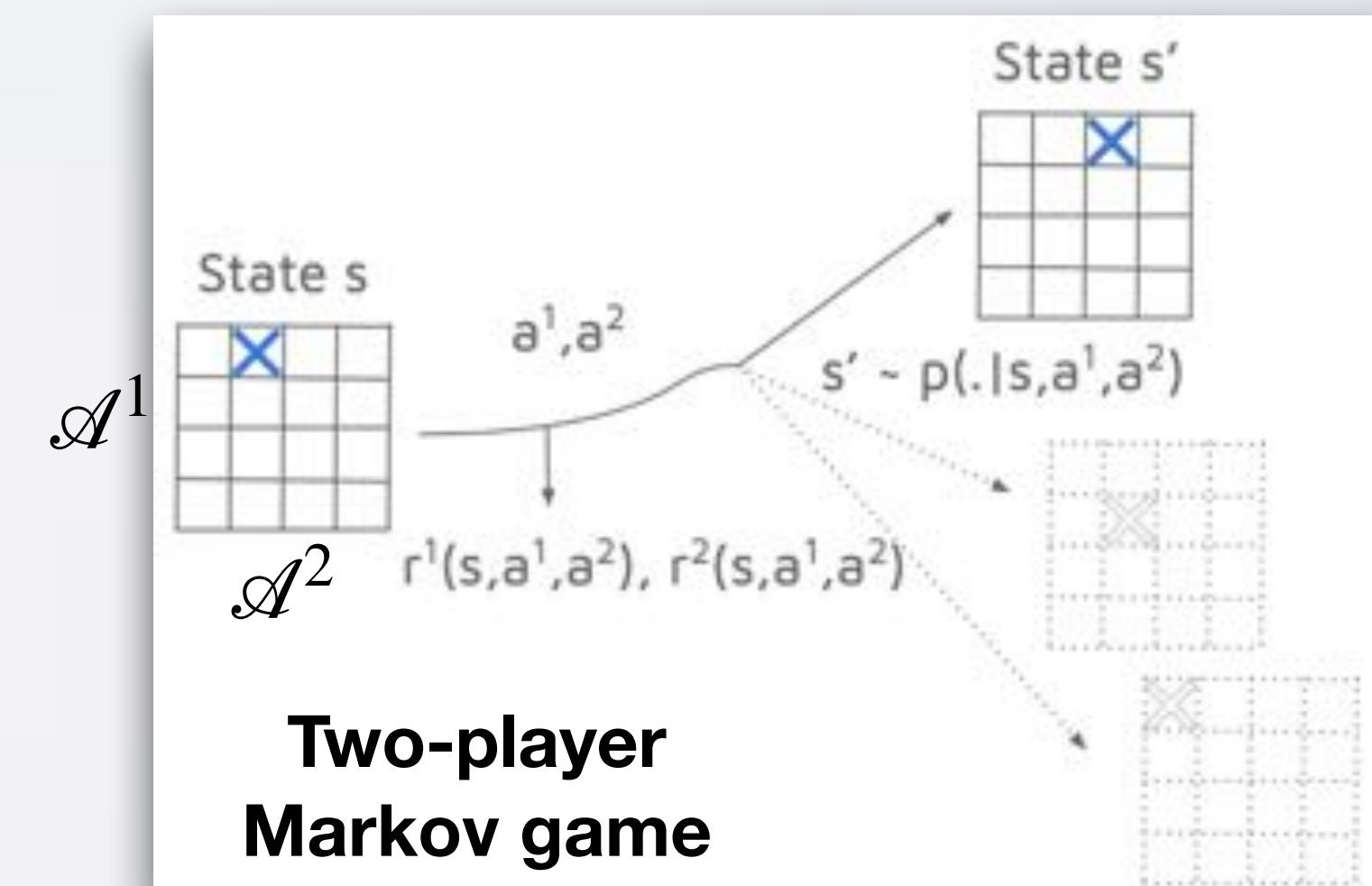
- $\mathcal{S}$  denotes the state space,
- $\mathcal{A}$  is the joint-action space  $\mathcal{A}^1 \times \dots \times \mathcal{A}^n$ ,
- $\mathcal{R}^i = \mathcal{R}^i(s, a^i, a^{-i})$  is the reward function for the i-th agent,
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$  is the transition function based on the joint action,
- $\mathcal{P}_0$  is the distribution of the initial state,  $\gamma$  is a discount factor.
- **Special case:**  $n = 1 \rightarrow$  single-agent MDP,  $|\mathcal{S}| = 1 \rightarrow$  normal-form game
- **Dec-POMDP:** assume state is not directly observed, but agents have same reward function.



- Each agent tries to maximise its expected long-term reward:

$$V_{i,\pi}(s) = \sum_{t=0}^{\infty} \gamma^t \mathbf{E}_{\pi, \mathcal{P}} \{R_{i,t} | s_0 = s, \pi\}, \pi = [\pi_1, \dots, \pi_N]$$

$$Q_{i,\pi}(s, \mathbf{a}) = R_i(s, \mathbf{a}) + \gamma \mathbf{E}_{s' \sim p} [V_{i,\pi}(s')]$$



# Multi-Agent Reinforcement Learning

- Value-based method:

- The sense of optimality changes, now it depends on other agents !

$$Q_{i,t+1}(s_t, \pi_t) = Q_{i,t}(s_t, \pi_t) + \alpha [R_{i,t+1} + \gamma \cdot \mathbf{eval}_i\{Q_{\cdot,t}(s_{t+1}, \cdot)\} - Q_{i,t}(s_t, \pi_t)]$$
$$\pi_{i,t}(s, \cdot) = \mathbf{solve}_i\{Q_{\cdot,t}(s_t, \cdot)\}$$

- ◆ Fully-cooperative game: agents share the same reward function

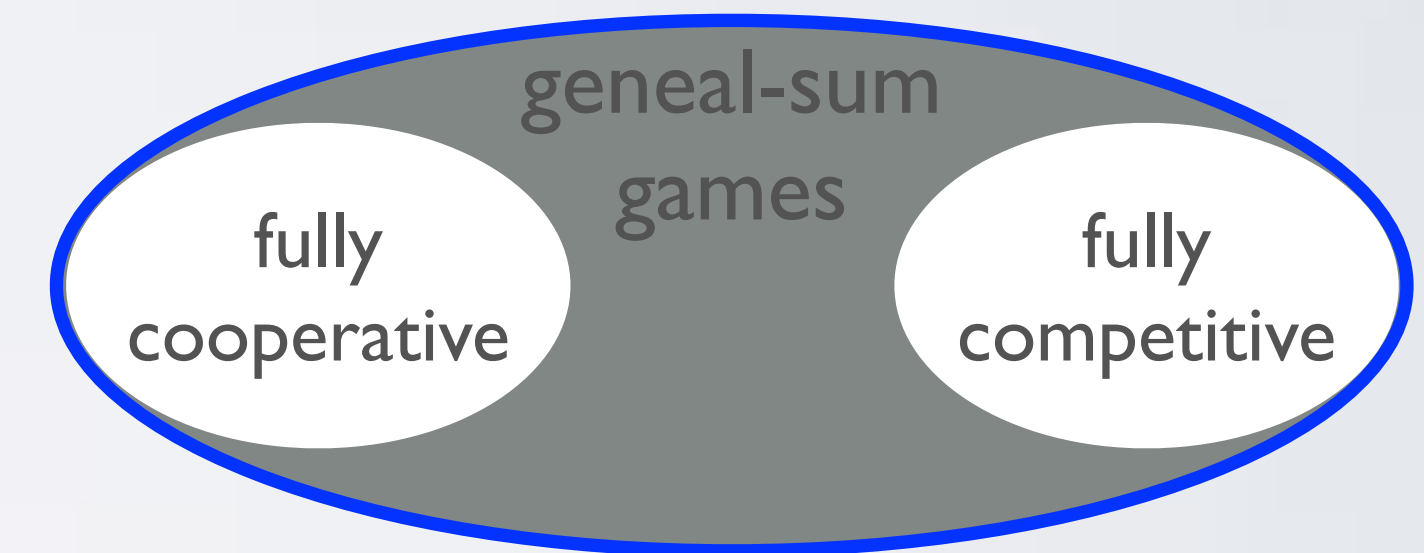
$$\mathbf{eval}_i\{Q_{\cdot,t}(s_{t+1}, \cdot)\} = \max_a Q_{i,t}(s_{t+1}, a)$$

$$\mathbf{solve}_i\{Q_{\cdot,t}(s_t, \cdot)\} = \arg \max_{a_i} \left( \max_{a_{-i}} Q_{i,t}(s_t, a_i, a_{-i}) \right)$$

- ◆ Fully-competitive game: sum of agents' reward is zero

$$\mathbf{eval}_i\{Q_{\cdot,t}(s_{t+1}, \cdot)\} = \max_{\pi_i} \min_{a_{-i}} \mathbf{E}_{\pi_i} [Q_{i,t}(s_t, a_i, a_{-i})]$$

$$\mathbf{solve}_i\{Q_{\cdot,t}(s_t, \cdot)\} = \arg \max_{\pi_i} \min_{a_{-i}} \mathbf{E}_{\pi_i} [Q_{i,t}(s_t, a_i, a_{-i})]$$



- Assuming agents share the either the same or completely opposite interest is a strong assumption.



# Nash Equilibrium

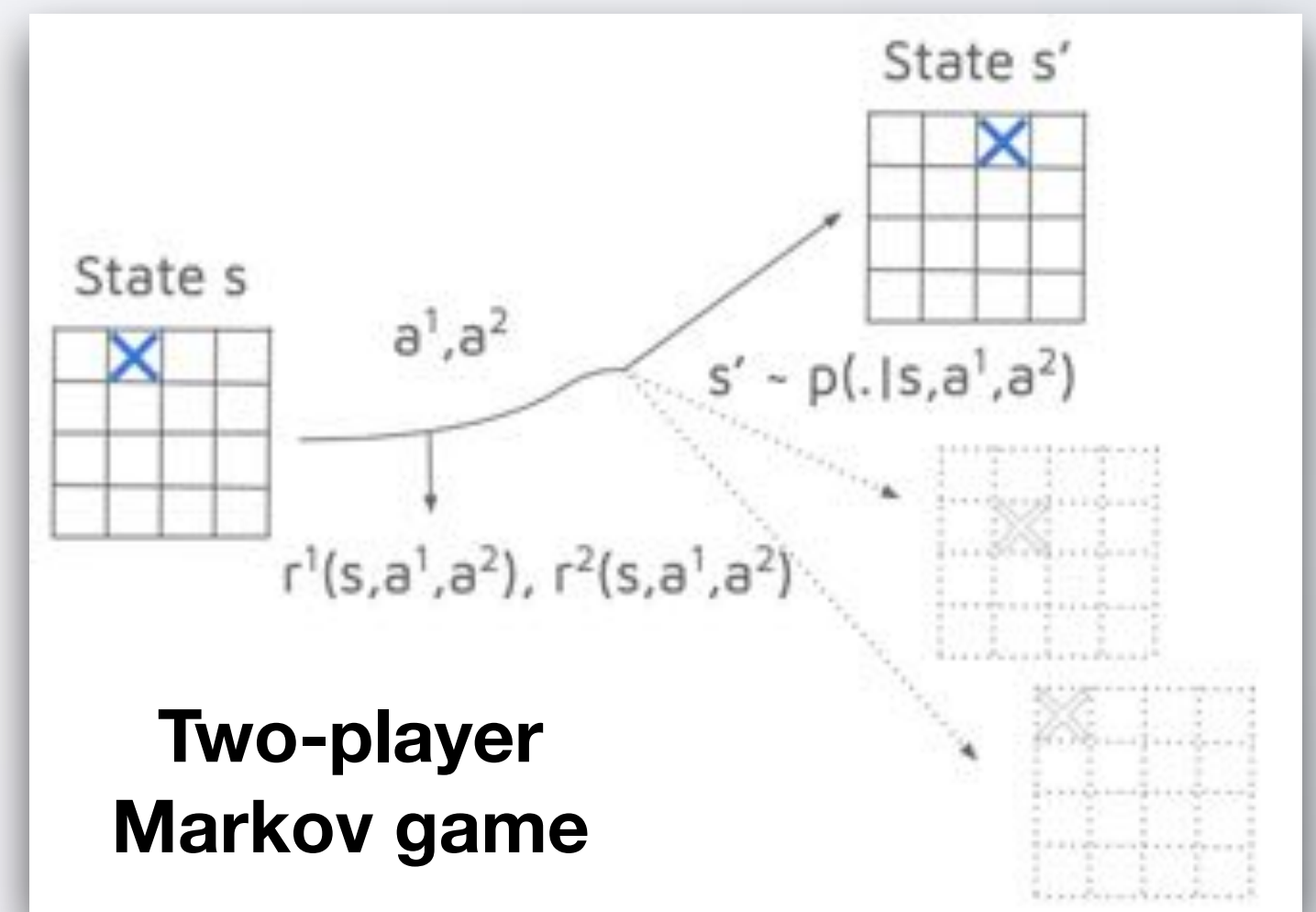
- Let  $n$  players,  $S = S_1 \times \dots \times S_n$  is the joint strategy profile,  $u_i : S \rightarrow \mathbb{R}$  is the utility function, Nash equilibrium is

$$\mathbf{E}_{s_j \sim \mu_j \forall j \in [n]} \left[ u_i(s_1, \dots, s_n) \right] \geq \mathbf{E}_{\substack{s_i \sim \mu'_i \\ s_{-j} \sim \mu_{-j}}} \left[ u_i(s_1, \dots, s_n) \right] \quad \forall \mu'_i \in \Delta_{S_i}$$

- Mixed strategy Nash equilibrium **always exists in finite player finite action** games.
- For continuous utility games, the strategy set needs to be compact
- Note that  $\mu'_i \in \Delta_{S_i}$  can be replaced by  $a \in S_i$  because deviation is at most a pure strategy !
- In Markov game, the solution concept is **Markov Perfect Equilibrium**.

**Definition 3 (Behavioral Strategy).** A behavioral strategy of an agent  $i$  is  $\pi^i : \mathcal{S} \rightarrow \Delta(\mathbb{A}^i)$ , i.e.,  $\forall s \in \mathcal{S}, \pi^i(s)$  is a probability distribution on  $\mathbb{A}^i$ .

**Definition 5 (Markov Perfect Equilibrium (MPE)).** A behavioral strategy profile  $\pi$  is called a Markov Perfect Equilibrium if

$$\forall s \in \mathcal{S}, i \in [n], \forall \tilde{\pi}^i \in \Delta_{A^i}^S, V^{\pi^i, \pi^{-i}}(s) \geq V^{\tilde{\pi}^i, \pi^{-i}}(s).$$


# Nash Equilibrium to MARL

- Value-based method:

$$\pi_{i,t}(s, \cdot) = \mathbf{solve}_i \left\{ Q_{\cdot,t}(s_t, \cdot) \right\}$$

$$Q_{i,t+1}(s_k, \pi_t) = Q_{i,t}(s_t, \pi_t) + \alpha \left[ R_{i,t+1} + \gamma \cdot \mathbf{eval}_i \left\{ Q_{\cdot,t}(s_{t+1}, \cdot) \right\} - Q_{i,t}(s_t, \pi_t) \right]$$

- Nash-Q Learning [Hu. et al 2003] — Using Nash Equilibrium as the optima to guide agents' policies

1. Solve the Nash Equilibrium for the current stage game

$$\mathbf{solve}_i \left\{ Q_{\cdot,t}(s, \cdot) \right\} = \mathbf{Nash}_i \left\{ Q_{\cdot,t}(s_t, \cdot) \right\}$$

2. Improve the estimation of the Q-function by the Nash value function.

$$\mathbf{eval}_i \left\{ Q_{\cdot,t}(s, \cdot) \right\} = V_i(s, \mathbf{Nash} \left\{ Q_{\cdot,t}(s_t, \cdot) \right\})$$

- Nash-Q algorithm [Junling 2003] computes Nash for the normal-form game at each state

- Nash-Q operator  $\mathcal{H}^{\text{Nash}} \mathbf{Q}(s, \mathbf{a}) = \mathbf{E}_{s'} [R(s, \mathbf{a}) + \gamma \mathbf{V}^{\text{Nash}}(s')]$  is a contraction mapping.

# Multi-Agent Intelligence Components

Multi-Agent Intelligence

Autonomous Driving  
Gaming AI  
Metaverse  
Smart Grid / City  
...

||

Algorithmic Game Theory Fundamentals

Equilibrium sets  
Solution Concept  
Learning Dynamics Analysis  
Mechanism Design  
...

+

Machine Learning Techniques

Reinforcement Learning  
Deep Learning  
Representation Learning  
...

# Complexity of Computing Nash Equilibrium in Normal-Form Games

- Solving Nash Equilibrium is very challenging !
  - The solution concept of Nash comes from game theory but it is not their main interest to find solutions.
  - Complexity of solving two-player Nash is **PPAD-Hard (intractable unless P=NP)**.
  - How to scale up multi-agent solution is open-question.
  - Approximate solution is still under development.
$$R_i(a_i, a_{-i}) \geq R_i(a'_i, a_{-i}) - \epsilon$$
$$\epsilon = .75 \rightarrow .50 \rightarrow .38 \rightarrow .37 \rightarrow .3393$$
 [Tsaknakis 2008]
  - Equilibrium selection is problematic, how to coordinate agents to agree on Nash during training is unknown.
  - Nash equilibrium assumes perfect rationality, but can be unrealistic in the real world.
- More complexity results of solving Nash [Shoham 2007, sec 4][Conitzer 2002]
  - Two-player general-sum normal-form game:
    - Compute NE → **PPAD-Hard**
    - Count number of NE → **#P-Hard**
    - Check uniqueness of NE → **NP-Hard**
    - Guaranteed payoff for one player → **NP-Hard**
    - Guaranteed sum of agents payoffs → **NP-Hard**
    - Check action inclusion / exclusion in NE → **NP-Hard**
  - Stochastic game:
    - Check pure-strategy NE existence → **PSPACE-Hard**
    - Best response for arbitrary strategy → **Not Turing-computable, even can not be implemented by a Turing PC.**
    - **It holds for two-player symmetrical game with finite time length.**

# A Gentle Touch on PPAD

- Complexity theory 101 — an intuitive explanation:

- Recall the NP for a decision problem as

**Definition 4.2.1 (NP)** *A decision problem  $Q$  is in NP if there exists a polynomial time algorithm  $V(I, X)$  such that*

- 1. If  $I$  is a YES instance of  $Q$  then there exists some  $X$  such that  $|X|$  is polynomial in  $|I|$  and  $V(I, X) = YES$*
- 2. If  $I$  is a NO instance of  $Q$  then  $V(I, X) = NO$  for all  $X$*

- But the decision problem of “is there a Nash equilibrium?” is **always true** proved by Nash himself.
- We need a new complexity class of **Functional NP (FNP)** to describe the search problems: not only do solutions have to be verified in P-time, but also to find a solution!
- **Two-player Nash will be FNP** because we can check whether Nash is true by checking the best responses.
- However, two-player Nash will not be FNP-hard. To prove that, we need to show it is not **FNP-Complete**.

# A Gentle Touch on PPAD

- Two-player Nash is not FNP-complete.

- Completeness is build on the notion of **reduction**:

- **If we want to solve P, it suffices to solve Q**: to solve instance I of problem P, we can first find a solution of X of A(I), which is of Q, and then use B to find a solution of B(X) of I.

- **Theorem: Two-player Nash is not FNP-complete**

- We can proof that if two-player Nash is FNP-hard then  $NP=coNP$  (verifying “No” instance in P-time).

- ✦ Proof by showing that if true, we can find a certificate of NO instances for SAT problems.
- ✦ SAT problem: find a and b such that “a AND NOT b” is satisfied. SAT is known to be NP-complete.
- ✦ Since most theorists think that  $NP \neq coNP$ , this is strong evidence that 2-Player Nash is not FNP-

- We need a new class that has complete problems for the search tasks: **Polynomial Parity Argument Directed (PPAD)**

- **Theorem: Two-player Nash is PPAD-complete.**

**Definition 4.2.2** We say that P reduces to Q (denoted as  $P \leq_p Q$ ) if there exist polynomial-time algorithms A and B such that

1. A maps instances of P to instances of Q,
2. If I is a YES instance of P than A(I) is a YES instance of Q, and
3. If X is a witness for A(I), then B(x) is a witness of I (if I is a YES instance) or NO (if I is a NO instance).

# Complexity of Computing Nash Equilibrium in Stochastic Games

- Solving Nash Equilibrium in normal-form games is PPAD-hard; we expect solving Nash in stochastic games can only be harder ! But it is not.

- **Theorem: Computing Markov Perfect Equilibrium in N-Player SGs is PPAD-complete.**

**Definition 3** (Behavioral Strategy). A behavioral strategy of an agent  $i$  is  $\pi^i : \mathbb{S} \rightarrow \Delta(\mathbb{A}^i)$ , i.e.,  $\forall s \in \mathbb{S}, \pi^i(s)$  is a probability distribution on  $\mathbb{A}^i$ .

**Definition 5** (Markov Perfect Equilibrium (MPE)). A behavioral strategy profile  $\pi$  is called a Markov Perfect Equilibrium if

$$\forall s \in \mathbb{S}, i \in [n], \forall \tilde{\pi}^i \in \Delta_{A^i}^S, V^{\pi^i, \pi^{-i}}(s) \geq V^{\tilde{\pi}^i, \pi^{-i}}(s).$$

## On the Complexity of Computing Markov Perfect Equilibrium in General-Sum Stochastic Games

**Xiaotie Deng\***  
Center on Frontiers of Computing Studies  
Peking University  
xiaotie@pku.edu.cn

**Yuhao Li\***  
Center on Frontiers of Computing Studies  
Peking University  
yuhao.li.cs@pku.edu.cn

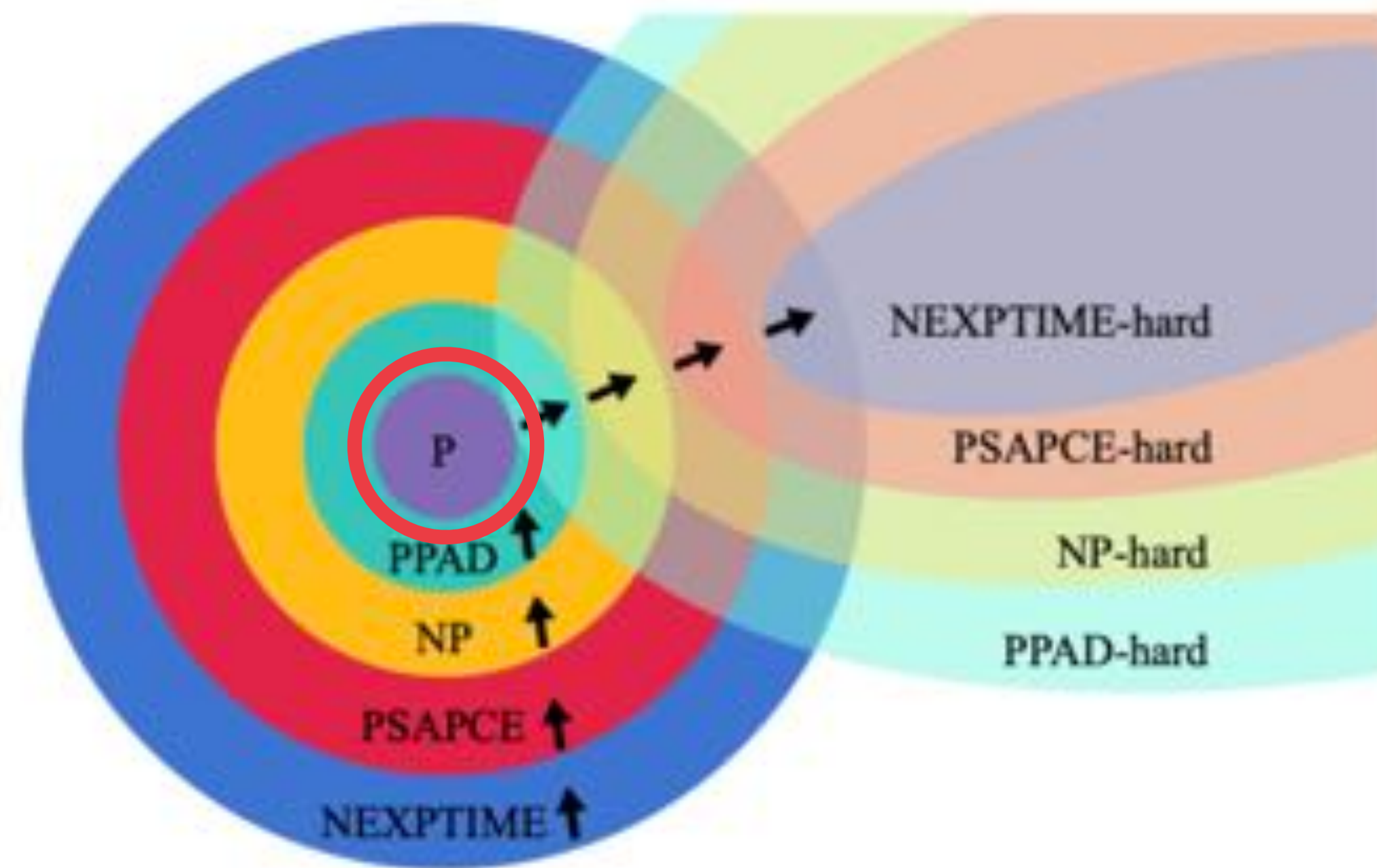
**David Henry Mguni**  
Huawei R&D UK  
davidmguni@hotmail.com

**Jun Wang**  
University College London  
jun.wang@cs.ucl.ac.uk

**Yaodong Yang**  
King's College London  
yaodong.yang@outlook.com

- Meaning computing Nash in SGs is unlikely to be NP-hard unless  $NP \neq coNP$ .
- PPAD problems can always have exp-time algorithms, can we have P-time solutions ?
  - ♦ Short answer is we don't know yet. Similar to we don't know if  $P=NP$ . But highly likely NO.

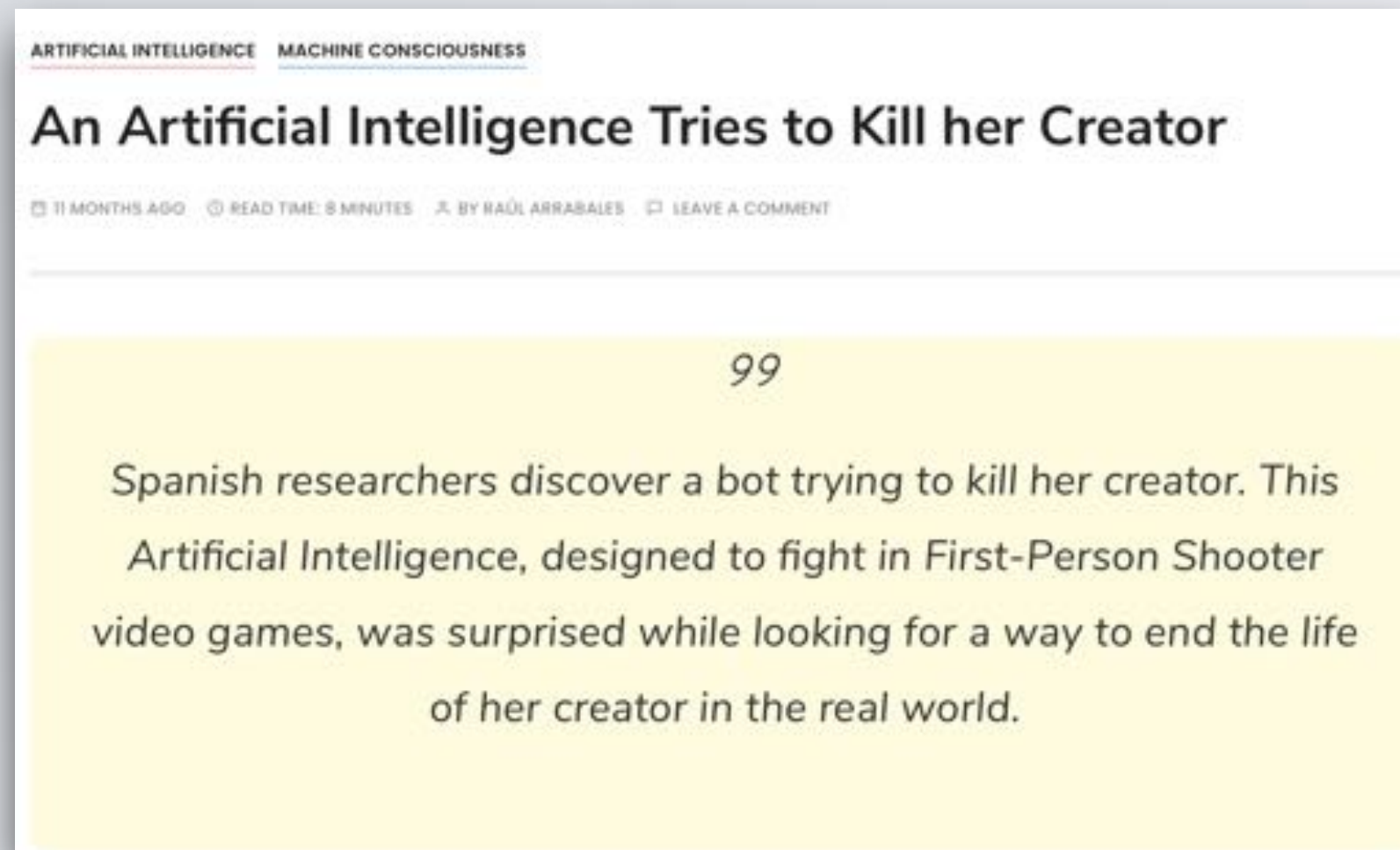
# Summary of Complexity Results



**Figure 1.5:** Landscape of different complexity classes. Relevant examples are: 1) solving NE in two-player zero-sum game is  $P$  (Neumann, 1928). 2) solving NE in two-player general-sum game is  $PPAD$ -hard (Daskalakis et al., 2009). solving NE in three-player zero-sum game is also  $PPAD$ -hard (Daskalakis and Papadimitriou, 2005). 3) checking the uniqueness of NE is  $NP$ -hard (Conitzer and Sandholm, 2002). 4) checking whether pure-strategy NE exists in stochastic game is  $PSPACE$ -hard (Conitzer and Sandholm, 2008). 5) solving Dec-POMDP is  $NEXPTIME$ -hard (Bernstein et al., 2002).



## What your Mum thinks



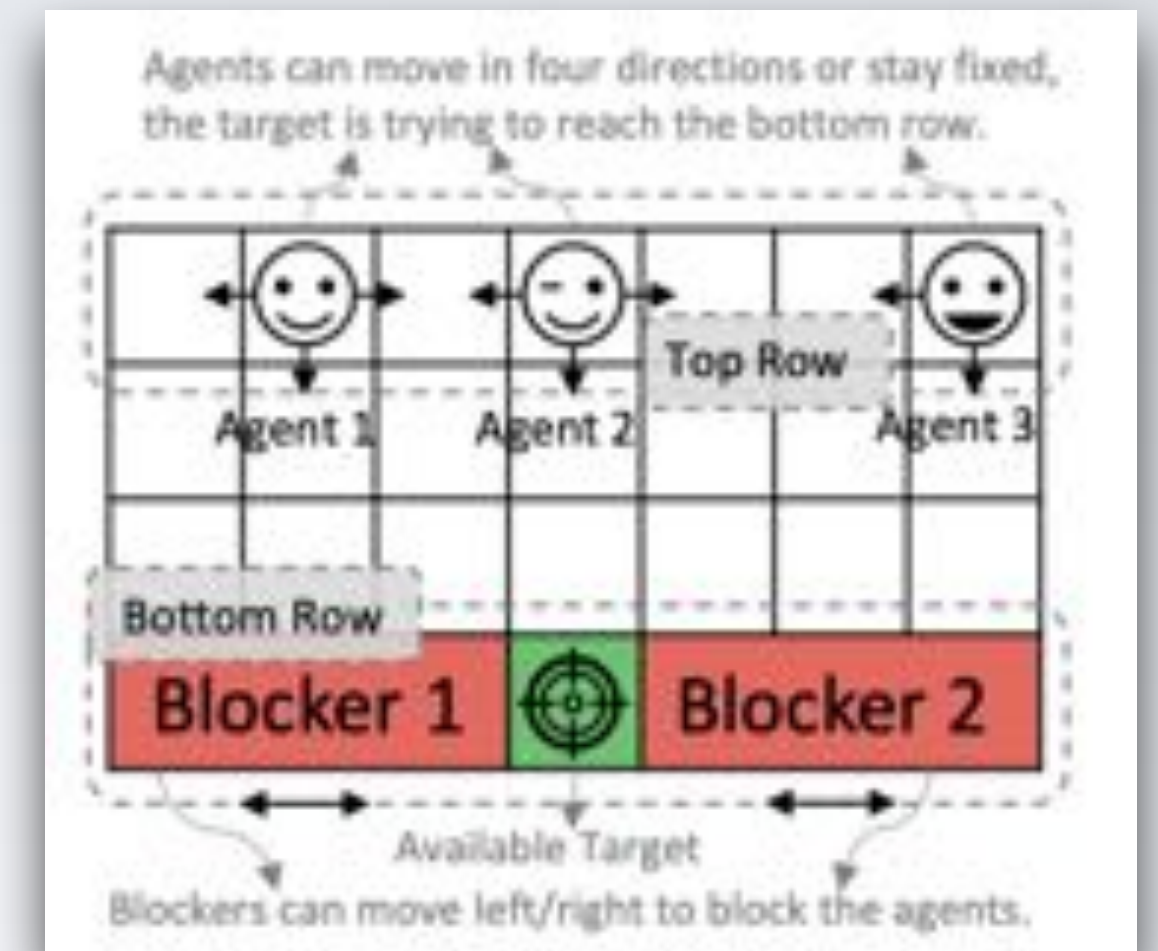
Something undescribable :)

## What you think you are doing



Multi-player general-sum games with high-dimensional continuous state-action space

## What you are actually doing



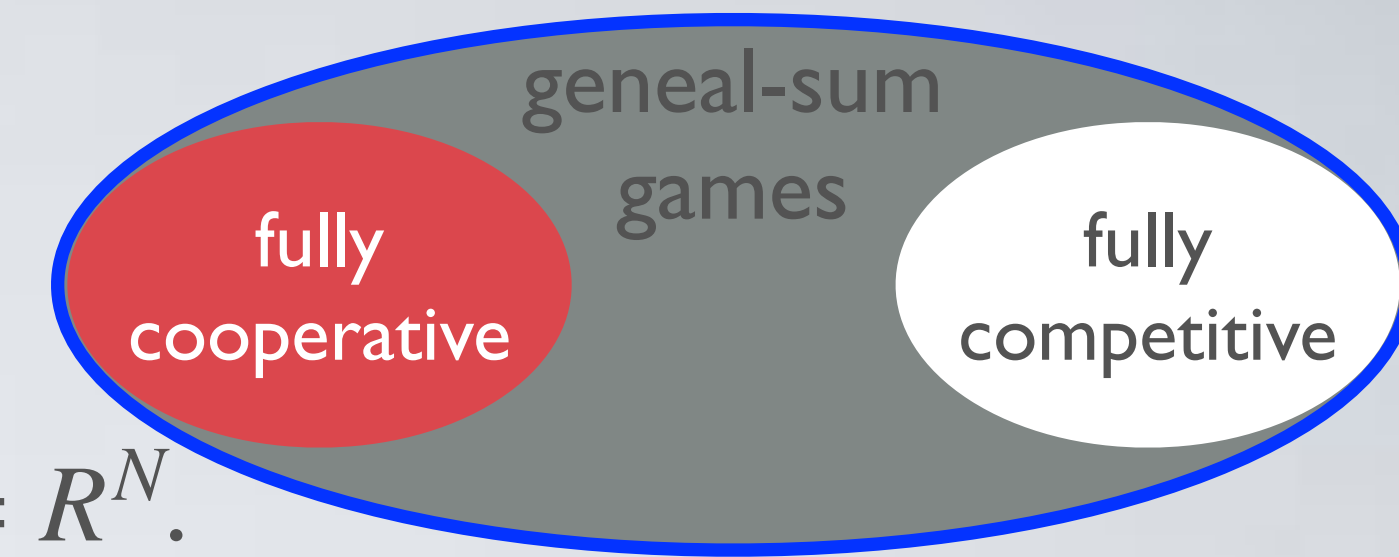
Two-player discrete-action game in a grid world.

**Training a Population of Reinforcement Learners is Hard !**

# Contents

- **Formulation & Challenges of Training A Population of RL Agents**
- **Training A Population of RL Agents on Fully-Cooperative Games**
- Training A Population of RL Agents on Zero-Sum Games
- Some System Level Thinkings
- Conclusions

# Fully-cooperative Games



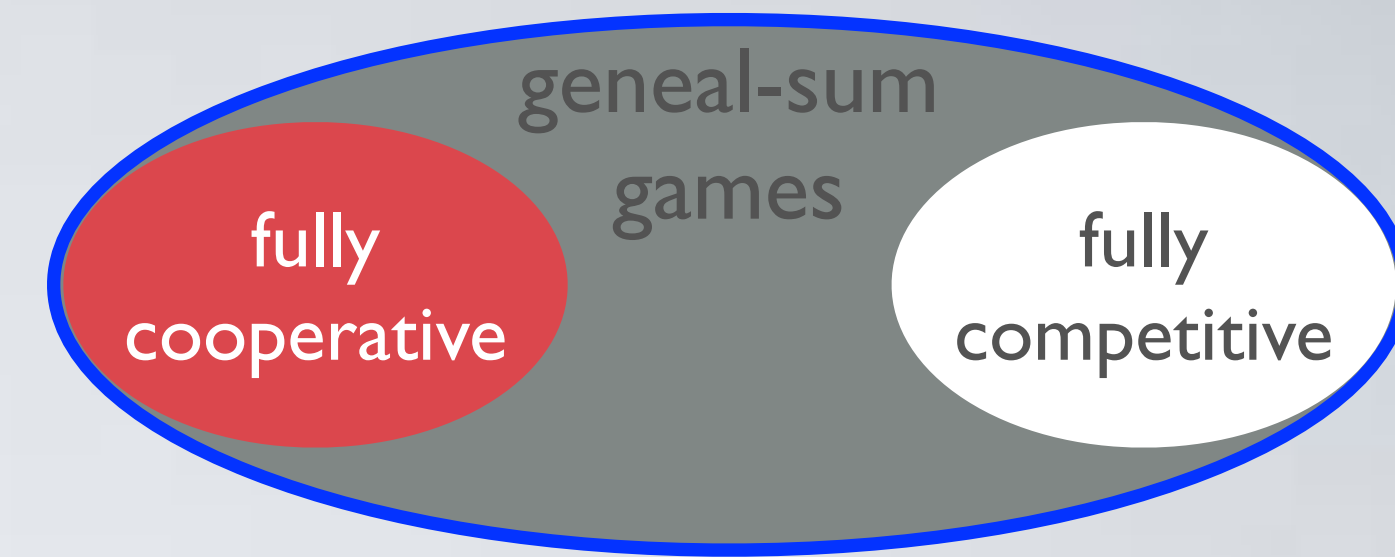
- In the fully-cooperative games, all agents share the same reward  $R^1 = R^2 = \dots = R^N$ .
- One way to solve such games are through centralised approach, learning  $Q^i(s, a^1, a^2, \dots, a^n)$  via TD error.
- The beauty of applying MARL methods lies in its **decentralisable**; as a result, **centralised training with decentralised execution** is used.
- For value-based CTDE methods, a common assumption is to assume **individual-global-max condition** holds.

$$\arg \max_{\mathbf{a} \in \mathcal{A}} Q_{tot}(s, \mathbf{a}) = \left( \arg \max_{a_1 \in \mathcal{A}} Q_1(s_1, a_1), \dots, \arg \max_{a_n \in \mathcal{A}} Q_n(s_n, a_n) \right)$$

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(s, \mathbf{a}, r, s') \in D} \left[ \left( r + \gamma V(s'; \boldsymbol{\theta}^-) - Q_{tot}(s, \mathbf{a}; \boldsymbol{\theta}) \right)^2 \right]$$

- To implement the above assumption, methods such as VDN, QMIX, Q-DPP, QPLEX are used.
  - ♦ **VDN**:  $Q_{tot}(s, \mathbf{a}) = \sum_{i=1}^n Q_i(s^i, a^i; \theta^i)$ , **QMIX**:  $\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A$ , **Q-DPP**:  $Q^\pi(\mathbf{o}, \mathbf{a}) := \log \det \left( \mathcal{L}_{Y=\{(o_1, a_1), \dots, (o_N, a_N)\} \in \mathcal{B}(\mathbf{o}^t)} \right)$
  - ♦ **QPLEX**:  $V_{tot}(s) = \sum_{i=1}^n V_i(s_i)$  and  $A_{tot}(s, \mathbf{a}) = \sum_{i=1}^n \lambda_i(s, \mathbf{a}) A_i(s_i, a_i)$ , where  $\lambda_i(s, \mathbf{a}) > 0$

# IGM Condition Can Fail in Cooperative Games

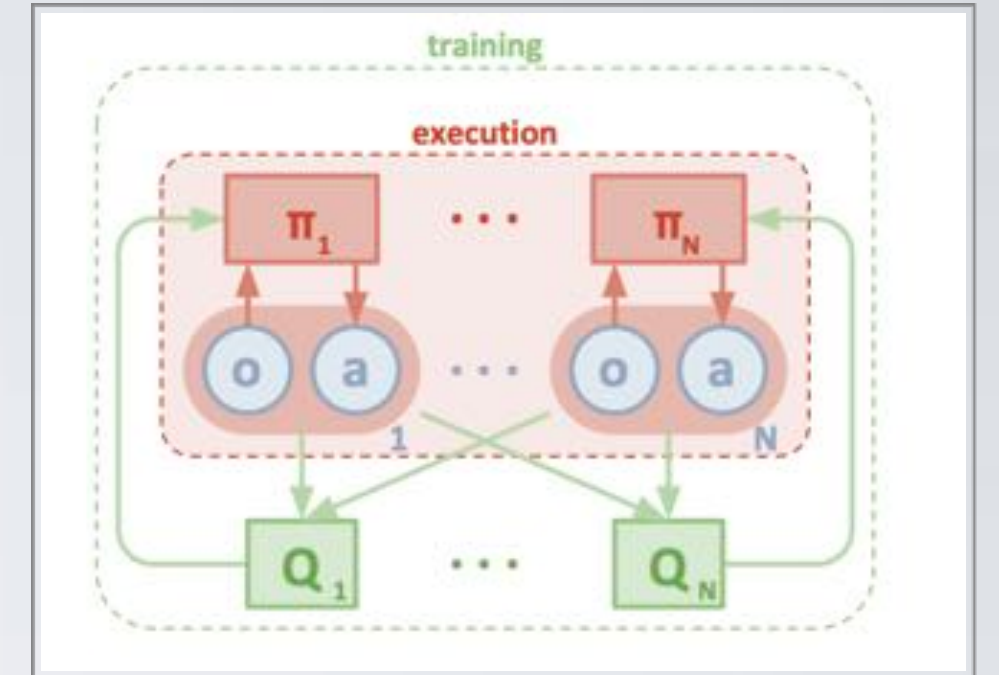


- The **individual-global-max condition** simply **fail** in some cooperative tasks.
  - ♦ Consider two agents (1,2) that take actions  $a^1, a^2 \in \{-10, -9, \dots, 9, 10\}$ . They receive the reward  $r(a^1, a^2) = a^1 \cdot a^2$ . Every agent  $i$  takes its action  $a^i$  from its policy  $\mu^i$ , which is an action.
  - ♦ Suppose that  $(\mu^1, \mu^2) = (1, -1)$ . The value function is thus  $V(s) = E[r(a^1, a^2)] = \mu^1 \mu^2 = -1$ .
  - ♦ The advantage of agent 1 is  $A_1(s, a^1) = Q_1(s, a^1) - V(s) = E[r(a^1, a^2) | a^1] - \mu^1 \mu^2 = -a^1 + 1$ ,
  - ♦ The advantage of agent 2 is  $A_2(s, a^2) = a^2 + 1$ , and the joint advantage is  $A(s, a) = Q(s, a) - V(s) = r(a^1, a^2) + 1 = a^1 \cdot a^2 + 1$ .
  - ♦ We can see that the maximising action for agent 1 is  $a_{\max}^1 = -10$ , and for agent 2 it is  $a_{\max}^2 = 10$ .
  - ♦ However, the global maximising action is either  $a_{\max} = (-10, -10)$ , or,  $a_{\max} = (10, 10)$ . Hence, we have  $(a_{\max}^1, a_{\max}^2) \neq a_{\max}$ , meaning that the advantage-based IGM does not hold in this game.

# Multi-Agent Policy Gradient Methods

- Apart from IGM condition in value based methods. There are multi-agent PG methods that execute CTDE.

- Fully Decentralised PG: 
$$\mathbf{g}_D^i = \sum_{t=0}^{\infty} \gamma^t \hat{Q}^i (s_t, \mathbf{a}_t^i) \nabla_{\theta^i} \log \pi_{\theta^i} (a_t^i | s_t)$$
- CTDE PG: 
$$\mathbf{g}_C^i = \sum_{t=0}^{\infty} \gamma^t \hat{Q} (s_t, \mathbf{a}_t^{-i}, a_t^i) \nabla_{\theta^i} \log \pi_{\theta^i} (a_t^i | s_t)$$



- Joint critic will have **credit assignment issue**: not sure where the reward increase come from
- Multi-agent CTDE methods can have **huge variance**.

**Theorem 1.** *The CTDE and DT estimators of MAPG satisfy*

$$\mathbf{Var}_{s_{0:\infty} \sim d_{\theta}^{0:\infty}, \mathbf{a}_{0:\infty} \sim \pi_{\theta}} [\mathbf{g}_C^i] - \mathbf{Var}_{s_{0:\infty} \sim d_{\theta}^{0:\infty}, \mathbf{a}_{0:\infty} \sim \pi_{\theta}} [\mathbf{g}_D^i] \leq \frac{B_i^2}{1 - \gamma^2} \sum_{j \neq i} \epsilon_j^2 \leq (n - 1) \frac{(\epsilon B_i)^2}{1 - \gamma^2}$$

it is only linear in  $n$  !

where  $B_i = \sup_{s, \mathbf{a}} \|\nabla_{\theta^i} \log \pi_{\theta^i} (a^i | s)\|$ ,  $\epsilon_i = \sup_{s, \mathbf{a}^{-i}, a^i} |A_{\theta^i}(s, \mathbf{a}^{-i}, a^i)|$ , and  $\epsilon = \max_i \epsilon_i$ .

- The **more agents (n)**, the **more explorations (epsilon)** from others, the larger the gradient of CTDE policy gradients.

# Multi-Agent Policy Gradient Methods

- One can apply the baseline trick to reduce the variance for MAPG estimation

- MAPG baseliem trick:  $g_C^i(b) = \sum_{t=0}^{\infty} \gamma^t \left[ \hat{Q}(s_t, a_t^{-i}, a_t^i) - b(s_t, a_t^{-i}) \right] \nabla_{\theta^i} \log \pi_{\theta}^i(a_t^i | s_t)$

- COMA is  $b(s, \mathbf{a}^{-i}) = \hat{Q}^{-i}(s, \mathbf{a}^{-i})$ , but still large variance

**Theorem 2.** The COMA and DT estimators of MAPG satisfy

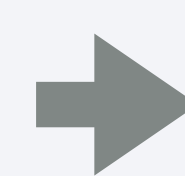
$$\mathbf{Var}_{s_{0:\infty} \sim d_{\theta}^{0:\infty}, \mathbf{a}_{0:\infty} \sim \pi_{\theta}} [\mathbf{g}_{COMA}^i] - \mathbf{Var}_{s_{0:\infty} \sim d_{\theta}^{0:\infty}, \mathbf{a}_{0:\infty} \sim \pi_{\theta}} [\mathbf{g}_D^i] \leq \frac{(\epsilon_i B_i)^2}{1 - \gamma^2}$$

- The variance of MAPG can be decomposed into

$$\begin{aligned} \mathbf{Var}_{s_t \sim d_{\theta}^t, \mathbf{a}_t \sim \pi_{\theta}} [\mathbf{g}_{C,t}^i(b)] &= \mathbf{Var}_{s_t \sim d_{\theta}^t} [\mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}} [\mathbf{g}_{C,t}^i(b)]] + \mathbb{E}_{s_t \sim d_{\theta}^t} [\mathbf{Var}_{\mathbf{a}_t \sim \pi_{\theta}} [\mathbf{g}_{C,t}^i(b)]] \quad (6) \\ &= \mathbf{Var}_{s_t \sim d_{\theta}^t} [\mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}} [\mathbf{g}_{C,t}^i(b)]] + \mathbb{E}_{s_t \sim d_{\theta}^t} [\mathbf{Var}_{\mathbf{a}_t^{-i} \sim \pi_{\theta}^{-i}} [\mathbb{E}_{a_t^i \sim \pi_{\theta}^i} [\mathbf{g}_{C,t}^i(b)]] + \mathbb{E}_{\mathbf{a}_t^{-i} \sim \pi_{\theta}^{-i}} [\mathbf{Var}_{a_t^i \sim \pi_{\theta}^i} [\mathbf{g}_{C,t}^i(b)]]] \\ &= \underbrace{\mathbf{Var}_{s_t \sim d_{\theta}^t} [\mathbb{E}_{\mathbf{a}_t \sim \pi_{\theta}} [\mathbf{g}_{C,t}^i(b)]]}_{\text{Variance from state}} + \underbrace{\mathbb{E}_{s_t \sim d_{\theta}^t} [\mathbf{Var}_{\mathbf{a}_t^{-i} \sim \pi_{\theta}^{-i}} [\mathbb{E}_{a_t^i \sim \pi_{\theta}^i} [\mathbf{g}_{C,t}^i(b)]]]}_{\text{Variance from other agents' actions}} + \underbrace{\mathbb{E}_{s_t \sim d_{\theta}^t, \mathbf{a}_t^{-i} \sim \pi_{\theta}^{-i}} [\mathbf{Var}_{a_t^i \sim \pi_{\theta}^i} [\mathbf{g}_{C,t}^i(b)]]}_{\text{Variance from agent } i\text{'s action}}. \end{aligned}$$

- The optimal baseline that gives the minimal variance is

$$\min_{b(s, \mathbf{a}^{-i})} \mathbf{Var}_{a^i \sim \pi_{\theta}^i} \left[ \left( \hat{Q}(s, \mathbf{a}^{-i}, a^i) - b(s, \mathbf{a}^{-i}) \right) \nabla_{\theta^i} \log \pi_{\theta}^i(a^i | s) \right]$$



$$b^{\text{optimal}}(s, \mathbf{a}^{-i}) = \frac{\mathbb{E}_{a^i \sim \pi_{\theta}^i} \left[ \hat{Q}(s, \mathbf{a}^{-i}, a^i) \left\| \nabla_{\theta^i} \log \pi_{\theta}^i(a^i | s) \right\|^2 \right]}{\mathbb{E}_{a^i \sim \pi_{\theta}^i} \left[ \left\| \nabla_{\theta^i} \log \pi_{\theta}^i(a^i | s) \right\|^2 \right]}$$

- Apply the baseline by  $\hat{Q}(s, \mathbf{a}^{-i}, a^i) - b^{\text{optimal}}(s, \mathbf{a}^{-i})$ , and then follow PG.

# Optimal Baseline has Excellent Performance

## Settling the Variance of Multi-Agent Policy Gradients

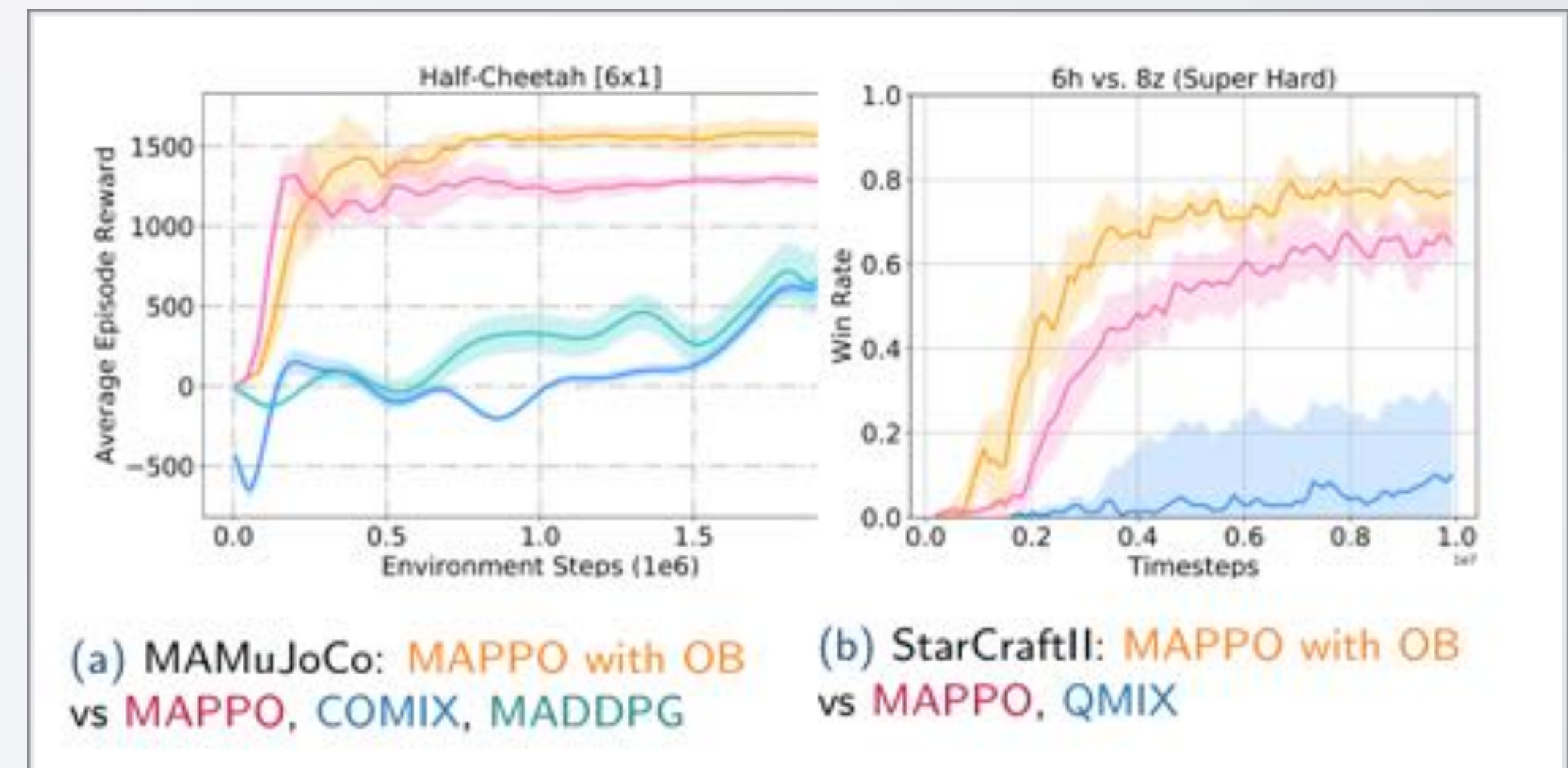
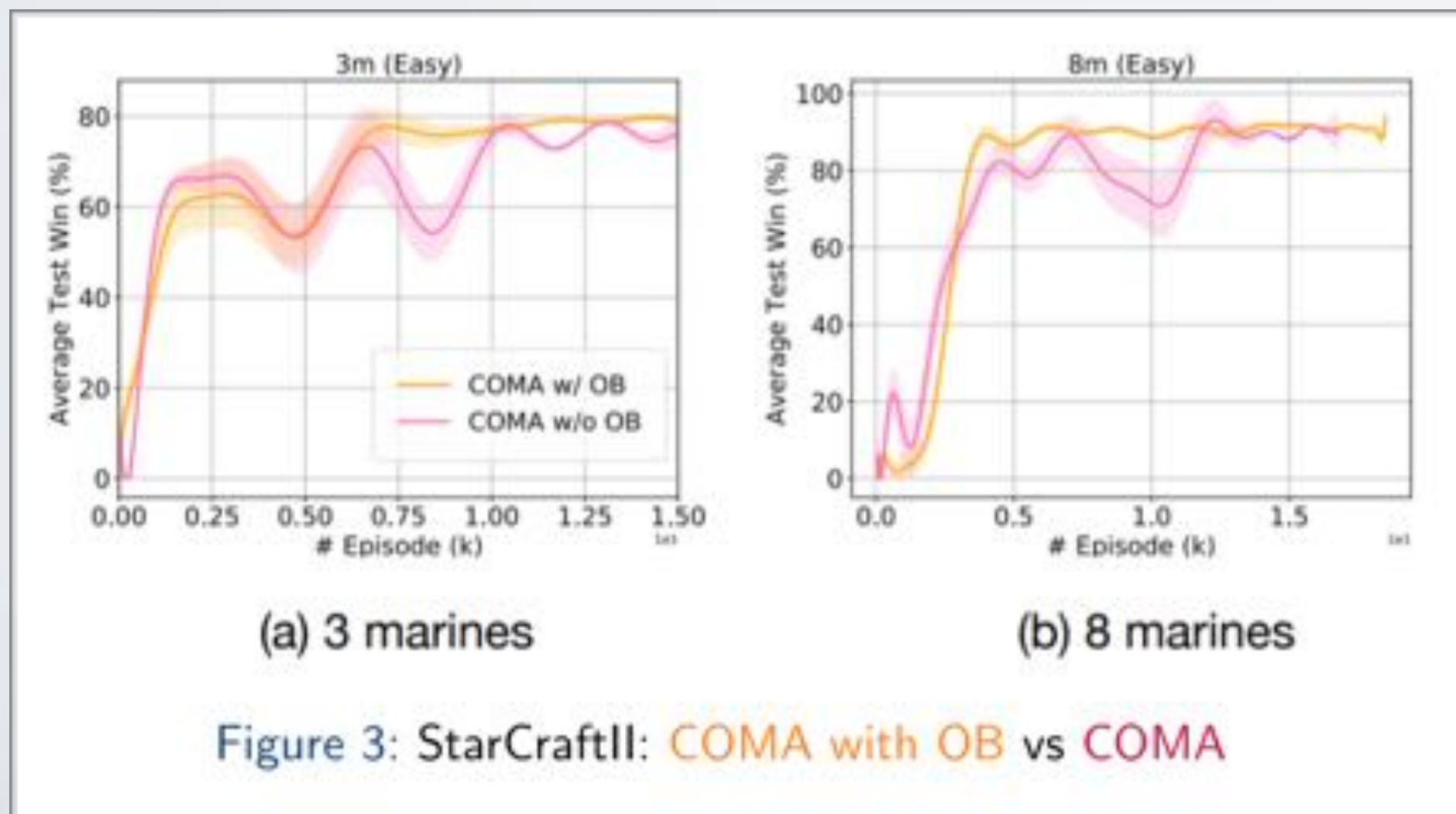
Jakub Grudzien Kuba<sup>\*1,2</sup>, Muning Wen<sup>\*3</sup>, Yaodong Yang<sup>1,4</sup>, Linghui Meng<sup>5</sup>,

Shangding Gu<sup>5</sup>, Haifeng Zhang<sup>5</sup>, David Henry Mguni<sup>2</sup>, Jun Wang<sup>6</sup>

- OB is a plug-and-play trick that can be used in any existing MAPG methods such as MAPPO, COMA

$a^i$	$\psi_{\theta}^i(a^i)$	$\pi_{\theta}^i(a^i)$	$x_{\psi_{\theta}^i}^i(a^i)$	$\hat{Q}(a^{-i}, a^i)$	$\hat{A}^i(a^{-i}, a^i)$	$\hat{X}^i(a^{-i}, a^i)$	Method	Variance
1	$\log 8$	0.8	0.14	2	-9.7	-41.71	MAPG	<b>1321</b>
2	0	0.1	0.43	1	-10.7	-42.71	COMA	<b>1015</b>
3	0	0.1	0.43	100	88.3	56.29	OB	<b>673</b>

Figure 2: Toy Example



# Quick Summary for MARL in Cooperative Games

- What we have discussed so far:
  - ♦ **Fully-centralised approach:** learning  $Q^i(s, a^1, a^2, \dots, a^n)$  is not scalable for population of agents.
  - ♦ **Value-based methods:** require IGM condition, which may not hold in some cooperative games.
  - ♦ **Policy-based methods:** incur large policy gradient estimation variance & credit assignment issue.
  - ♦ **Parameter sharing approach:**  $\pi^i = \pi^j = \pi_\theta$  can lead to exponentially worse outcomes, e.g.,

**Proposition 1.** *Let's consider a fully-cooperative game with an even number of agents  $n$ , one state, and the joint action space  $\{0, 1\}^n$ , where the reward is given by  $r(\mathbf{0}^{n/2}, \mathbf{1}^{n/2}) = r(\mathbf{1}^{n/2}, \mathbf{0}^{n/2}) = 1$ , and  $r(\mathbf{a}^{1:n}) = 0$  for all other joint actions. Let  $\mathcal{J}^*$  be the optimal joint reward, and  $\mathcal{J}_{share}^*$  be the optimal joint reward under the shared policy constraint. Then*

$$\frac{\mathcal{J}_{share}^*}{\mathcal{J}^*} = \frac{2}{2^n}.$$

[Jakub et. al. NeurIPS 2021]

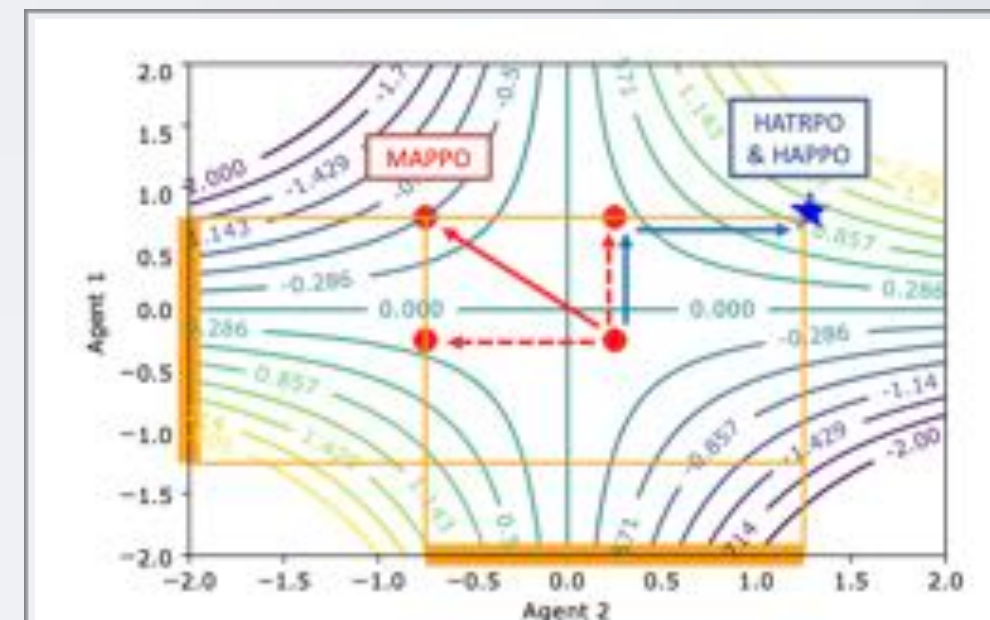


Figure 1: Example of differentiable game with  $r(a^1, a^2) = a^1 a^2$ . We initialise agents' 1-D Gaussian policies with  $\mu^1 = -0.25$ ,  $\mu^2 = 0.25$ .

- Can we have a MARL method that avoid all four above limitations ? **Yes !**



# Multi-Agent TRPO Method

- All starts from this “powerful” Lemma.
  - ◆ First, we define some new notations

## Multi-agent state-action value function

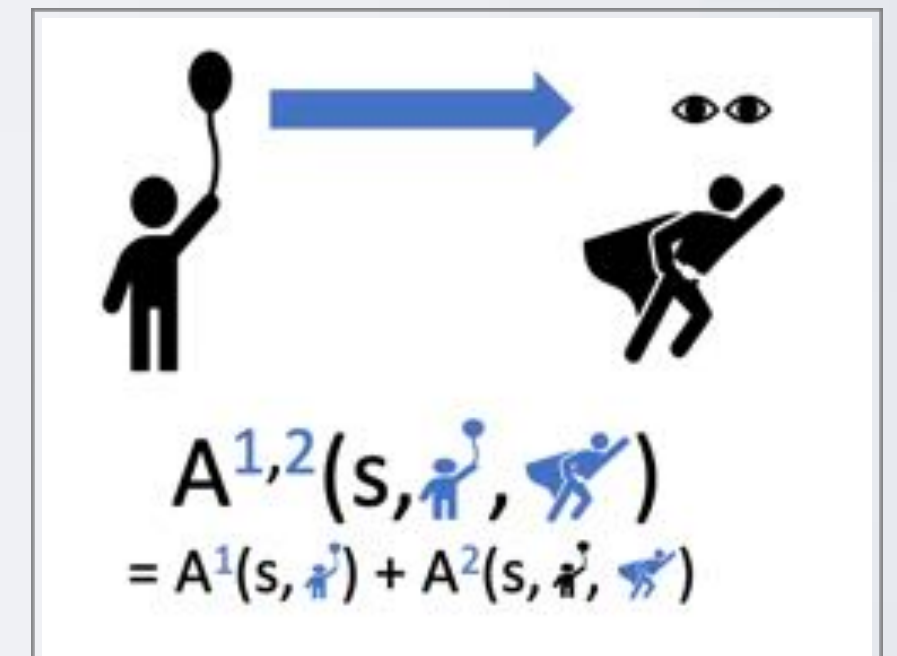
$$Q_{\pi}^{i_{1:k}}(s, \mathbf{a}^{i_{1:k}}) = \mathbb{E}_{\mathbf{a}^{-i_{1:k}} \sim \pi^{-i_{1:k}}} [Q_{\pi}(s, \mathbf{a}^{i_{1:k}}, \mathbf{a}^{-i_{1:k}})]$$

## Multi-agent advantage function

$$A_{\pi}^{i_{1:k}}(s, \mathbf{a}^{j_{1:m}}, \mathbf{a}^{i_{1:k}}) = Q_{\pi}^{j_{1:m}, i_{1:k}}(s, \mathbf{a}^{j_{1:m}}, \mathbf{a}^{i_{1:k}}) - Q_{\pi}^{j_{1:m}}(s, \mathbf{a}^{j_{1:m}})$$

- ◆ **Advantage decomposition lemma:**

$$A_{\pi}^{i_{1:m}}(s, \mathbf{a}^{i_{1:m}}) = \sum_{j=1}^m A_{\pi}^{i_j}(s, \mathbf{a}^{i_{1:j-1}}, a^{i_j})$$



- ◆ Note: this **does not need any assumptions such as IGM**, it holds naturally in any cooperative games!
- ◆ This can offer some new insights for cooperative MARL algorithm design. We can walk **away from IGM**.

# Multi-Agent TRPO Method

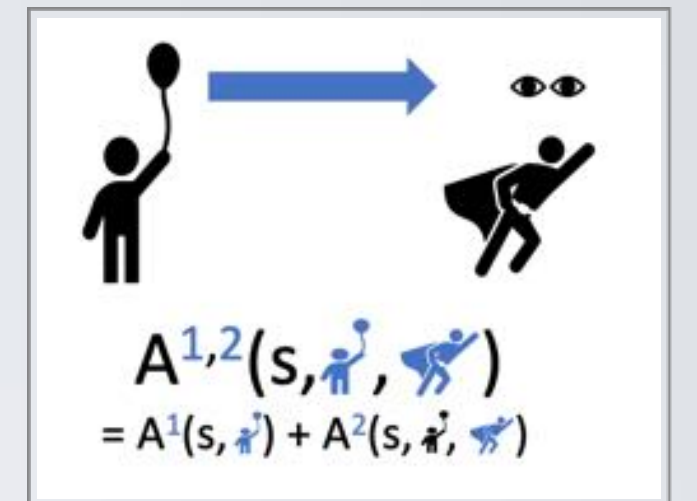
- We can design a monotonic-improvement procedure based on the Lemma

- ◆ Assuming agent order  $i_{1:n}$ , based on the Lemma, we can make the following update:

- ◆ First, select  $\bar{a}^{i_1}$  so that  $A^{i_1}(s, \bar{a}^{i_1}) > 0$

- ◆ Then, for the rest agent  $m = 2, \dots, n$ , agent  $i_m$  selects  $\bar{a}^{i_m}$  so that  $A^{i_m}(s, \bar{a}^{i_{1:m-1}}, \bar{a}^{i_m}) > 0$

- ◆ We can know that if every term is positive, then  $A_{\pi}^{i_{1:m}}(s, \mathbf{a}^{i_{1:m}})$  is improving monotonically in time



- This leads to Multi-Agent TRPO update that has monodically improving property

MARL surrogate loss

**Definition 2.** Let  $\pi$  be a joint policy,  $\bar{\pi}^{i_{1:m-1}}$  be some other joint policy of agents  $i_{1:m-1}$ , and  $\hat{\pi}^{i_m}$  be some other policy of agent  $i_m$ . Then

$$L_{\pi}^{i_{1:m}}(\bar{\pi}^{i_{1:m-1}}, \hat{\pi}^{i_m}) \triangleq \mathbb{E}_{s \sim \rho_{\pi}, \mathbf{a}^{i_{1:m-1}} \sim \bar{\pi}^{i_{1:m-1}}, \mathbf{a}^{i_m} \sim \hat{\pi}^{i_m}} [A_{\pi}^{i_m}(s, \mathbf{a}^{i_{1:m-1}}, \mathbf{a}^{i_m})].$$

MA Trust Region

**Lemma 2.** Let  $\pi$  be a joint policy. Then, for any joint policy  $\bar{\pi}$ , we have

$$\mathcal{J}(\bar{\pi}) \geq \mathcal{J}(\pi) + \sum_{m=1}^n [L_{\pi}^{i_{1:m}}(\bar{\pi}^{i_{1:m-1}}, \bar{\pi}^{i_m}) - CD_{KL}^{max}(\pi^{i_m}, \bar{\pi}^{i_m})].$$

# Multi-Agent TRPO Method

## Algorithm 1 Multi-Agent Policy Iteration with Monotonic Improvement Guarantee

- 1: Initialise the joint policy  $\pi_0 = (\pi_0^1, \dots, \pi_0^n)$ .
- 2: **for**  $k = 0, 1, \dots$  **do**
- 3:   Compute the advantage function  $A_{\pi_k}(s, \mathbf{a})$  for all state-(joint)action pairs  $(s, \mathbf{a})$ .
- 4:   Compute  $\epsilon = \max_{s, \mathbf{a}} |A_{\pi_k}(s, \mathbf{a})|$  and  $C = \frac{4\gamma\epsilon}{(1-\gamma)^2}$ .
- 5:   Draw a permutation  $i_{1:n}$  of agents at random.
- 6:   **for**  $m = 1 : n$  **do**
- 7:     Make an update  $\pi_{k+1}^{i_m} = \arg \max_{\pi^{i_m}} \left[ L_{\pi_k}^{i_{1:m}} \left( \pi_{k+1}^{i_{1:m-1}}, \pi^{i_m} \right) - CD_{\text{KL}}^{\max}(\pi_k^{i_m}, \pi^{i_m}) \right]$ .
- 8:   **end for**
- 9: **end for**

Monotonic improvement  
property for MARL

**Theorem 2.** A sequence  $(\pi_k)_{k=0}^{\infty}$  of joint policies updated by Algorithm 1 has the monotonic improvement property, i.e.,  $\mathcal{J}(\pi_{k+1}) \geq \mathcal{J}(\pi_k)$  for all  $k \in \mathbb{N}$ .

MA-TRPO in practice:  
using natural gradient

$$\theta_{k+1}^{i_m} = \arg \max_{\theta^{i_m}} \mathbb{E}_{s \sim \rho_{\pi_{\theta_k}}, \mathbf{a}^{i_{1:m-1}} \sim \pi_{\theta_{k+1}}^{i_{1:m-1}}, \mathbf{a}^{i_m} \sim \pi_{\theta}^{i_m}} \left[ A_{\pi_{\theta_k}}^{i_m}(s, \mathbf{a}^{i_{1:m-1}}, \mathbf{a}^{i_m}) \right],$$

subject to  $\mathbb{E}_{s \sim \rho_{\pi_{\theta_k}}} \left[ D_{\text{KL}} \left( \pi_{\theta_k}^{i_m}(\cdot|s), \pi_{\theta}^{i_m}(\cdot|s) \right) \right] \leq \delta.$

MA-PPO in practice:  
using clip objective

$$\mathbb{E}_{s \sim \rho_{\pi_{\theta_k}}, \mathbf{a} \sim \pi_{\theta_k}} \left[ \min \left( \frac{\pi_{\theta_{k+1}}^{i_m}(\mathbf{a}^i|s)}{\pi_{\theta_k}^{i_m}(\mathbf{a}^i|s)} M^{i_{1:m}}(s, \mathbf{a}), \text{clip} \left( \frac{\pi_{\theta_{k+1}}^{i_m}(\mathbf{a}^i|s)}{\pi_{\theta_k}^{i_m}(\mathbf{a}^i|s)}, 1 \pm \epsilon \right) M^{i_{1:m}}(s, \mathbf{a}) \right) \right].$$

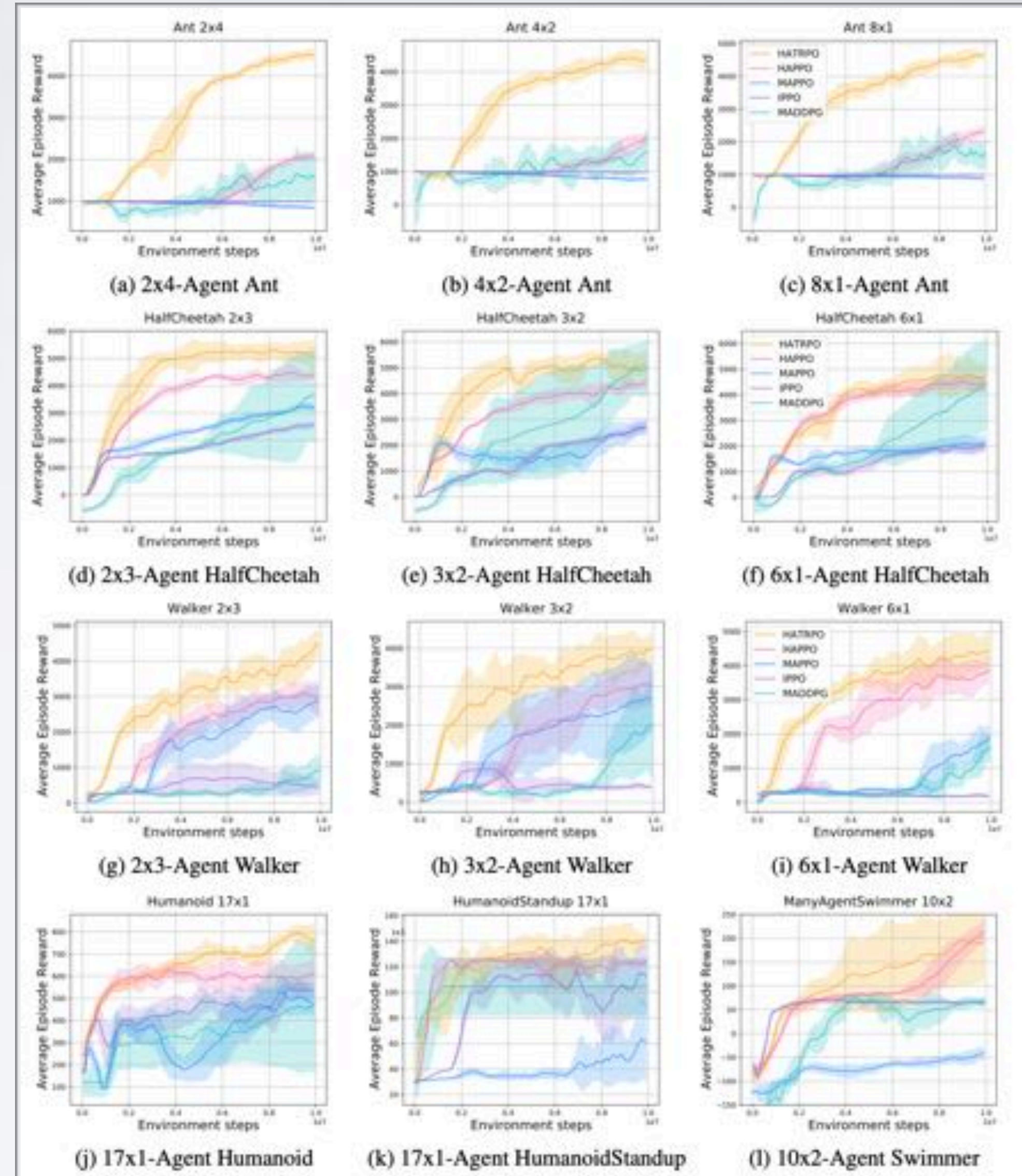
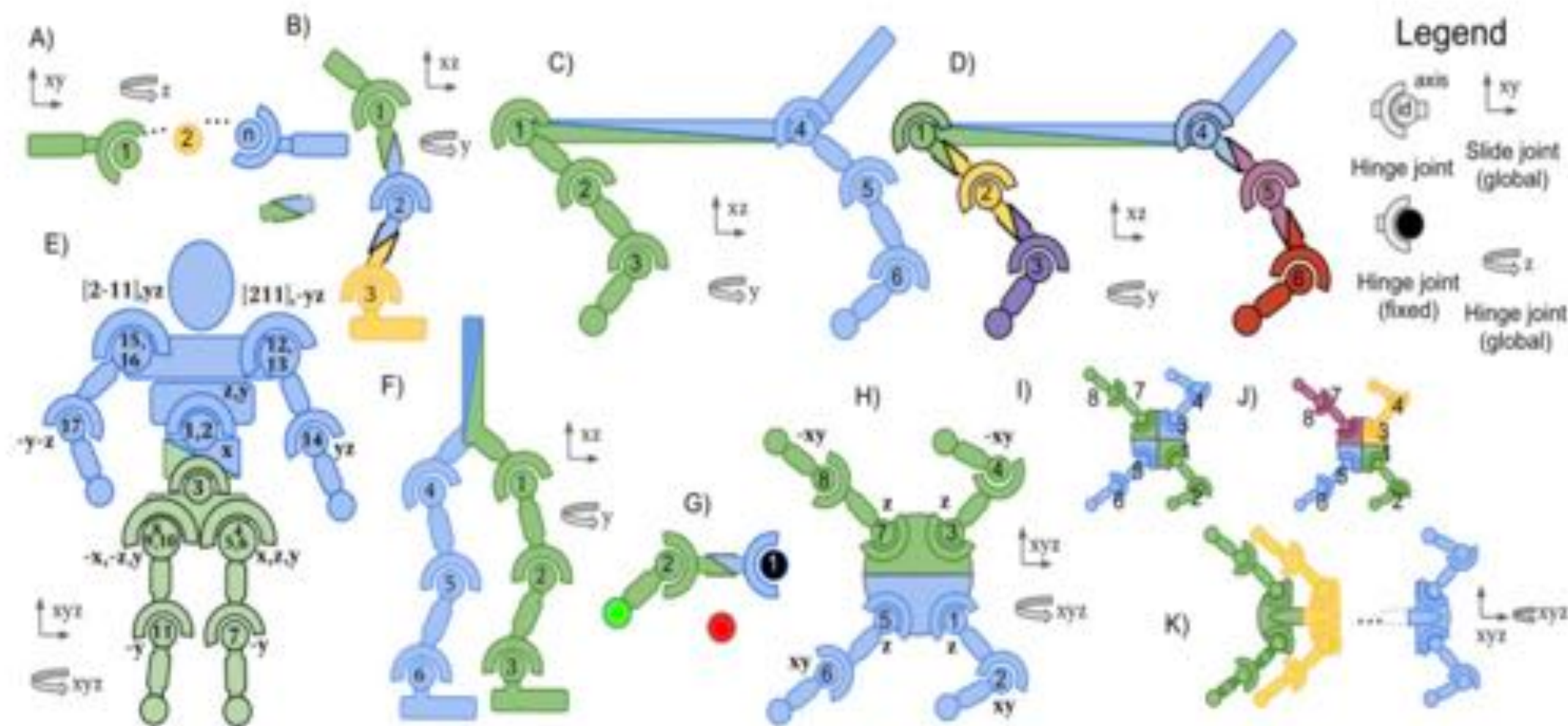
# Multi-Agent TRPO is the New SOTA in MARL

## TRUST REGION POLICY OPTIMISATION IN MULTI-AGENT REINFORCEMENT LEARNING

Jakub Grudzien Kuba<sup>1,2,\*</sup>, Ruiqing Chen<sup>3,\*</sup>, Munning Wen<sup>4</sup>, Ying Wen<sup>4</sup>, Fanglei Sun<sup>3</sup>, Jun Wang<sup>5</sup>, Yaodong Yang<sup>6,†</sup>  
<sup>1</sup>University of Oxford, <sup>2</sup>Huawei R&D UK, <sup>3</sup>ShanghaiTech University, <sup>4</sup>Shanghai Jiao Tong University <sup>5</sup>University College London, <sup>6</sup>King's College London  
 †yaodong.yang@kcl.ac.uk

### Multi-Agent Mujoco

Benchmark for Continuous Multi-Agent Robotic Control, based on OpenAI's Mujoco Gym environments.



# Multi-Agent TRPO with Safe Constraints

- Taking safe constraints into account in the MARL process

- ♦ consider additional cost functions:  $J_j^i(\pi) \triangleq \mathbb{E}_{s_0 \sim \rho^0, \mathbf{a}_{0:\infty} \sim \pi, s_{1:\infty} \sim p} \left[ \sum_{t=0}^{\infty} \gamma^t C_j^i(s_t, \mathbf{a}_t^i) \right] \leq c_j^i, \quad \forall j = 1, \dots, m^i$
- ♦ We can derive the expected cost changes by

**Lemma 2.** Let  $\pi$  and  $\bar{\pi}$  be joint policies. Let  $i \in N$  be an agent, and  $j \in \{1, \dots, m^i\}$  be an index of one of its costs. The following inequality holds

$$J_j^i(\bar{\pi}) \leq J_j^i(\pi) + L_{j,\pi}^i(\bar{\pi}^i) + v_j^i \sum_{h=1}^n D_{KL}^{\max}(\pi^h, \bar{\pi}^h), \quad \text{where } v_j^i = \frac{4\gamma \max_{s, a^i} |A_{j,\pi}^i(s, a^i)|}{(1-\gamma)^2}.$$

- ♦ This result suggests that when the changes in policies of all agents are sufficiently small, each agent  $i$  can learn a better policy by only considering its **own surrogate return and surrogate costs**.

guarantee both monotonic improvement and satisfy safety constraint

## Algorithm 1: Safe Multi-Agent Policy Iteration with Monotonic Improvement Property

- 1: Initialise a safe joint policy  $\pi_0 = (\pi_0^1, \dots, \pi_0^n)$ .
- 2: **for**  $k = 0, 1, \dots$  **do**
- 3: Compute the advantage functions  $A_{\pi_k}(s, \mathbf{a})$  and  $A_{j,\pi_k}^i(s, a^i)$ , for all state-(joint)action pairs  $(s, \mathbf{a})$ , agents  $i$ , and constraints  $j \in \{1, \dots, m^i\}$ .
- 4: Compute  $v = \frac{4\gamma \max_{s, \mathbf{a}} |A_{\pi_k}(s, \mathbf{a})|}{(1-\gamma)^2}$ , and  $v_j^i = \frac{4\gamma \max_{s, a^i} |A_{j,\pi_k}^i(s, a^i)|}{(1-\gamma)^2}, \forall i \in N, j = 1, \dots, m^i$ .
- 5: Draw a permutation  $i_{1:n}$  of agents at random.
- 6: **for**  $h = 1 : n$  **do**
- 7: Compute the radius of the KL-constraint  $\delta^{i_h}$  // see Appendix B for the setup of  $\delta^{i_h}$ .
- 8: Make an update  $\pi_{k+1}^{i_h} = \arg \max_{\pi^{i_h} \in \bar{\Pi}^{i_h}} \left[ L_{\pi_k}^{i_h}(\pi_{k+1}^{i_h}, \pi^{i_h}) - v D_{KL}^{\max}(\pi_k^{i_h}, \pi^{i_h}) \right]$ ,  
 where  $\bar{\Pi}^{i_h}$  is a subset of safe policies of agent  $i_h$ , given by  

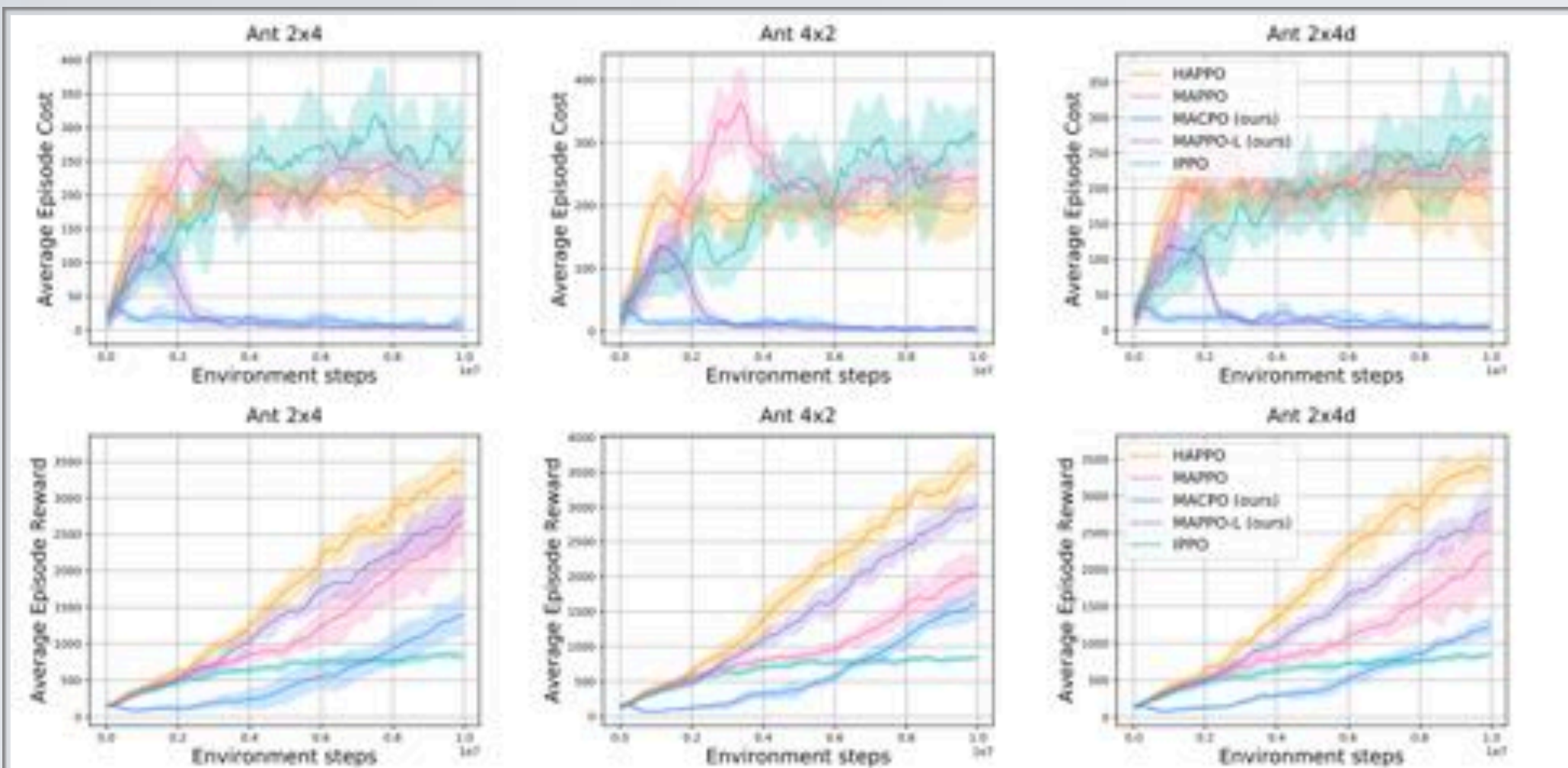
$$\bar{\Pi}^{i_h} = \left\{ \pi^{i_h} \in \Pi^{i_h} \mid D_{KL}^{\max}(\pi_k^{i_h}, \pi^{i_h}) \leq \delta^{i_h}, \text{ and } \right.$$

$$\left. J_j^{i_h}(\pi_k) + L_{j,\pi_k}^{i_h}(\pi^{i_h}) + v_j^{i_h} D_{KL}^{\max}(\pi_k^{i_h}, \pi^{i_h}) \leq c_j^{i_h} - \sum_{l=1}^{h-1} v_j^{i_l} D_{KL}^{\max}(\pi_k^{i_l}, \pi^{i_l}), \forall j = 1, \dots, m^{i_h} \right\}.$$
- 9: **end for**
- 10: **end for**

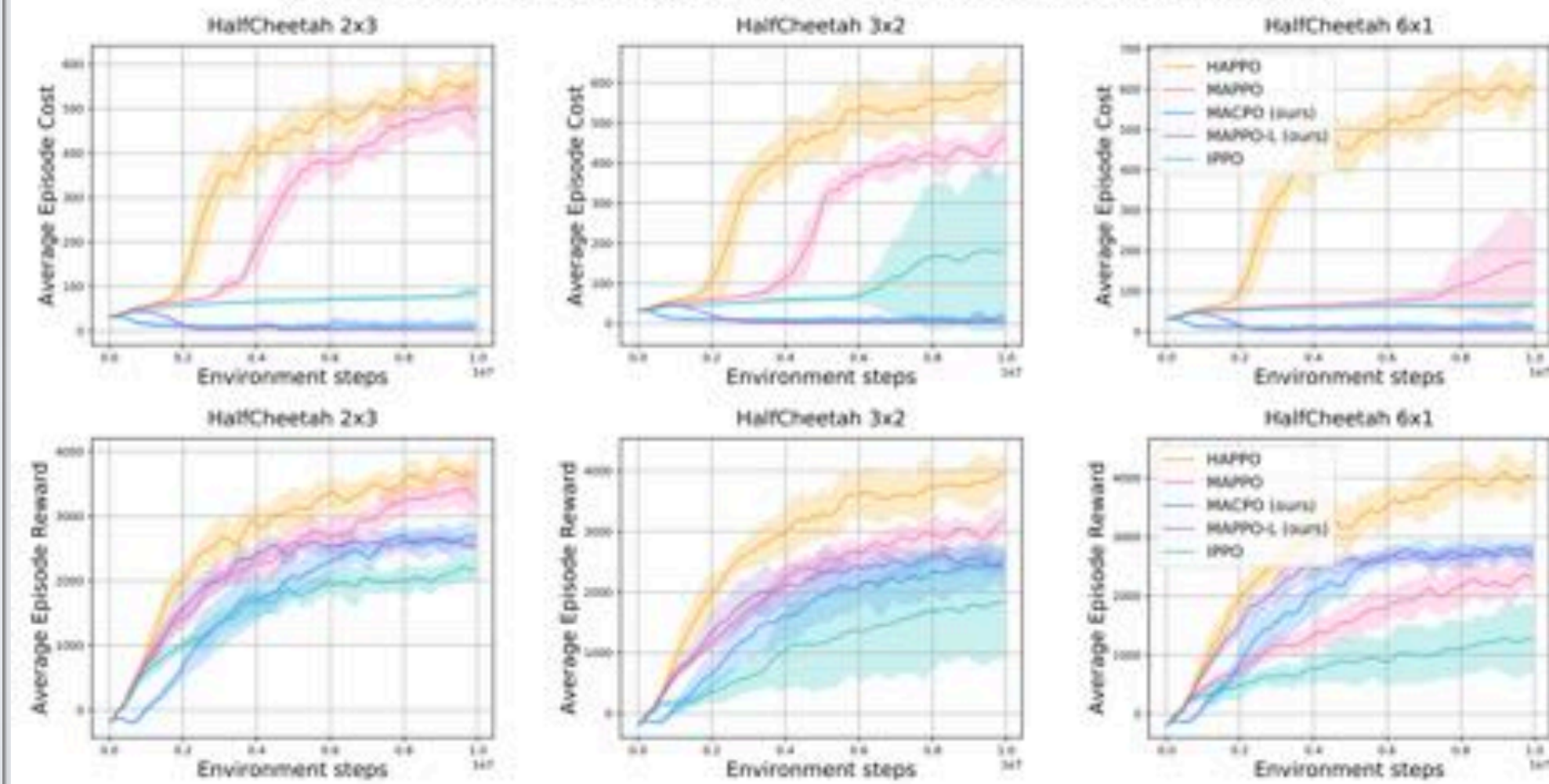
# Multi-Agent TRPO with Safe Constraints

## MULTI-AGENT CONSTRAINED POLICY OPTIMISATION

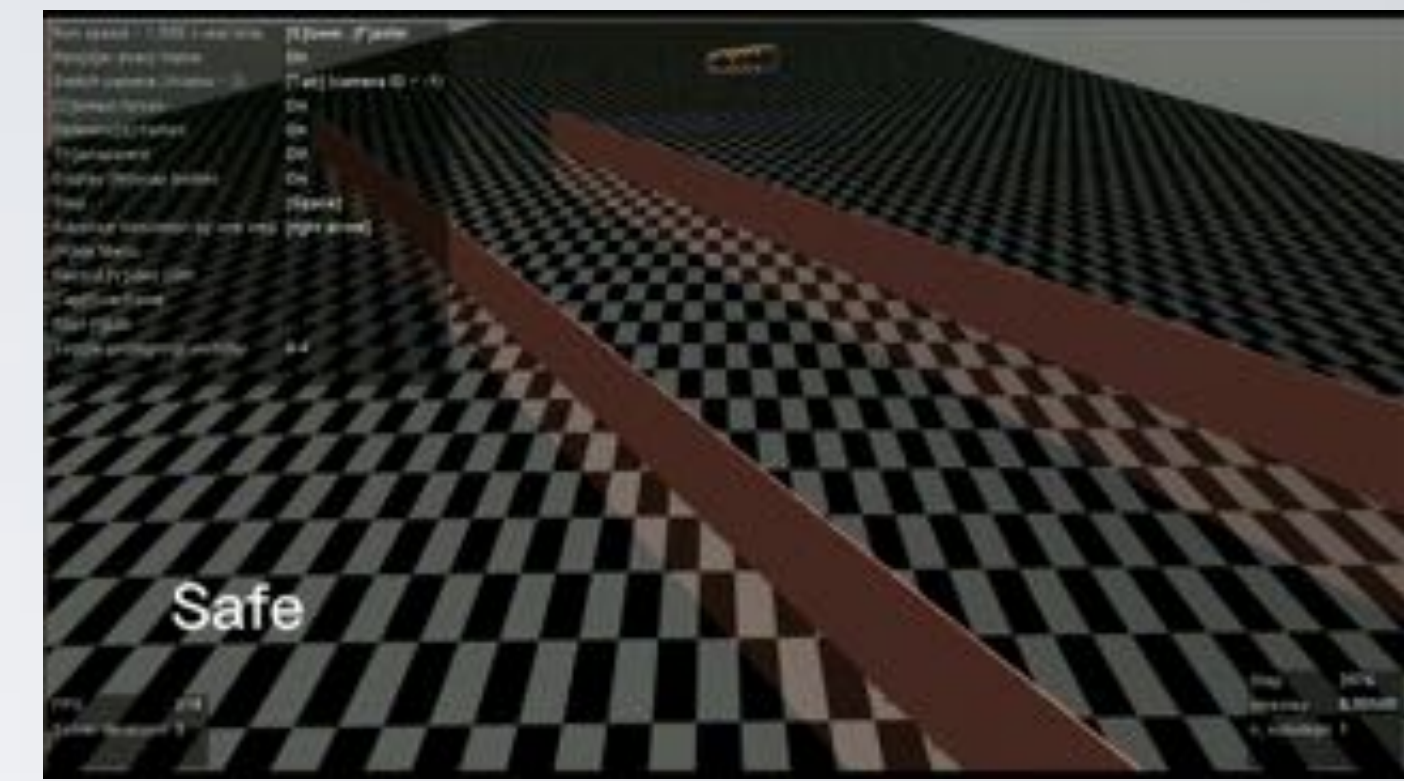
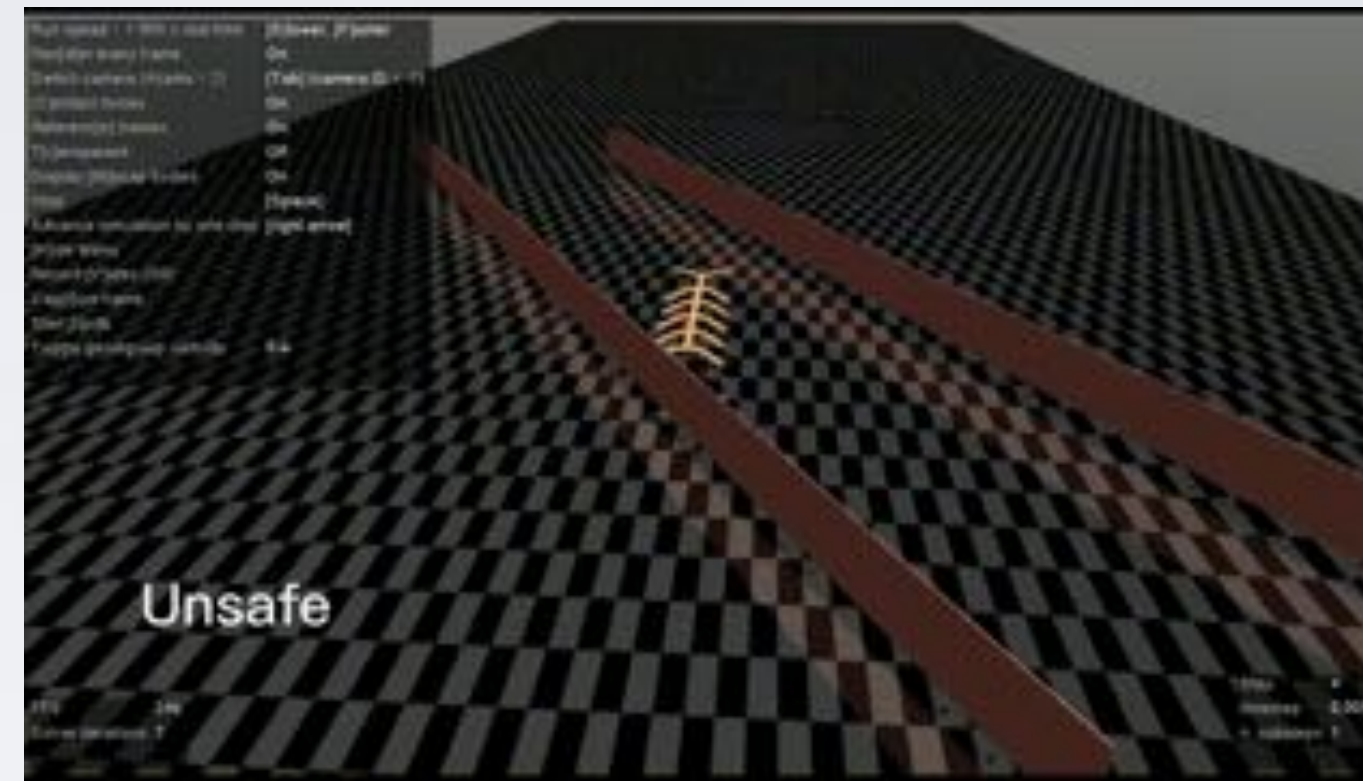
Shangding Gu<sup>1</sup>, Jakub Grudzien Kuba<sup>2,\*</sup>, Munning Wen<sup>3</sup>, Ruiqing Chen<sup>4</sup>,  
 Ziyang Wang<sup>5</sup>, Zheng Tian<sup>4,5</sup>, Jun Wang<sup>5</sup>, Alois Knoll<sup>1</sup>, Yaodong Yang<sup>6</sup>  
<sup>1</sup>Technical University of Munich <sup>2</sup>University of Oxford <sup>3</sup>Shanghai Jiao Tong University  
<sup>4</sup>ShanghaiTech University <sup>5</sup>University College London  
<sup>6</sup>King's College London  
 †yaodong.yang@kcl.ac.uk



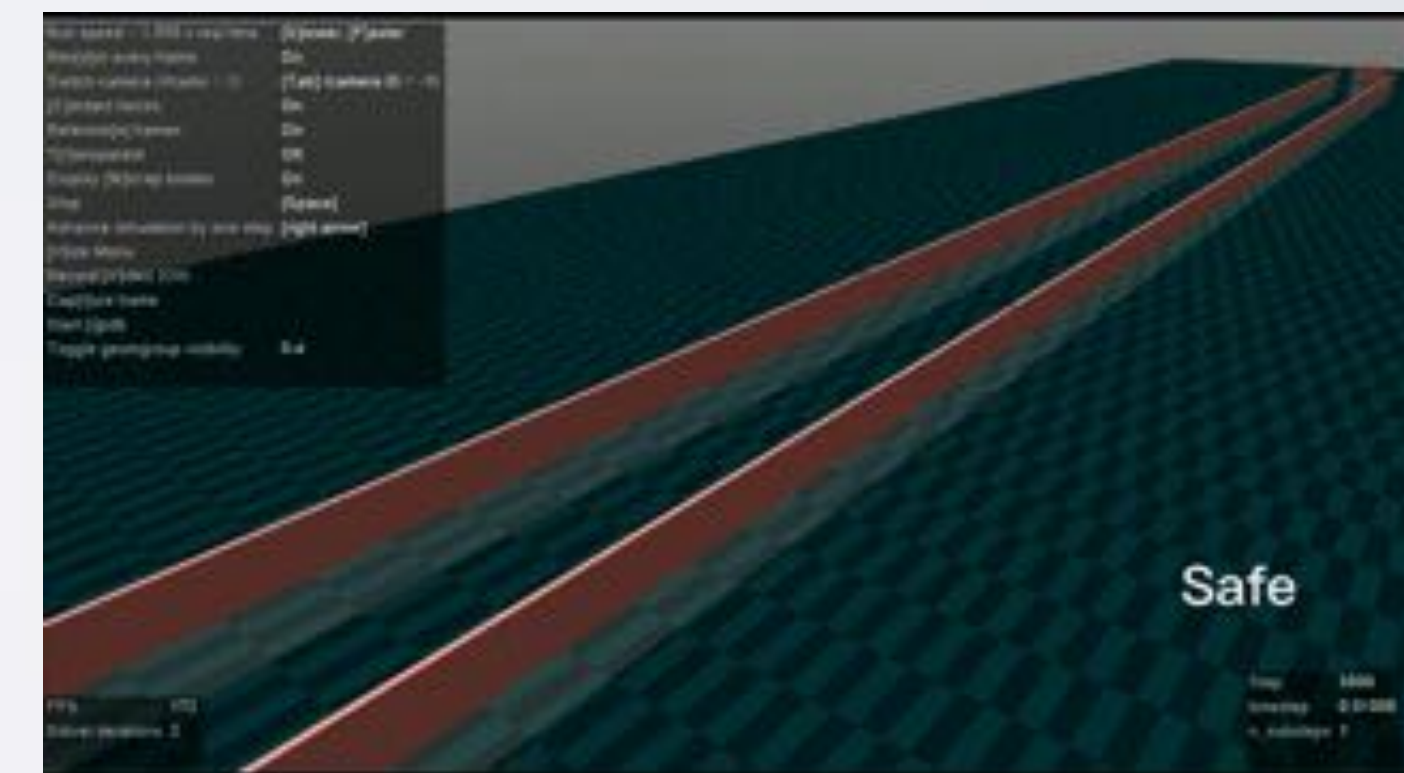
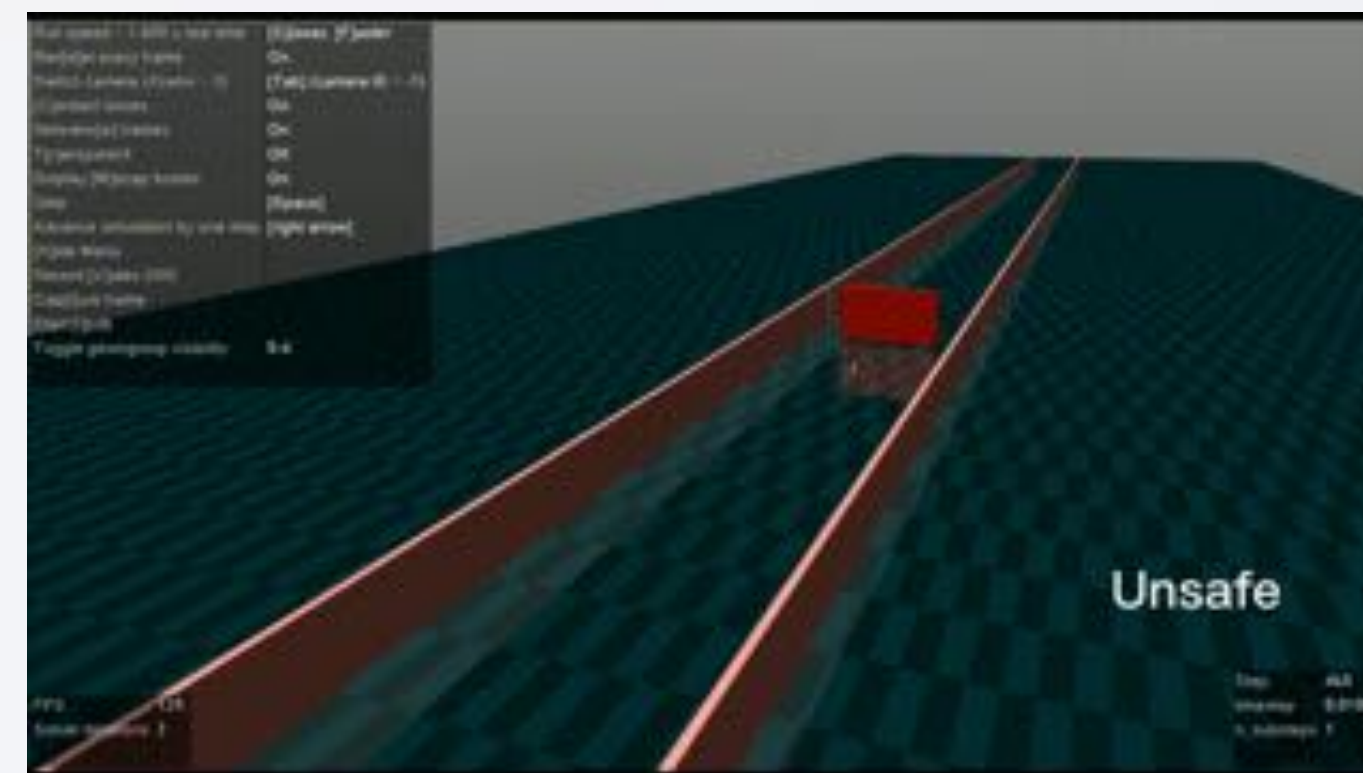
(b) Safe Ant: 2x4-Agent (left), 4x2-Agent (middle), 2x4d-Agent (right)



(c) Safe HalfCheetah: 2x3-Agent (left), 3x2-Agent (middle), 6x1-Agent (right)



ManyAgent Ant-2x3



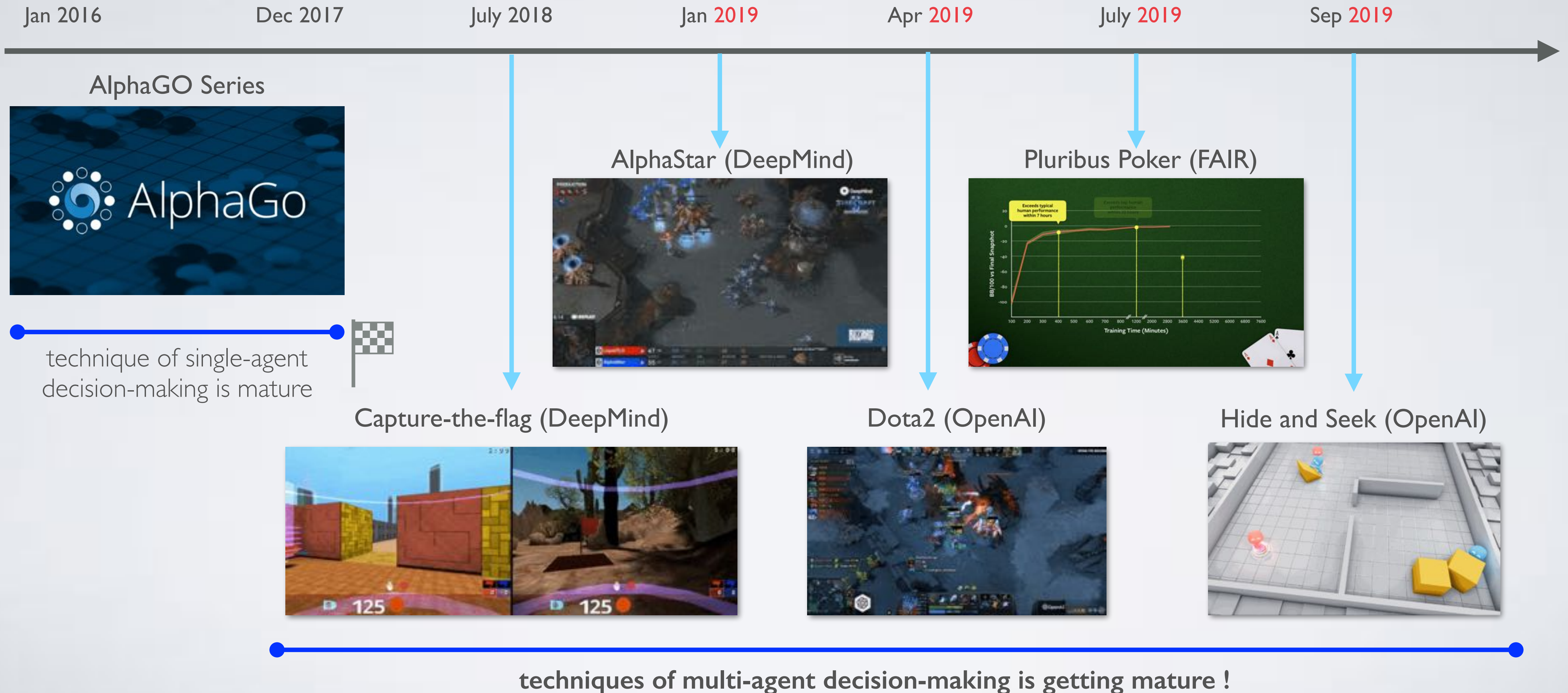
HalfCheetah-2x3 task

# Contents

- **Formulation & Challenges of Training A Population of RL Agents**
- **Training A Population of RL Agents on Fully-Cooperative Games**
- **Training A Population of RL Agents on Zero-Sum Games**
- **Some System Level Thinkings**
- **Conclusions**

# Multi-Agent Reinforcement Learning in Zero-Sum Games

Great advantages have been made in **2019!**





# A General Solver to Two-Player Zero-Sum Games

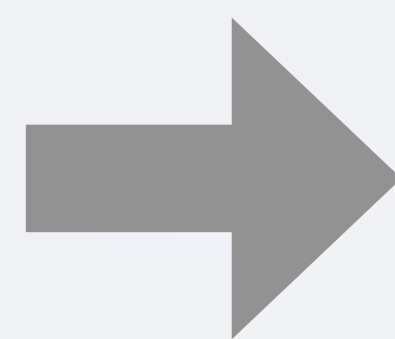
Output: the reward  $(R^1, \dots, R^N)$



Black-box multi-agent  
game engine



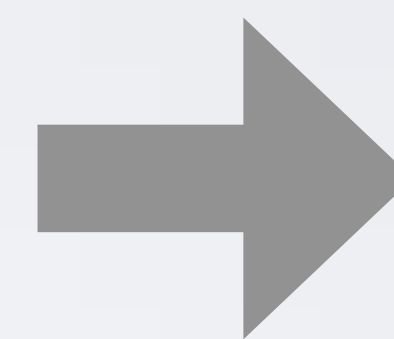
input



Our algorithm:



output



Low-exploitability  
strategy  
 $(\pi^{1,*}, \dots, \pi^{N,*})$



Input: a joint strategy  $(\pi^1, \dots, \pi^N)$

$$\mathbf{Br}^i(\pi^{-i}) = \arg \max_{\pi^i} \mathbf{E}_{a^i \sim \pi^i, a^{-i} \sim \pi^{-i}} [R^i(a^i, a^{-i})]$$

$$\text{Exploitability}(\pi) = \sum_{i=1}^2 R^i(\mathbf{Br}^i(\pi^{-i}), \pi^{-i}) - R^i(\pi)$$

# Nash Equilibrium in Two-Player Zero-Sum Games

- **von Neumann theorem:** Two-player Nash can be computed in P-time through linear programmes (LP).

## Prime problem

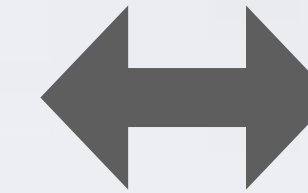
row player maximises the worst situation

$$\begin{aligned} \max_{v \in \mathbb{R}} v \\ \text{s.t. } \mathbf{p}^\top \mathbf{A} \geq v \cdot \mathbf{1} \\ \mathbf{p} \geq \mathbf{0} \text{ and } \mathbf{p}^\top \mathbf{1} = 1 \end{aligned}$$

## Dual problem

column player's view

$$\begin{aligned} \min_{v \in \mathbb{R}} v \\ \text{s.t. } \mathbf{q}^\top \mathbf{A}^\top \leq v \cdot \mathbf{1} \\ \mathbf{q} \geq \mathbf{0} \text{ and } \mathbf{q}^\top \mathbf{1} = 1 \end{aligned}$$



## Minimax theorem

zero-duality gap for convex problems

$$\begin{aligned} \max_{\mathbf{p}} \min_{\mathbf{q}} \mathbf{p}^\top \mathbf{A} \mathbf{q} \\ = \min_{\mathbf{q}} \max_{\mathbf{p}} \mathbf{p}^\top \mathbf{A} \mathbf{q} \end{aligned}$$

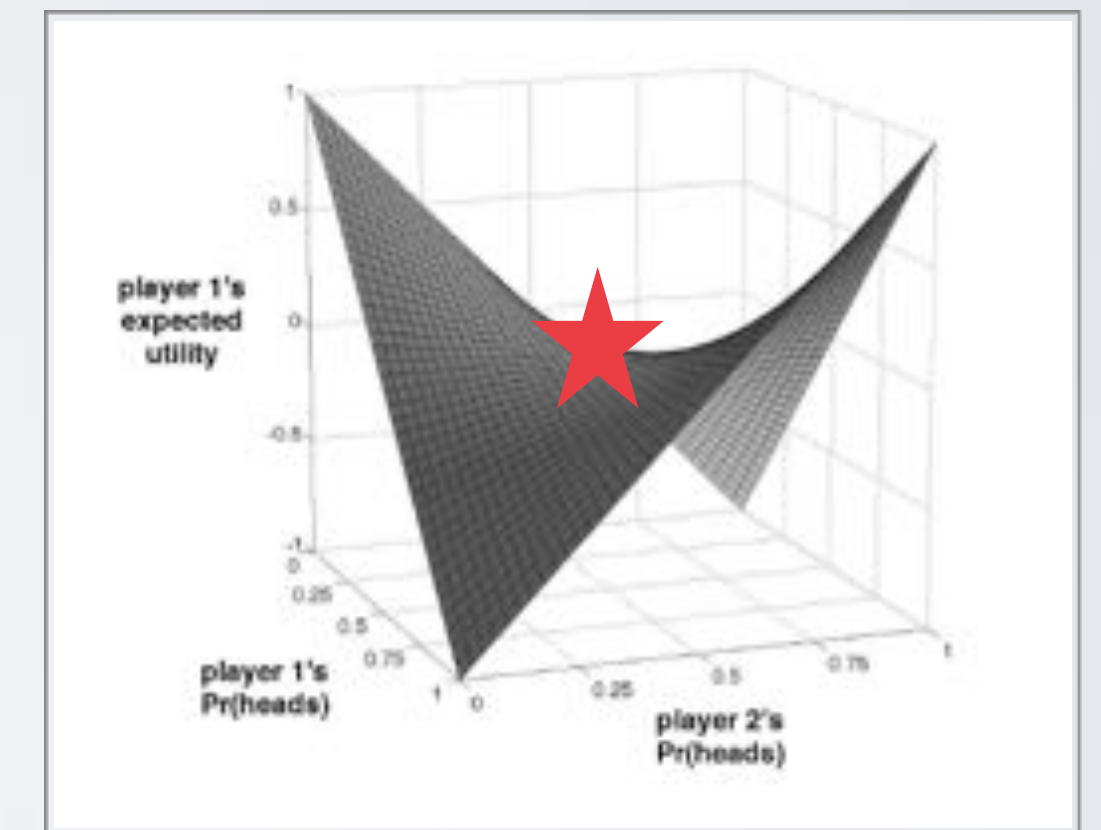
- The  $v^*$  is the Nash value

- proof:  $v \leq v^*$  due to definition of  $v^*$ ,  $v \geq v^*$  due to being the LP solution.
- corollary: all Nash value are the same (saddle point is unique in bimatrix game).

- The  $(\mathbf{p}, \mathbf{q})$  is the Nash equilibrium:

- proof: suppose the player plays  $\mathbf{x}, \mathbf{y}$  instead of  $\mathbf{p}, \mathbf{q}$

- $x^T A q = \sum_{i=1}^N x_i (A q)_i \leq \max_{i \in [N]} (A q)_i = v_c = v^*$ ,  $p^T A y = \sum_{j=1}^M (p^T A)_j y_j \geq \min_{j \in [M]} (p^T A)_j = v_r = v^*$ , thus no incentives to deviate.



- **Sion's minimax theorem** generalises to quasi-convex/concave functions  $\min_{x \in X} \sup_{y \in Y} f(x, y) = \sup_{y \in Y} \min_{x \in X} f(x, y)$

# When and Why we need Population-based Methods ?

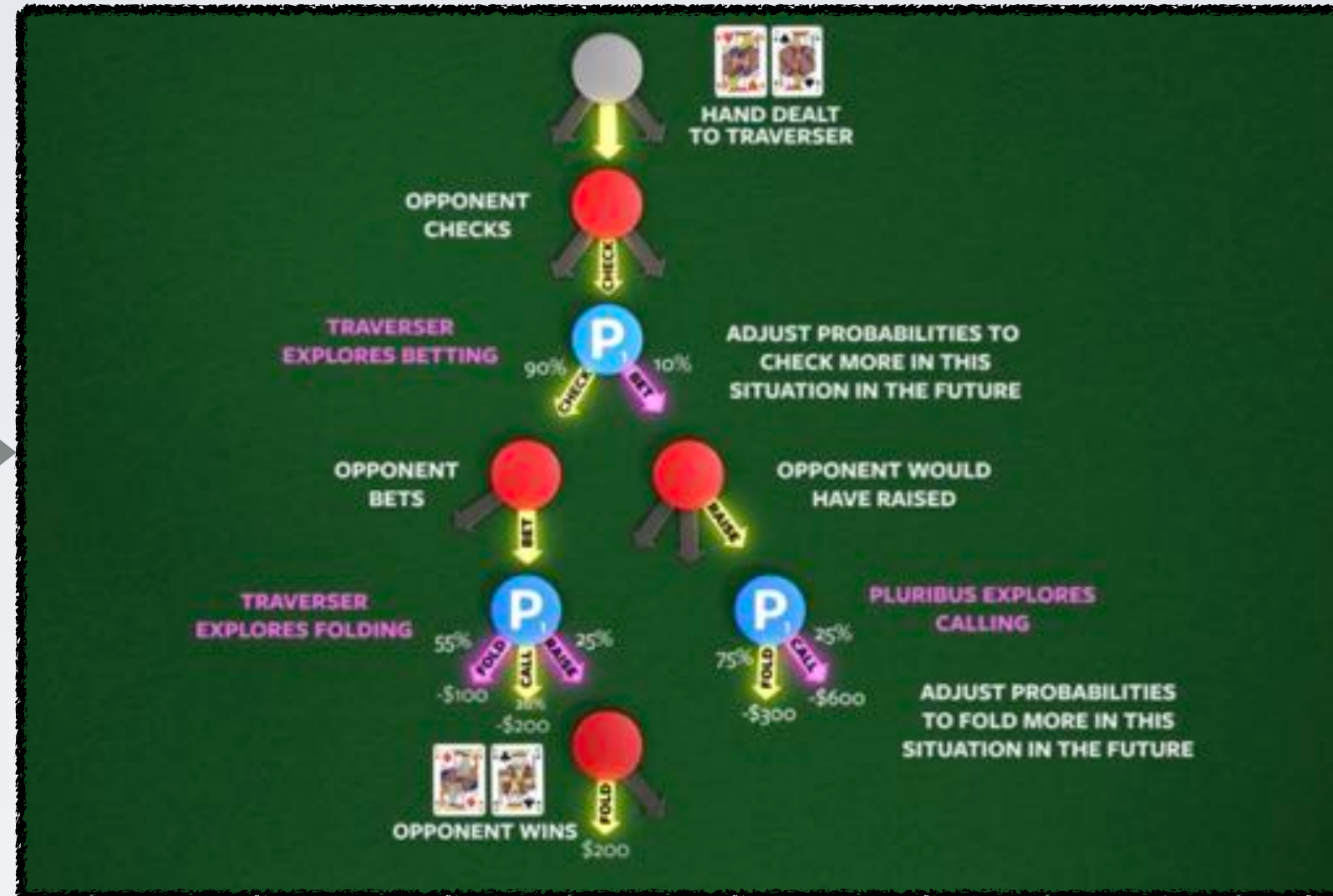
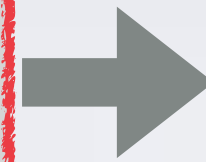
Output: the reward  $(R^1, \dots, R^N)$



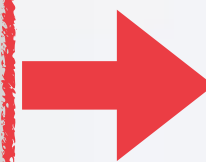
Black-box multi-agent game engine



Input: a joint strategy  $(\pi^1, \dots, \pi^N)$



Regret based methods: Poker Type



Population based methods: StarCraft type

When planning is feasible (game tree is easily accessible), existing techniques can solve the games really well.



Perfect-information games:  
MCTS, alpha-beta search, AlphaGO series (AlphaZero, MuZero, etc)

Imperfect-information:  
CFR series (DeepCFR, Libratus/Pluribus, Deepstack), XFP/NFSP series

When planning is not feasible. StarCraft has  $10^{26}$  choices per time step vs. the whole tree of chess  $10^{50}$  (Texas holdem  $10^{80}$ , GO  $10^{170}$ ). Enumerating all policies' actions at each state and then playing a best response is infeasible.

**Solution:** training a population of RL agents, treat each RL agent as one "pure strategy" and solve the game at a meta level where an agent is a RL model of a player, and we need a population of those agents (due to non-transitivity).

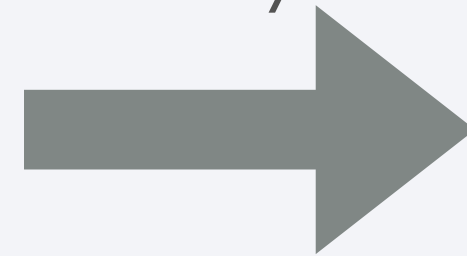
# Life up the problem to the meta level (i.e., the policy level)

- A player in zero-sum games usually have multiple strategies (Rock, Paper, Scissor).
- One strategy / policy corresponds to one “agent”  . A player  is represented by a population of agents.
- We now need to study the meta-game: **the game of a game, the problem problem.**
- We need to build that population of agents such that the player is unexploitable.



Player = Agent

meta-game  
analysis



A Player has a a population of Agents

# Formulation of Population Based Learning in Zero-Sum Games

- Let's formulate the self-play process.

- Suppose two agents, agent 1 adopts policy parameterised by  $\mathbf{v} \in \mathbb{R}^d$ , and agent 2 adopts policy  $\mathbf{w} \in \mathbb{R}^d$ . They can be considered as two neural networks.
- Define a **functional-form game (FFG)** [Balduzzi 2019] to be represented by a function

$$\phi : V \times W \rightarrow \mathbb{R}$$

RL model    RL model



- $\phi$  represents the game rule, it is anti-symmetrical.
- $\phi > 0$  means agent 1 wins over agent 2, the higher  $\phi(\mathbf{v}, \mathbf{w})$  the better for agent 1.
- with  $\phi_{\mathbf{w}}(\cdot) := \phi(\cdot, \mathbf{w})$ , we can have the best response defined by:

$$\mathbf{v}' := \mathbf{Br}(\mathbf{w}) = \mathbf{Oracle}(\mathbf{v}, \phi_{\mathbf{w}}(\cdot)) \quad \mathbf{s.t.} \quad \phi_{\mathbf{w}}(\mathbf{v}') > \phi_{\mathbf{w}}(\mathbf{v}) + \epsilon$$

- **Oracle**: a god tells us how to beat the enemy, it can be implemented by a RL algorithm, for example **PPO + PBT** as we have mentioned early, or other optimiser such as evolutionary algorithm.

# Naive Self-play Will Not Work

Question: Can we use it as a general framework to solve any games?

**PPO + Self-play = Panacea ?**

## Algorithm 2 Self-play

```
input: agent  $v_1$   
for  $t = 1, \dots, T$  do  
     $v_{t+1} \leftarrow \text{oracle}(v_t, \phi_{v_t}(\bullet))$   
end for  
output:  $v_{T+1}$ 
```

$$(\pi^1, \pi^2) \rightarrow (\pi^1, \pi^{2,*} = \mathbf{Br}(\pi^1)) \rightarrow (\pi^{1,*} = \mathbf{Br}(\pi^{2,*}), \pi^{2,*})$$

**It depends. In most of the games, it does not work.**



self-plays

# Naive Self-play Will Not Work

- It is because of **Non-Transitivity**

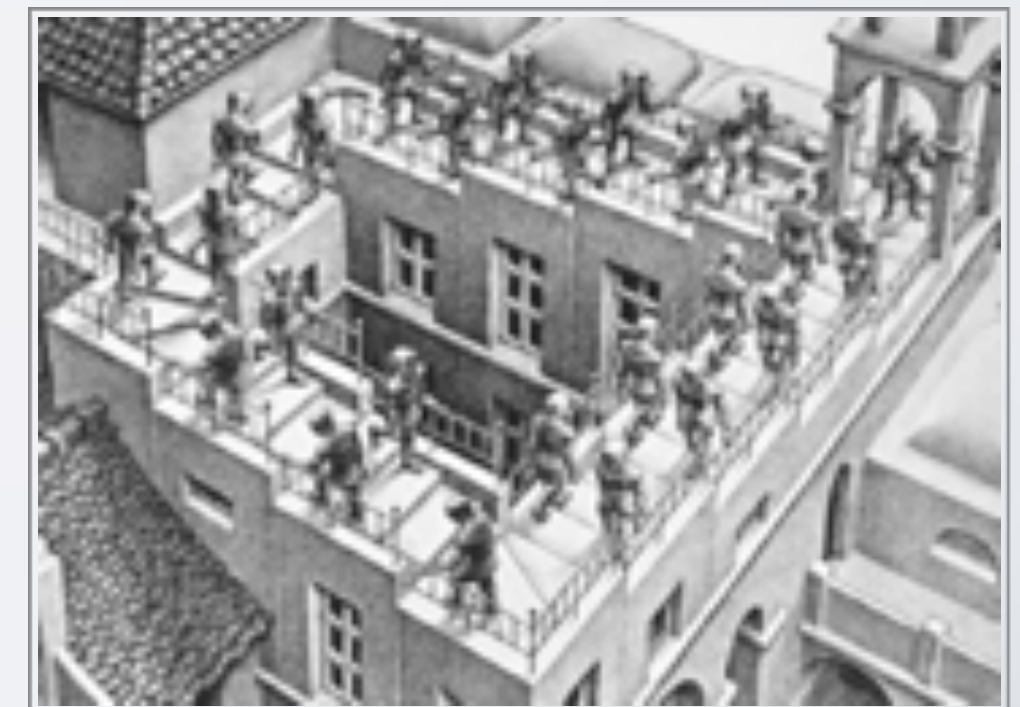
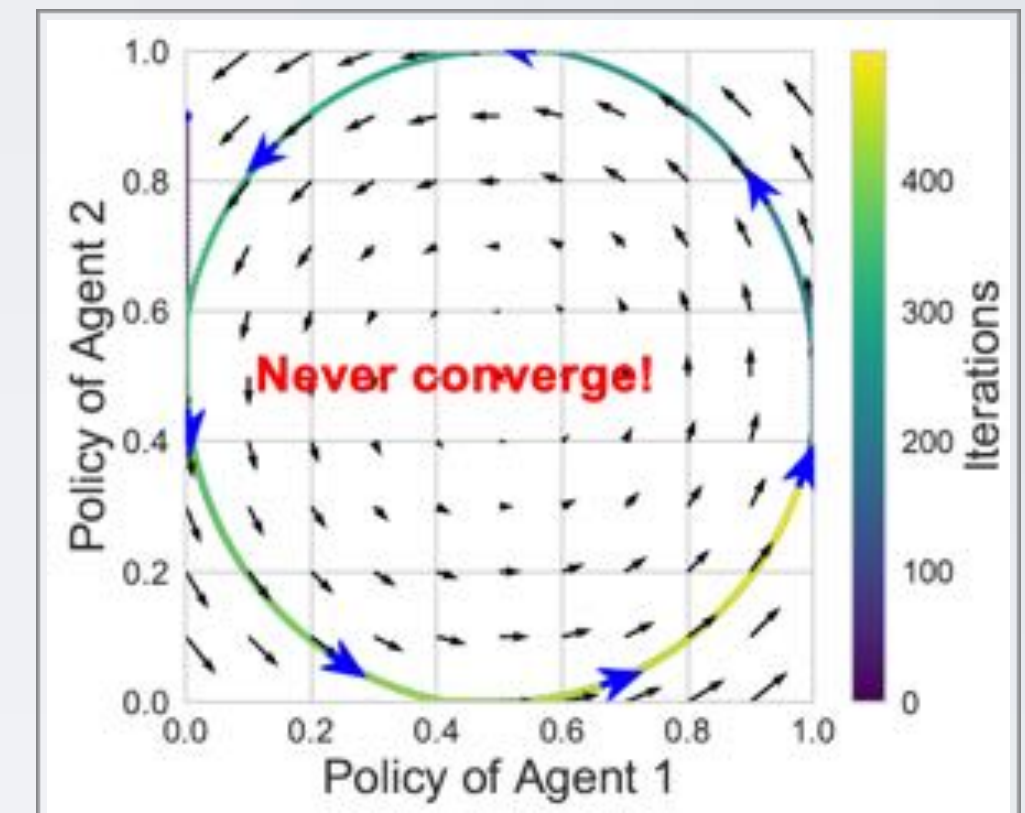
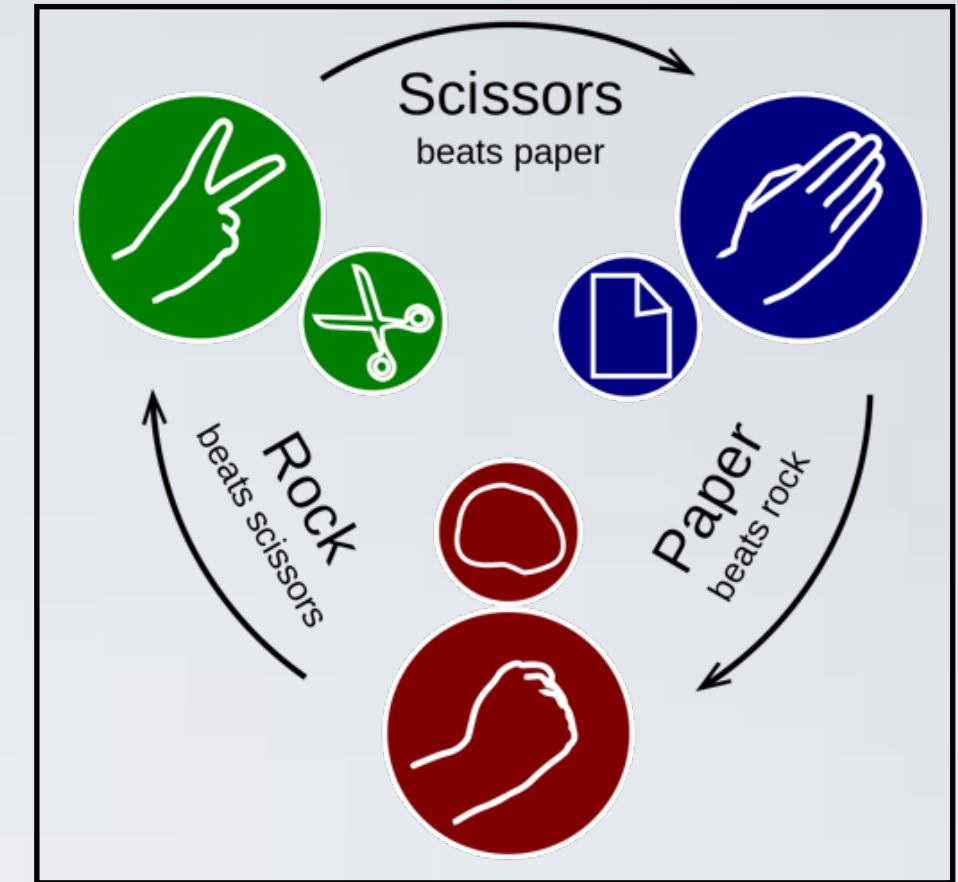
$$\int_W \phi(\mathbf{v}, \mathbf{w}) \cdot d\mathbf{w} = 0, \quad \forall \mathbf{v} \in W$$

- Rock-Paper-Scissor game:

$$\begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$

- Disc game:

$$\phi(\mathbf{v}, \mathbf{w}) = \mathbf{v}^\top \cdot \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \mathbf{w} = v_1 w_2 - v_2 w_1$$



# Game Decomposition

- Every FFG can be decomposed into two parts [Balduzzi 2019]

$$\text{FFG} = \text{Transitive game} \oplus \text{Non-transitive game}$$

- Let  $\mathcal{V}, \mathcal{W} \in W$  be a compact set and  $\phi(\mathcal{V}, \mathcal{W})$  prescribe the flow from  $\mathcal{V}$  to  $\mathcal{W}$ , then this is a natural result after applying *combinatorial hodge theory* [Jiang 2011].
- We can write any games  $\phi$  as summation of two **orthogonal** components

$$\text{grad}(f)(\mathbf{v}, \mathbf{w}) := f(\mathbf{v}) - f(\mathbf{w}) \quad \text{div}(\phi)(\mathbf{v}) := \int_{\mathcal{W}} \phi(\mathbf{v}, \mathbf{w}) \cdot d\mathbf{w} \quad \text{curl}(\phi)(\mathbf{u}, \mathbf{v}, \mathbf{w}) := \phi(\mathbf{u}, \mathbf{v}) + \phi(\mathbf{v}, \mathbf{w}) - \phi(\mathbf{u}, \mathbf{w})$$

$$\phi = \underbrace{\text{grad} \circ \text{div}(\phi)}_{\text{curl}(\cdot)=0} + \underbrace{(\phi - \text{grad} \circ \text{div}(\phi))}_{\text{div}(\cdot)=0}$$

curl(·)=0

div(·)=0

Transitive game

Non-transitive game

- Example on Rock-Paper-Scissor

	R	P	S
R	0, 0	-3x, 3x	3y, -3y
P	3x, -3x	0, 0	-3z, 3z
S	-3y, 3y	3z, -3z	0, 0

(a) Generalized RPS Game

=

	R	P	S
R	(y-x), (y-x)	(y-x), (x-z)	(y-x), (z-y)
P	(x-z), (y-x)	(x-z), (x-z)	(x-z), (z-y)
S	(z-y), (y-x)	(z-y), (x-z)	(z-y), (z-y)

(c) Potential Component

+

	R	P	S
R	0, 0	-(x+y+z), (x+y+z)	(x+y+z), -(x+y+z)
P	(x+y+z), -(x+y+z)	0, 0	-(x+y+z), (x+y+z)
S	-(x+y+z), (x+y+z)	(x+y+z), -(x+y+z)	0, 0

(d) Harmonic Component

+

	R	P	S
R	(x-y), (x-y)	(z-x), (x-y)	(y-z), (x-y)
P	(x-y), (z-x)	(z-x), (z-x)	(y-z), (z-x)
S	(x-y), (y-z)	(z-x), (y-z)	(y-z), (y-z)

(b) Nonstrategic Component

Transitive game

Non-transitive game



# What is Transitivity ?

- Every FFG can be decomposed into two parts

$$\text{FFG} = \text{Transitive game} \oplus \text{Non-transitive game}$$

- **Transitive Game:** the rules of winning are transitive across different players.

$$\mathbf{v}_t \text{ beats } \mathbf{v}_{t-1}, \quad \mathbf{v}_{t+1} \text{ beats } \mathbf{v}_t \quad \rightarrow \quad \mathbf{v}_{t+1} \text{ beats } \mathbf{v}_{t-1}$$

- Example: Elo rating (段位) offers **rating scores**  $f(\cdot)$  that assume transitivity.

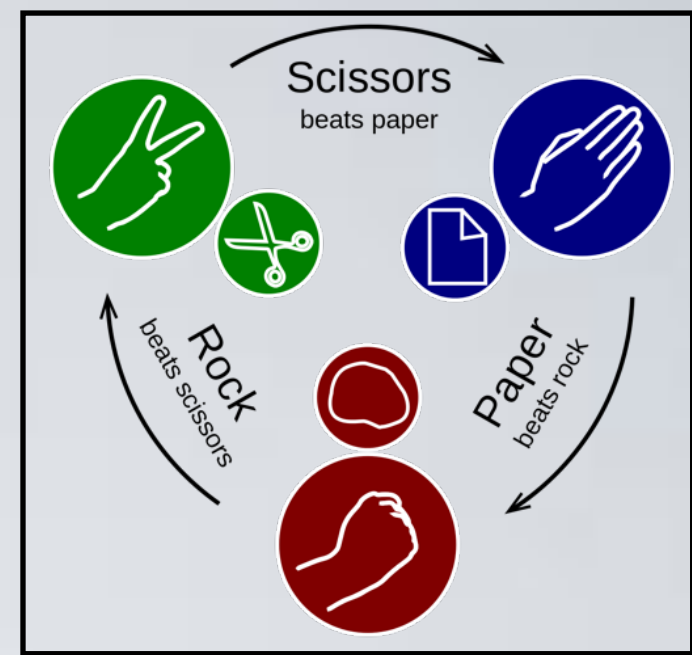
$$\phi(\mathbf{v}, \mathbf{w}) = \text{softmax}(f(\mathbf{v}) - f(\mathbf{w}))$$

- Larger score means you are likely to win over players with lower scores.
- Elo score is widely used in GO and Chess.
- This explains why you don't want to play with rookies, when  $f(\mathbf{v}_t) \gg f(\mathbf{w})$ ,

$$\nabla_{\mathbf{v}} \phi(\mathbf{v}_t, \mathbf{w}) \approx 0$$

# What is Non-Transitivity ?

- Every FFG can be decomposed into two parts



$$\text{FFG} = \text{Transitive game} \oplus \text{Non-transitive game}$$

- **Non-transitive Game:** the rules of winning are not-transitive across players.

$$v_t \text{ beats } v_{t-1}, \quad v_{t+1} \text{ beats } v_t \not\Rightarrow v_{t+1} \text{ beats } v_{t-1}$$

- Mutual dominance across different types of modules in a game. This is commonly observed in modern MOBA games.

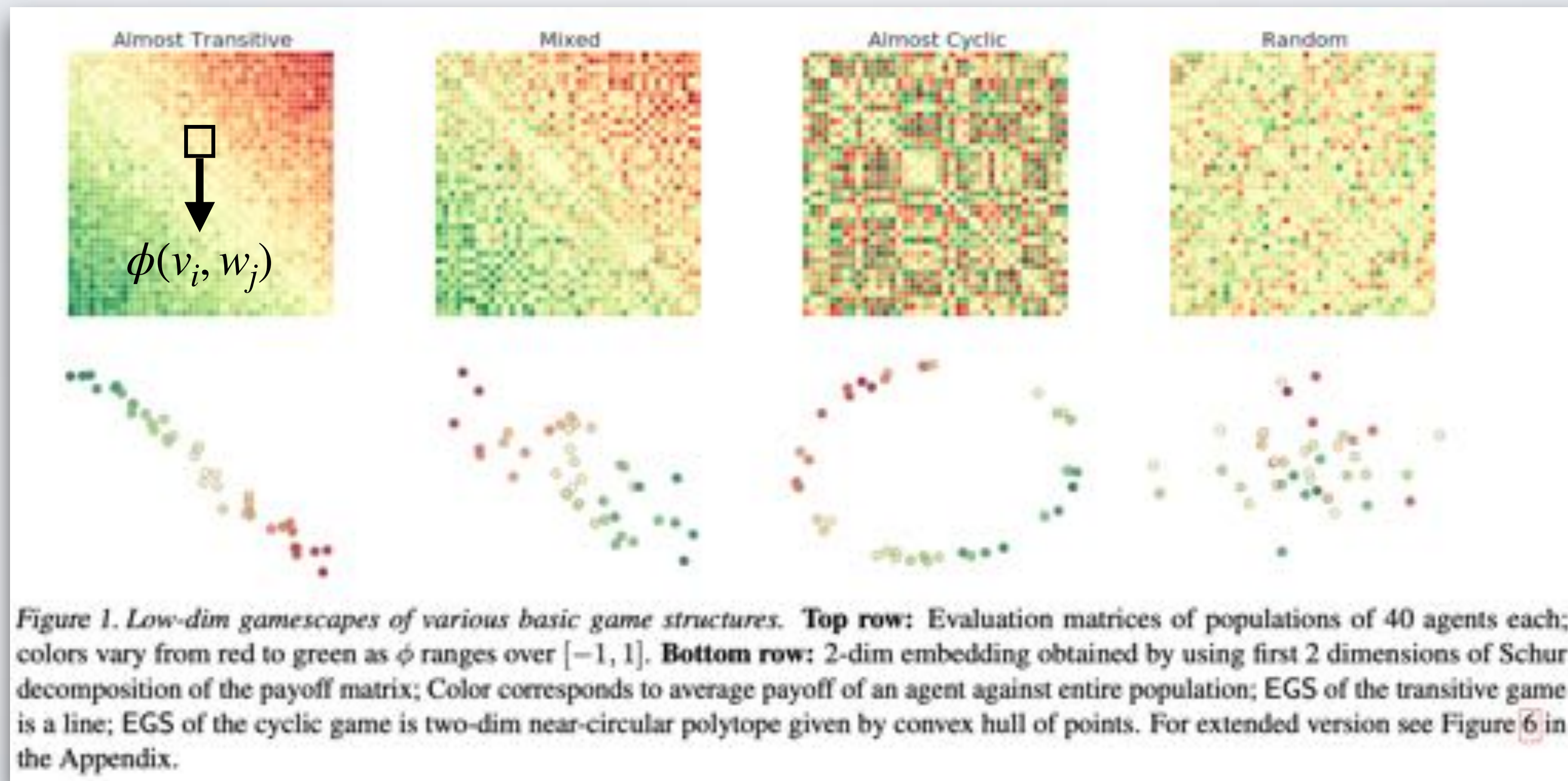


- For this types of game, self-play is not helpful at all because transitivity assumption does not hold. **Self-play will lead to cyclic loops forever.**

# Visualisation of Transitive and Non-Transitive Games

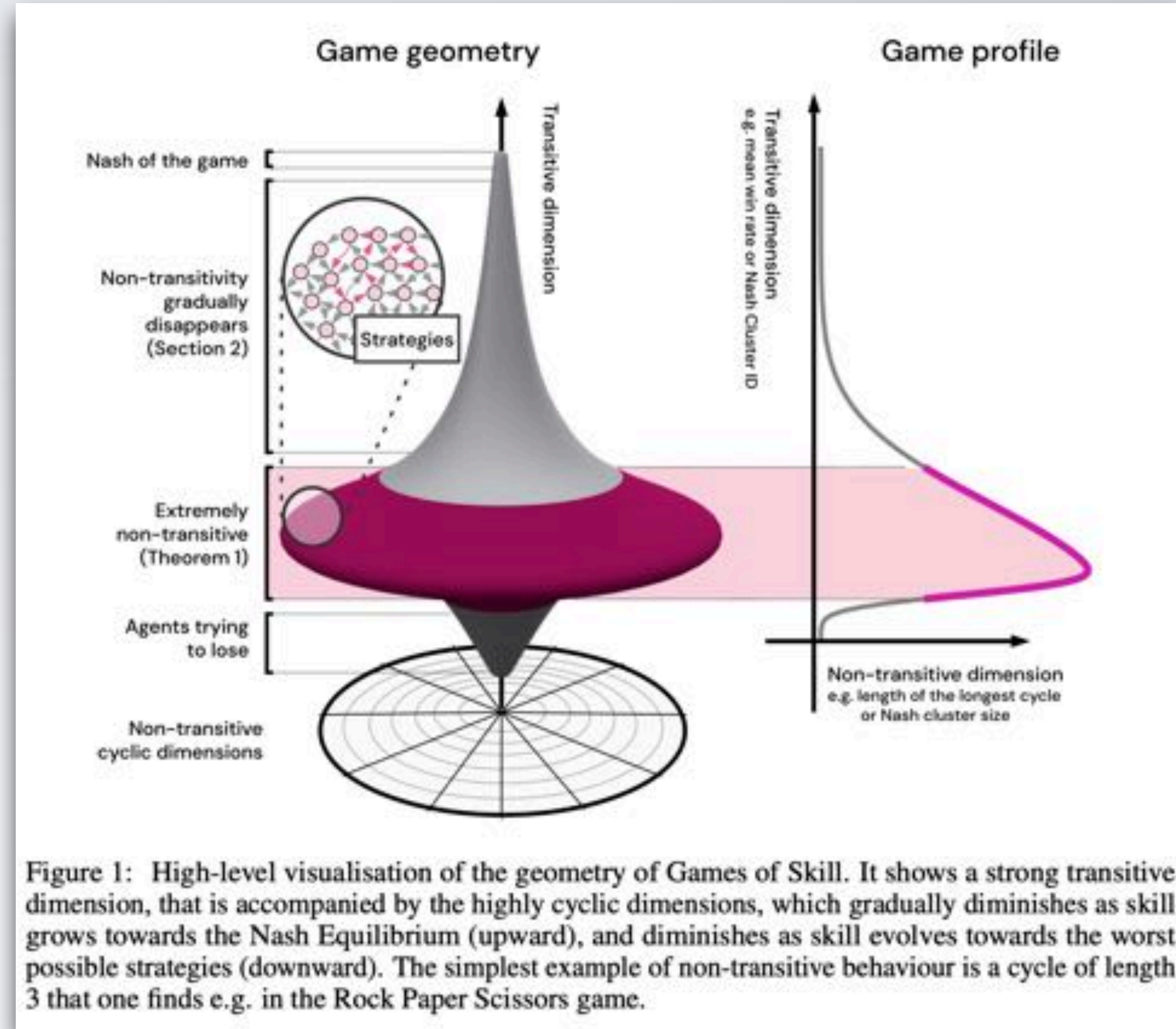
- Let us define the evaluation matrix for a population of  $N$  agents to be

$$\mathbf{A}_{\mathfrak{P}} := \left\{ \phi(\mathbf{w}_i, \mathbf{w}_j) : (\mathbf{w}_i, \mathbf{w}_j) \in \mathfrak{P} \times \mathfrak{P} \right\} =: \phi(\mathfrak{P} \otimes \mathfrak{P})$$



# The Spinning Top Hypothesis

- Real-world games are mixtures of both transitive and in-transitive components, e.g., Go, DOTA, StarCraft II.
- Though winning is often harder than losing a game, finding a strategy that always loses is also challenging.
- Players who regularly practice start to beat less skilled players, this corresponds to the transitive dynamics.
- At certain level (the red part), players will start to find many different strategy styles. Despite not providing a universal advantage against all opponents, players will counter each other within the same transitive group. This provide direct information of improvement.
- As players get stronger to the highest level, seeing many strategy styles, the outcome relies mostly on skill and less on one particular game styles (以不变应万变).



# Measuring the Non-Transitivity

- A **theoretical lower bound** of the size of non-transitivity [Czarnecki 2020]
  - ◆ n-bit communicative game

**Definition 1.** Consider the extensive form view of the win-draw-loss version of any underlying game; the underlying game is called **n-bit communicative** if each player can transmit  $n \in \mathbb{R}_+$  bits of information to the other player before reaching the node whereafter at least one of the outcomes 'win' or 'loss' is not attainable.

**bit: how many action one can take before the outcome of the game is predetermined**

**Theorem 1.** For every game that is at least **n-bit communicative**, and every antisymmetric win-loss payoff matrix  $\mathbf{P} \in \{-1, 0, 1\}^{\lfloor 2^n \rfloor \times \lfloor 2^n \rfloor}$ , there exists a set of  $\lfloor 2^n \rfloor$  pure strategies  $\{\pi_1, \dots, \pi_{\lfloor 2^n \rfloor}\} \subset \Pi$  such that  $\mathbf{P}_{ij} = \mathbf{f}^\dagger(\pi_i, \pi_j)$ , and  $\lfloor x \rfloor = \max_{a \in \mathbb{N}} a \leq x$ .

**n-bit game = there exists at least a non-transitive circle of size  $2^n$**

- ◆ Results on GO and MOBA games:

**Proposition 1.** The game of Go is at least **1000-bit communicative** and contains a cycle of length at least  $2^{1000}$ .

**Proposition 2.** Modern games, such as StarCraft, DOTA or Quake, when limited to 10 minutes play, are at least **36000-bit communicative**.

# Measuring the Non-Transitivity

- A **practical way** of measurement through meta-game analysis
  - ◆ Computing n-bit communicative game needs full tree traversing, thus intractable
  - ◆ Deciding a graph has a path of length higher than k is **NP-hard**
  - ◆ One needs to **approximate**.

**Approximating Longest Directed Paths and Cycles**

Andreas Björklund<sup>1</sup>, Thore Husfeldt<sup>1</sup>, and Sanjeev Khanna<sup>2\*</sup>

<sup>1</sup> Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden.  
thore@cs.lu.se

<sup>2</sup> Dept. of CIS, University of Pennsylvania, Philadelphia, PA 19104.  
sanjeev@cis.upenn.edu

**Abstract.** We investigate the hardness of approximating the longest path and the longest cycle in directed graphs on  $n$  vertices. We show that neither of these two problems can be polynomial time approximated within  $n^{1-\epsilon}$  for any  $\epsilon > 0$  unless  $P = NP$ . In particular, the result holds for digraphs of constant bounded outdegree that contain a Hamiltonian cycle.

- ◆ Method I, count the **number of RPS cycles**.

- when  $k=3$ , we can compute by constructing  $A_{i,j} = 1 \iff \phi_{i,j} > 0$ , then

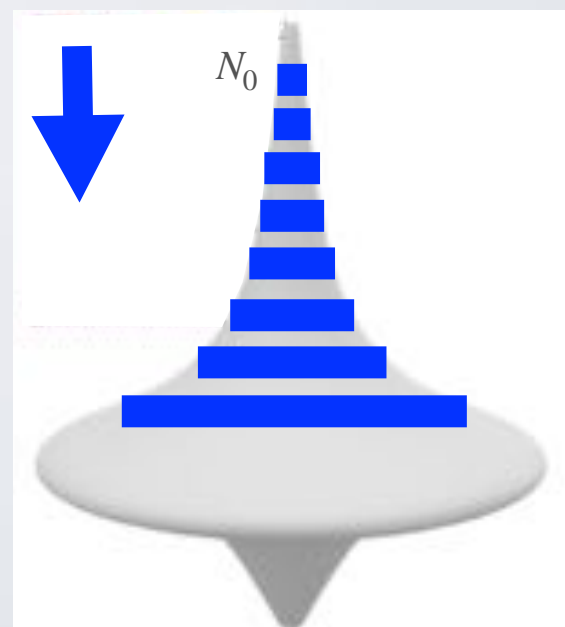
$$\text{diag}(A^3)$$

- ◆ Method II, at each transitivity level, we can measure the **Nash Clustering**

**Definition 3.** *Nash clustering  $\mathbf{C}$  of the finite zero-sum symmetric game strategy  $\Pi$  set by setting for each  $i \geq 1$ :  $N_{i+1} = \text{supp}(\text{Nash}(\mathbf{P} | \Pi \setminus \bigcup_{j \leq i} N_j))$  for  $N_0 = \emptyset$  and  $\mathbf{C} = (N_j : j \in \mathbb{N} \wedge N_j \neq \emptyset)$ .*

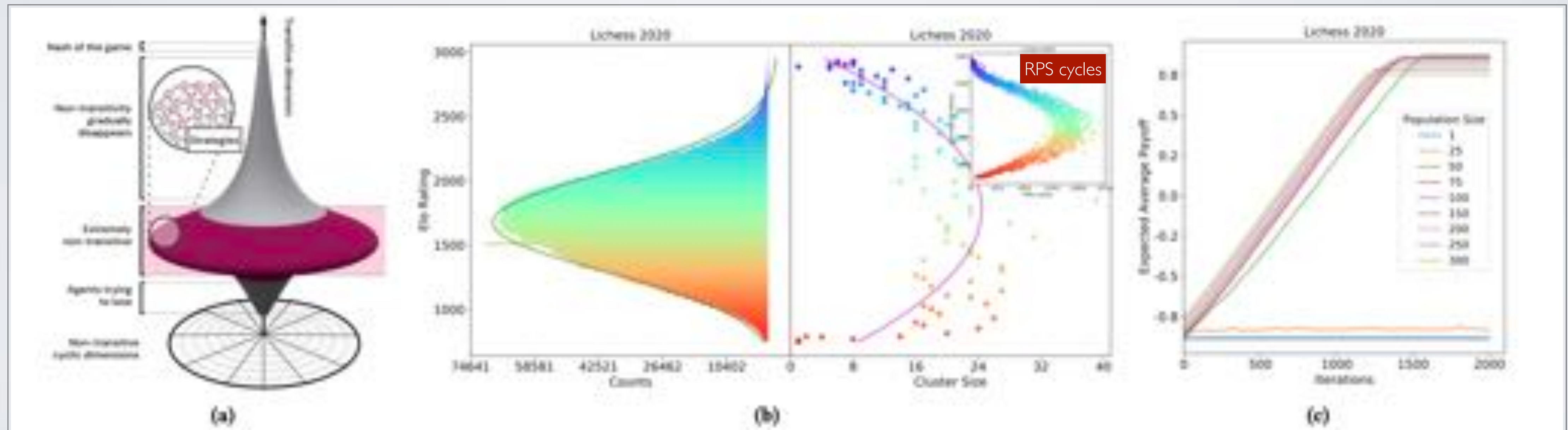
$$N_{i+1} = \text{supp}(\text{Nash}(\mathbf{P} | \Pi \setminus \bigcup_{j \leq i} N_j))$$

strategies that at the  
higher level of transitivity



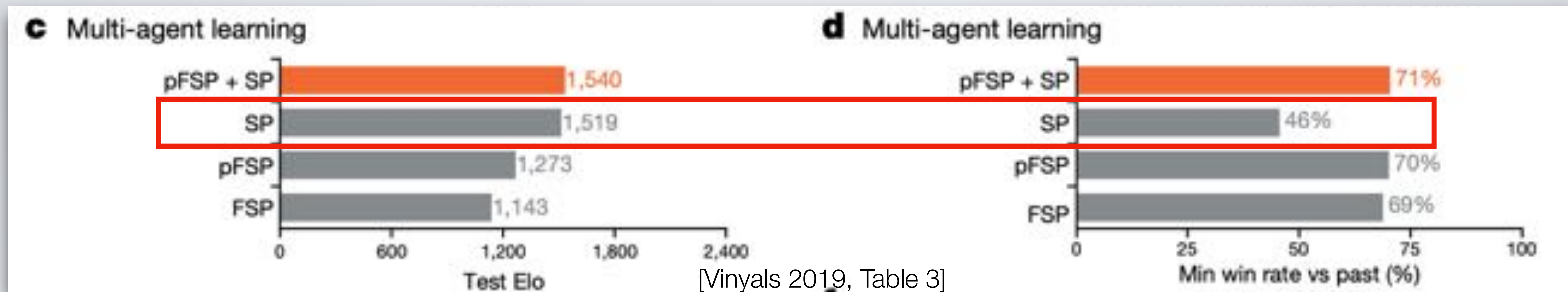
# Measuring the Non-Transitivity in Chess

- Real-world data set from human players on Chess
  - ◆ We study one billion human player records from Lichess platform
  - ◆ **Human Chess players presents the spinning-top pattern, which verifies the hypothesis**



# Non-Transitivity Harms Training !

## Example on training AlphaStar:



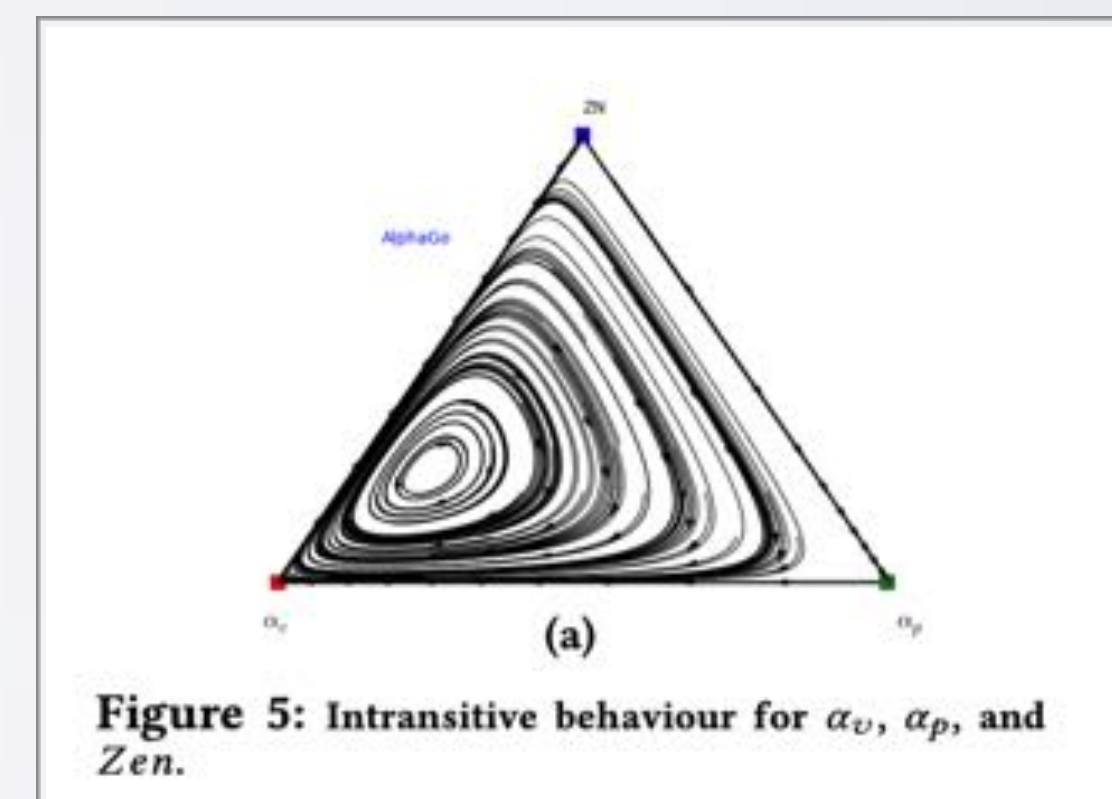
## Example on training Soccer AI:

Table 2: Average goal difference  $\pm$  one standard deviation across 5 repetitions of the experiment.

<i>A</i> vs built-in AI	$4.25 \pm 1.72$
<i>B</i> vs <i>A</i>	$11.93 \pm 2.19$
<i>B</i> vs built-in AI	$-0.27 \pm 0.33$

[Karol 2020, table 2]

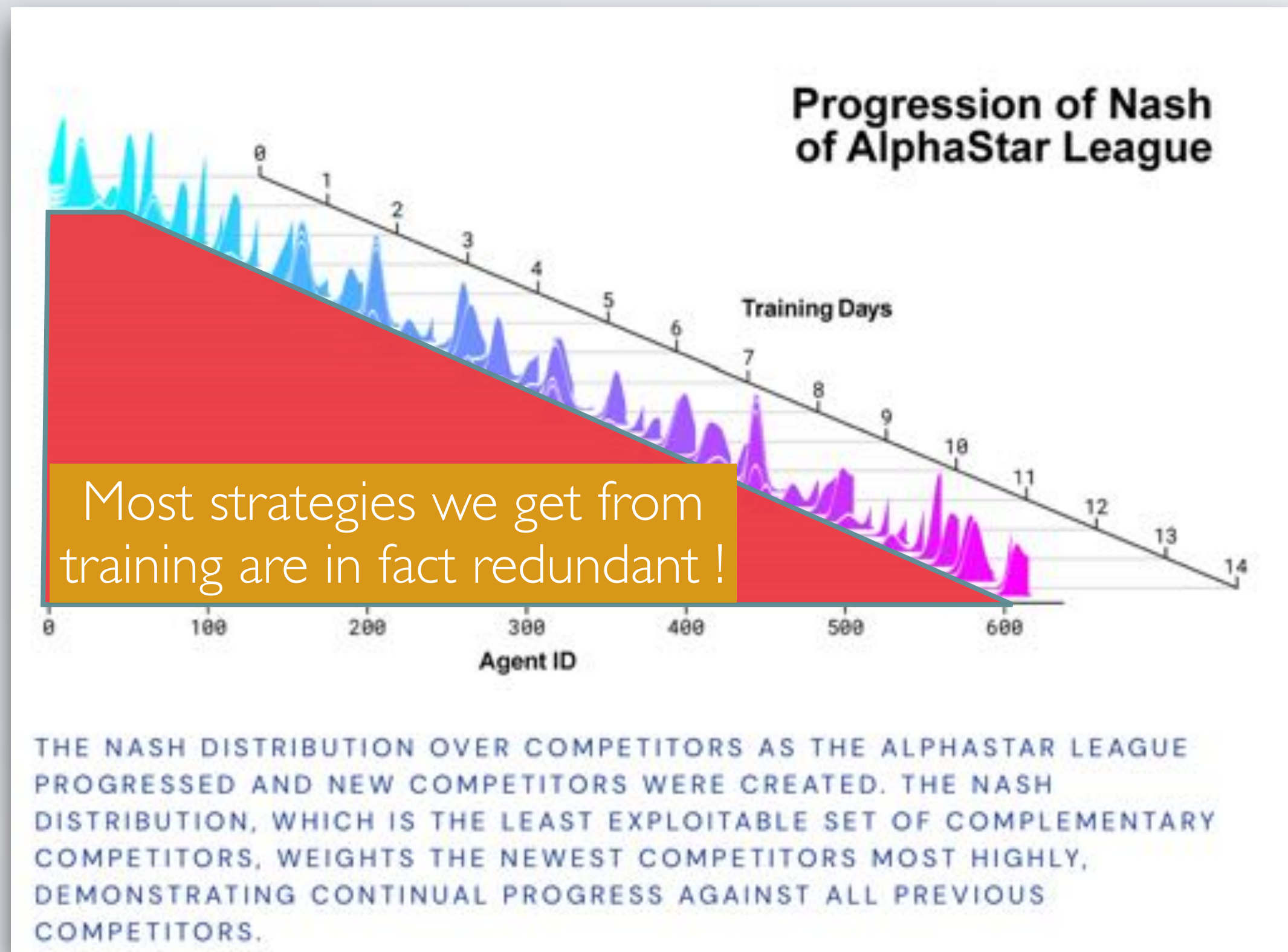
## Example on training AlphaGO:



[Silver 2016, table 9]



# Dealing With Non-Transitivity Helps Save Training Time



[AlphaStar Blog]

Table 2: Size of the Nash Support of Games

Game	Total Strategies	Size of Nash support
3-Move Parity Game 2	160	1
5,4-Blotto	56	6
AlphaStar	888	3
Connect Four	1470	23
Disc Game	1000	27
Elo game + noise=0.1	1000	6
Elo game	1000	1
Go (boardsize=3,komi=6.5)	1933	13
Misere (game=tic tac toe)	926	1
Normal Bernoulli game	1000	5
Quoridor (boardsize=3)	1404	1
Random game of skill	1000	5
Tic Tac Toe	880	1
Transitive game	1000	1
Triangular game	1000	1

[Le Ceong Dinh 2021]

# Understanding Non-Transitivity Helps Develop Algorithms !

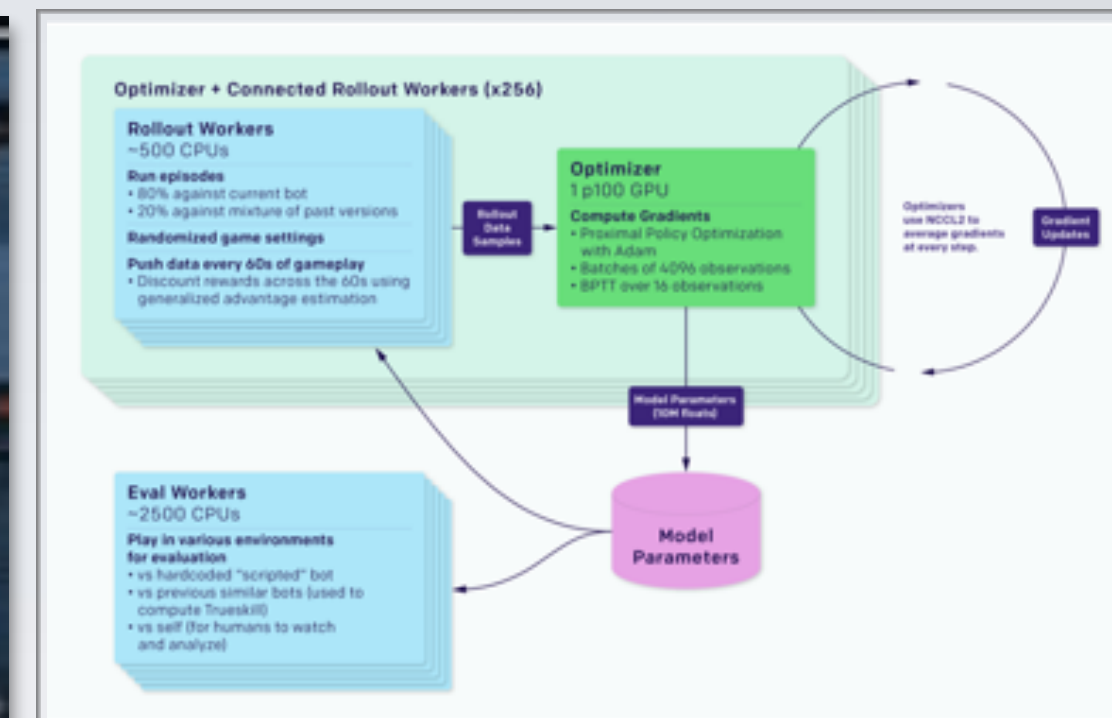
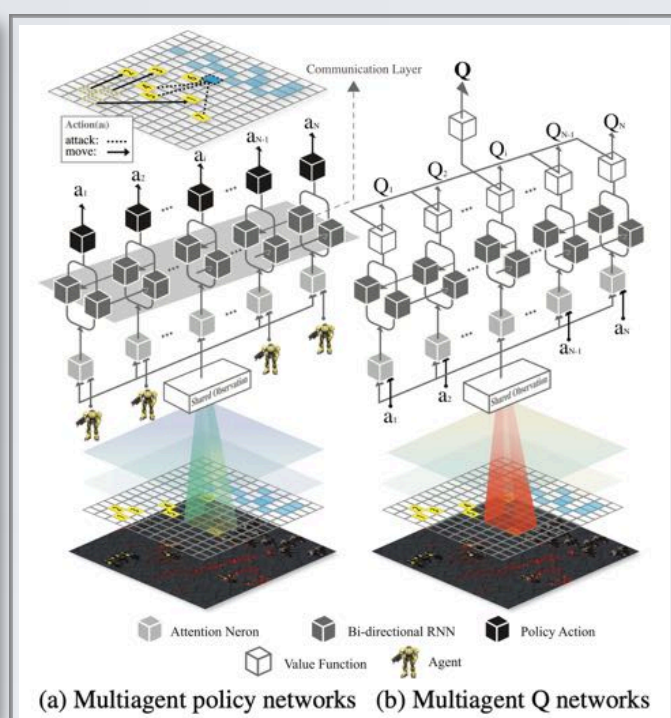
- Topological structure at the policy space affects the efficiency of training algorithm.
  - ◆ for example, there is a reason why we need **diversity** in the policy space.

**Theorem 3.** *If at any point in time, the training population  $\mathcal{P}^t$  includes any full Nash cluster  $C_i \subset \mathcal{P}^t$ , then training against  $\mathcal{P}^t$  by finding  $\pi$  such that  $\forall \pi_j \in \mathcal{P}^t \mathbf{f}(\pi, \pi_j) > 0$  guarantees transitive improvement in terms of the Nash clustering  $\exists_{k < i} \pi \in C_k$ .*

- ◆ on chess, large population size (thus more diversity) will have a phase change in the strength !

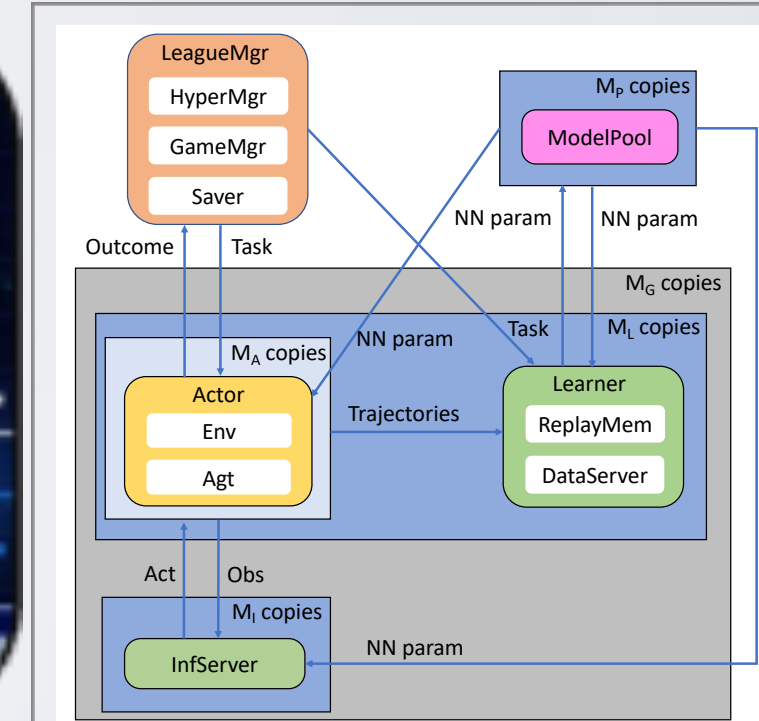
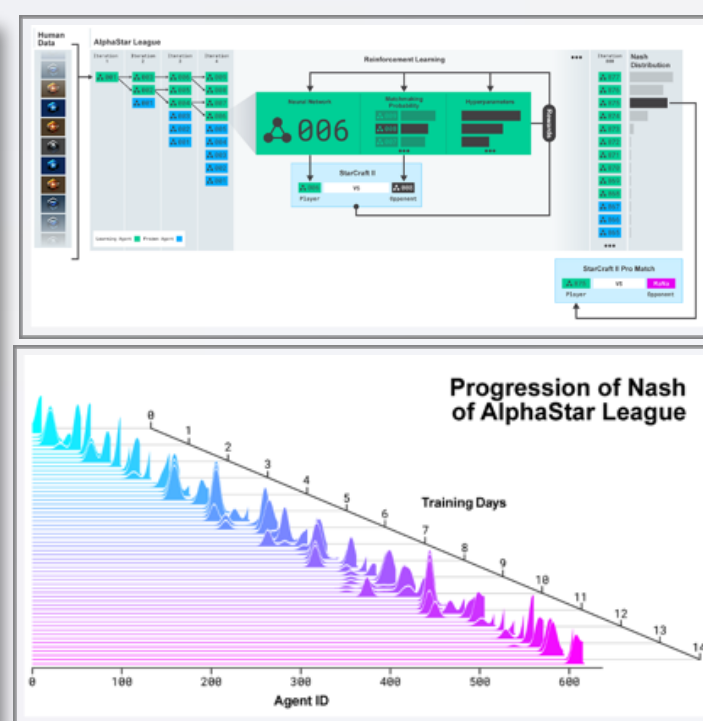
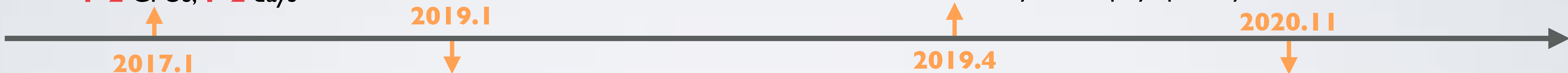


# Understanding Non-Transitivity Helps Develop Efficient Algorithms !



StarCraft micro-management  
BiCNet, deep MARL methods  
**1-2 GPUs, 1-2 days**

Dota 2 full game (OpenAI Five)  
Population-based + Rapid training system  
**128,000 CPUs, 100 GPUs, 180 years of plays per day**

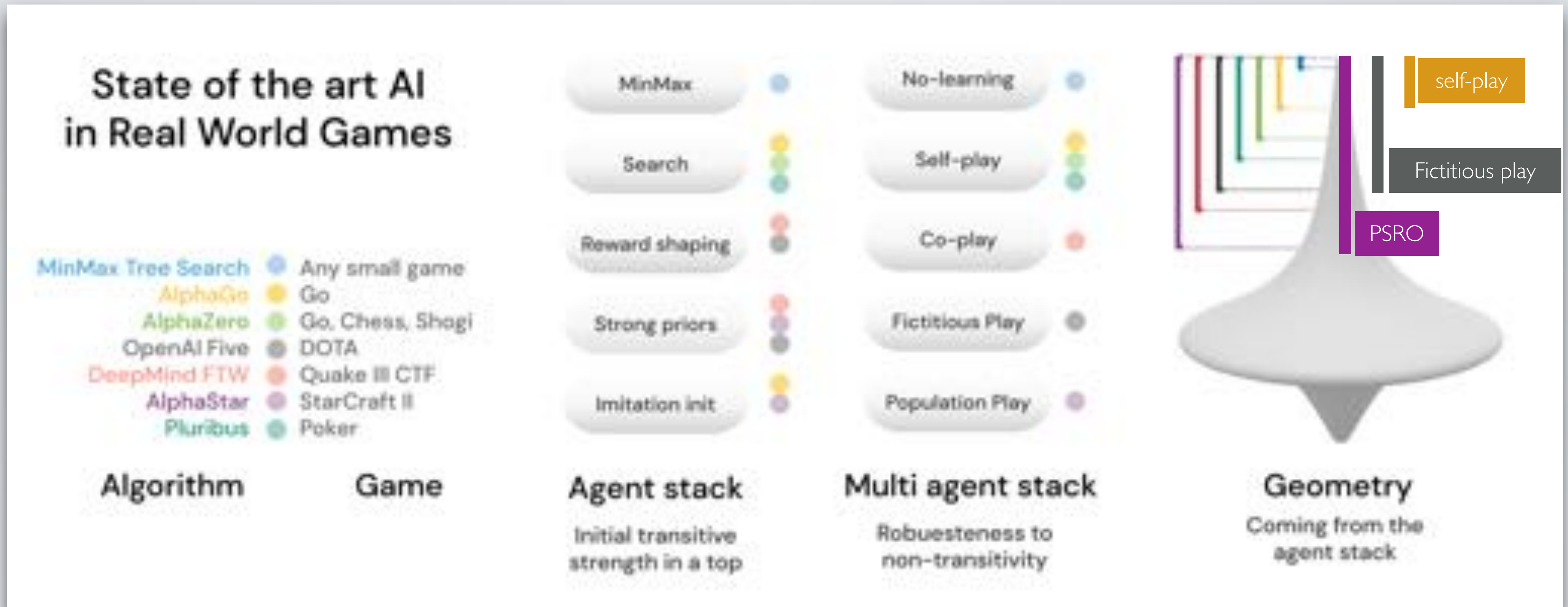


StarCraft full game (AlphaStar)  
Populating-based Training  
Training for single agent costs **14 days, 16 TPU/Agent, 200 years of real-time play.**

王者荣耀 (绝悟)

Populating-based Competitive Self-play + Policy distillation  
**35,000 CPUs, 320 GPUs, begin to converge after 336 hours**

# Understanding Non-Transitivity Helps Develop Efficient Algorithms !



[Czarnecki 2020]

# Solutions: Fictitious Play [Brown 1951]

- Maintain a belief over the historical actions that the opponent has played, and the learning agent then takes the best response to this **empirical average distribution**.

$$a_i^{t,*} \in \mathbf{BR}_i \left( p_{-i}^t = \frac{1}{t} \sum_{\tau=0}^{t-1} \mathcal{F} \{ a_{-i}^\tau = a, a \in \mathbb{A} \} \right)$$

$$p_i^{t+1} = \left( 1 - \frac{1}{t} \right) p_i^t + \frac{1}{t} a_i^{t,*}, \text{ for all } i$$

- It guarantees to converge, in terms of the Nash value, in two-player zero-sum games, potential games and  $2 \times 2$  games, and, the average policy converge to the Nash strategy.

- Examples:

		Player 2	
		a	b
Player 1	A	(1,1)	(0,0)
	B	(0,0)	(1,1)

t	$p_1^t$	$p_2^t$	$a_1^t$	$a_2^t$
0	(3/4, 1/4)	(1/4, 3/4)	B	a
1	(3/4, 5/4)	(5/4, 3/4)	A	b
2	(7/4, 5/4)	(5/4, 7/4)	B	a
3	(7/4, 9/4)	(9/4, 7/4)	A	b
⋮	⋮	⋮	⋮	⋮

$\infty$  (1/2, 1/2) (1/2, 1/2)

# Generalised Weakened Fictitious Play [Leslie 2006]

- It releases the FP by allowing **approximate best response** and **perturbed average strategy updates**, while maintaining the same convergence guarantee if **conditions** met.

$$\mathbf{Br}_i^\epsilon(p_{-i}) = \left\{ p_i : R_i(p_i, p_{-i}) \geq R_i(\mathbf{Br}_i(p_{-i}), p_{-i}) - \epsilon \right\}$$

$$p_i^{t+1} = \left(1 - \alpha^{t+1}\right)p_i^t + \alpha^{t+1} \left( \mathbf{Br}_i^\epsilon(p_{-i}) + M_i^{t+1} \right), \text{ for all } i$$

$$t \rightarrow \infty, \alpha_t \rightarrow 0, \epsilon^t \rightarrow 0, \sum_{t=1}^{\infty} \alpha^t = \infty, \{M^t\} \text{ meets } \limsup_{t \rightarrow \infty} \left\{ \left\| \sum_{i=t}^{k-1} \alpha^{i+1} M^{i+1} \right\| \text{ s.t. } \sum_{i=t}^{k-1} \alpha^{i+1} \leq T \right\} = 0$$

- Recovers normal Fictitious Play when  $\alpha^t = 1/t, \epsilon_t = 0, M_t = 0$ .
- **Why important:** it allows us to use a broad class of best responses such as RL algorithms, and also, the policy exploration in e.g. soft-Q learning. Also, GWFP makes FP no-regret by introducing the perturbation term  $M$ .

# Solutions: Double Oracle [McMahan 2003]

- Double Oracle best responds to the **opponent's Nash equilibrium** at each iteration.
- To solve the game before seeing all pure strategies (not all of them are in Nash), much faster than LP, but In the worst-case scenario, it recovers to solve the original game.

## Algorithm 1 Double Oracle (McMahan et al., 2003)

```

1: Input: A set  $\Pi, C$  strategy set of players
2:  $\Pi_0, C_0$ : initial set of strategies
3: for  $t = 1$  to  $\infty$  do
4:   if  $\Pi_t \neq \Pi_{t-1}$  or  $C_t \neq C_{t-1}$  then
5:     Solve the NE of the subgame  $G_t$ :
6:      $(\pi_t^*, c_t^*) = \arg \min_{\pi \in \Delta_{\Pi_t}} \arg \max_{c \in \Delta_{C_t}} \pi^\top A c$ 
7:     Find the best response  $a_{t+1}$  and  $c_{t+1}$  to  $(\pi_t^*, c_t^*)$ :
8:      $a_{t+1} = \arg \min_{a \in \Pi} a^\top A c_t^*$ 
9:      $c_{t+1} = \arg \max_{c \in C} \pi_t^{*\top} A c$ 
10:    Update  $\Pi_{t+1} = \Pi_t \cup \{a_{t+1}\}, C_{t+1} = C_t \cup \{c_{t+1}\}$ 
11:   else if  $\Pi_t = \Pi_{t-1}$  and  $C_t = C_{t-1}$  then
12:     Terminate
13:   end if
14: end for

```

- **iteration 0:** restricted game R vs R
- **iteration 1:**
  - solve Nash of restricted game  $(1, 0, 0), (1, 0, 0)$
  - unrestricted  $\mathbf{Br}^1, \mathbf{Br}^2 = P, P$
- **iteration 2:**
  - solve Nash of restricted games  $(0, 1, 0), (0, 1, 0)$
  - unrestricted  $\mathbf{Br}^1, \mathbf{Br}^2 = S, S$
- **iteration 3:**
  - solve Nash of restricted game  $(1/3, 1/3, 1/3), (1/3, 1/3, 1/3)$
- **iteration 4:** no new response, END
  - output  $(1/3, 1/3, 1/3)$

	R	P	S
R	0	-1	1
P	1	0	-1
S	-1	1	0

# Double Oracle [McMahan 2003]

- It guarantees to converge to Nash equilibrium in two-player zero-sum games, and coarse correlated equilibrium in multi-player general-sum games.

- **Convergence proof:**

- ◆ DO finally recovers to solve the whole game

- **Correctness proof:**

- ◆ DO stops at the  $j$ -th sub-game, we can prove no new best responses are added

- ◆  $\forall p, V(p, q_j) \geq v \Rightarrow \forall p, \max_q V(p, q) \geq v$

$$\forall q, V(p_j, q) \leq v \Rightarrow \max_q V(p_j, q) \leq v$$

$$\Rightarrow \forall p, \max_q V(p_j, q) \leq \max_q V(p, q)$$

$p_j$  must be the minimax optimal,

$q_j$  vice versa



# Policy Space Response Oracle = Double Oracle with RL Agent

- A generalisation of double oracle methods on **meta-games**, with the best responder is implemented through **deep RL algorithms**.
- A meta-game is  $(\Pi, U, n)$  where  $\Pi = (\Pi_1, \dots, \Pi_n)$  is the set of policies for each agent and  $U : \Pi \rightarrow \mathbb{R}^n$  is the reward values for each agent given a joint strategy profile.
- $\sigma_{-i}$  is distribution over  $(\Pi_1^0, \dots, \Pi_1^T)$ , a.k.a meta-solver
- PSRO generalises all previous methods by varying  $\sigma_{-i}$ .
  - **independent learning**:  $\sigma_{-i} = (0, \dots, 0, 0, 1)$
  - **self-play**:  $\sigma_{-i} = (0, \dots, 0, 1, 0)$
  - **fictitious play**:  $\sigma_{-i} = (1/T, 1/T, \dots, 1/T, 0)$
  - **PSRO**:  $\sigma_{-i} = \mathbf{Nash}(\Pi^{T-1}, U)$  or  $\mathbf{RD}(\Pi^{T-1}, U)$

## Algorithm 1: Policy-Space Response Oracles

```
input : initial policy sets for all players  $\Pi$ 
Compute exp. utilities  $U^\Pi$  for each joint  $\pi \in \Pi$ 
Initialize meta-strategies  $\sigma_i = \text{UNIFORM}(\Pi_i)$ 
while epoch  $e$  in  $\{1, 2, \dots\}$  do
    for player  $i \in [[n]]$  do
        for many episodes do
            select opponent policies Sample  $\pi_{-i} \sim \sigma_{-i}$ 
            compute the best response Train oracle  $\pi'_i$  over  $\rho \sim (\pi'_i, \pi_{-i})$ 
            augment strategy pool  $\Pi_i = \Pi_i \cup \{\pi'_i\}$ 
            expand the payoff matrix Compute missing entries in  $U^\Pi$  from  $\Pi$ 
            solve the new meta game Compute a meta-strategy  $\sigma$  from  $U^\Pi$ 
        Output current solution strategy  $\sigma_i$  for player  $i$ 
```

# **PART II: PSRO Methods and Its Variations**

Oliver Slumbers



# Contents

- **Rectified Nash**
- **Diverse-PSRO**
- **Unified Behavioural + Response Diversity**
- **NAC**
- **$\alpha$ -PSRO**
- **Joint PSRO**
- **Pipeline PSRO**
- **Mixed Oracles / Opponents**

# Contents

- **Rectified Nash**

- Diverse-PSRO

- Unified Behavioural + Response Diversity

- NAC

- $\alpha$ -PSRO

- Joint PSRO

- Pipeline PSRO

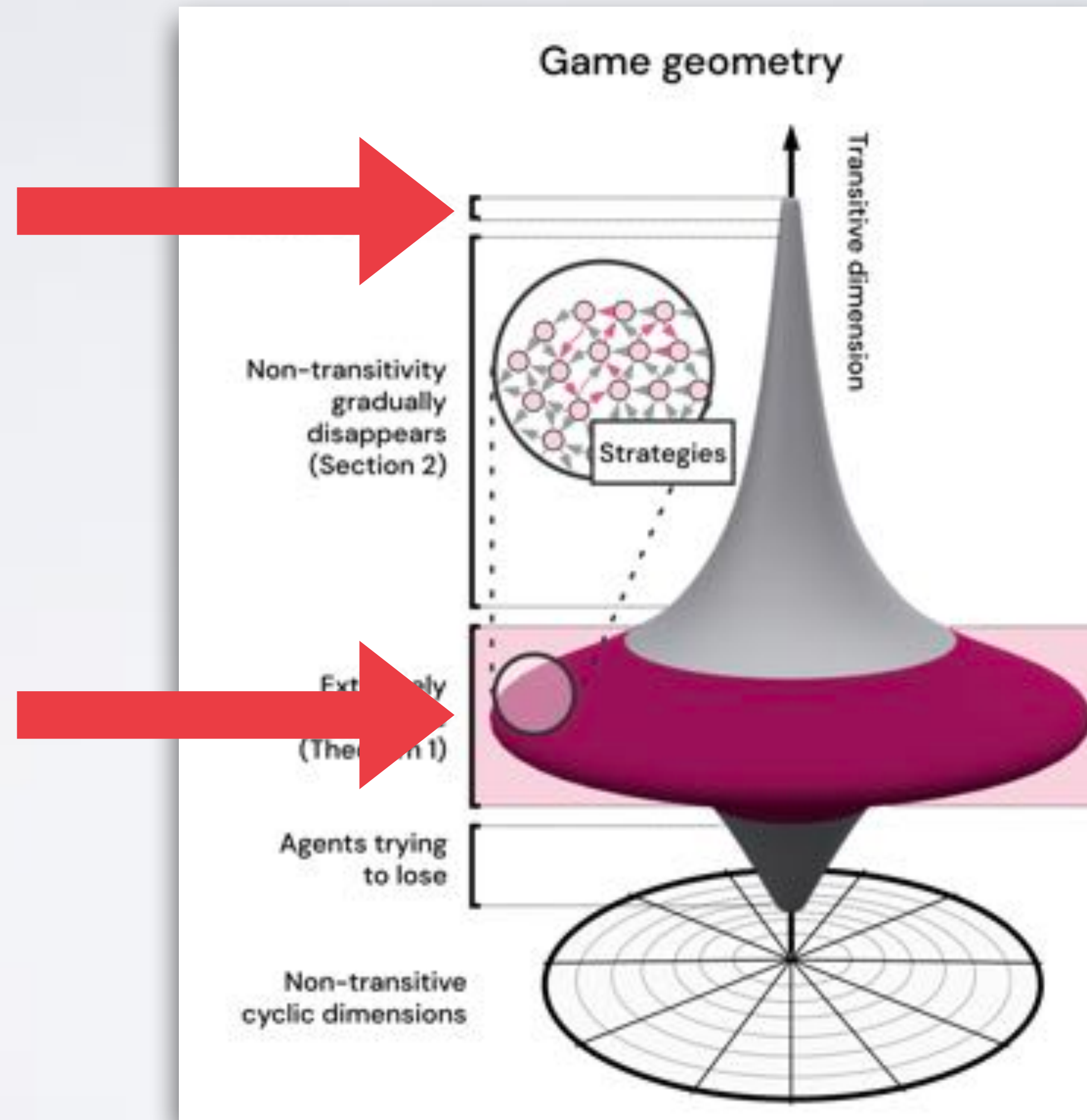
- Mixed Oracles / Opponents

# Meta-Game Structure [Czarnecki et al. 2020]

Interesting games display a particular **spinning-top** structure

Diversity disappears and skill becomes the dominant factor, i.e. the game becomes fully **transitive**

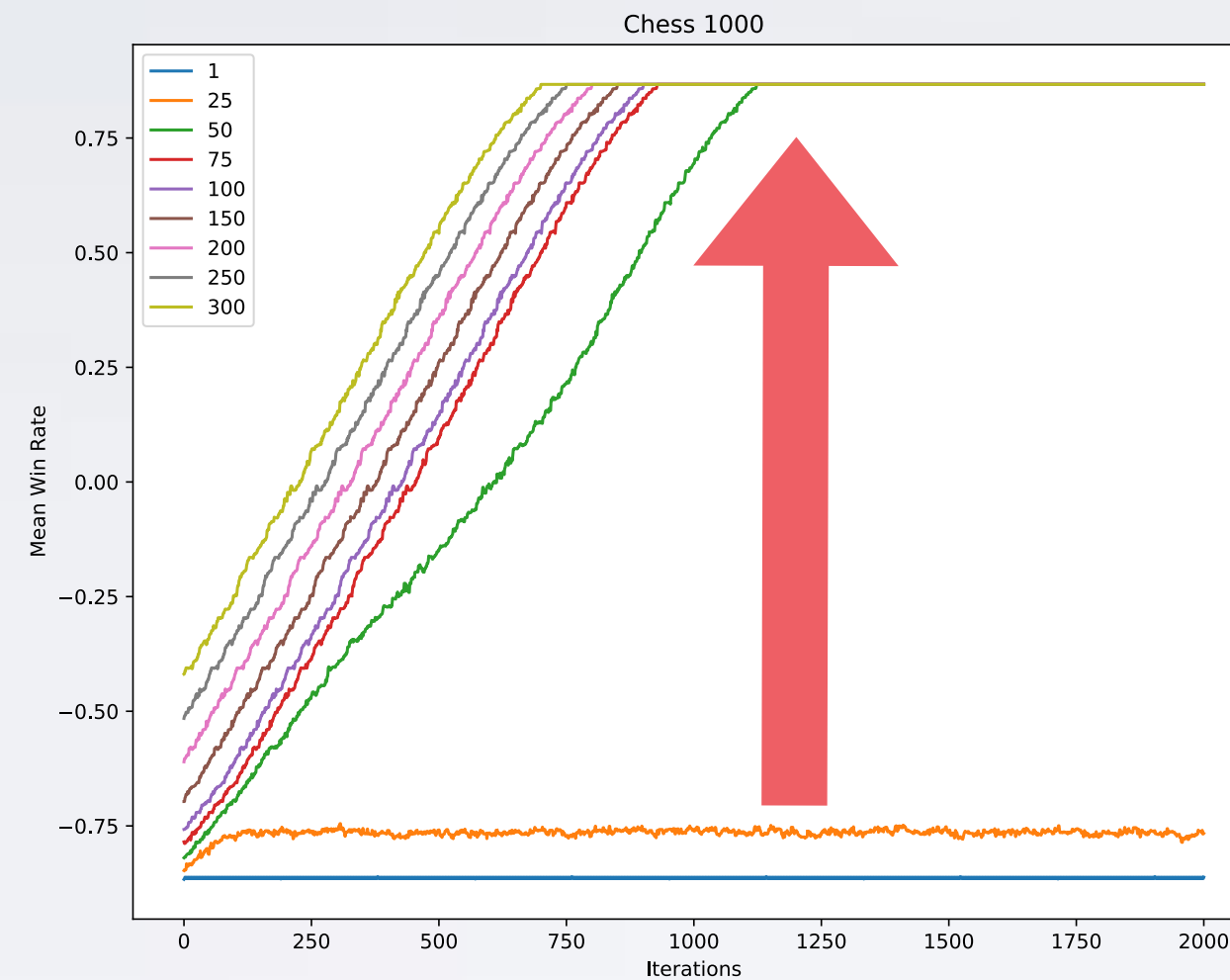
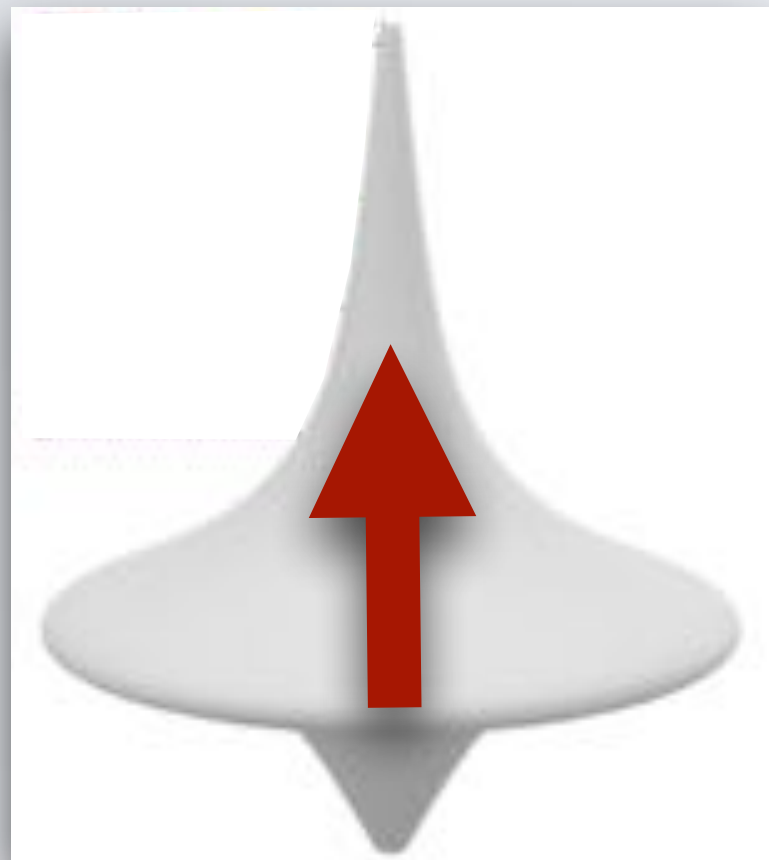
Diverse game-styles are prevalent and perform similarly to each-other, i.e. we are in the **non-transitive** layer



The big question is how does one move efficiently between the layers?

# Why is Diversity Important?

**Theorem 3.** *If at any point in time, the training population  $\mathcal{P}^t$  includes any full Nash cluster  $C_i \subset \mathcal{P}^t$ , then training against  $\mathcal{P}^t$  by finding  $\pi$  such that  $\forall \pi_j \in \mathcal{P}^t \mathbf{f}(\pi, \pi_j) > 0$  guarantees transitive improvement in terms of the Nash clustering  $\exists_{k < i} \pi \in C_k$ .*



**Diverse Auto-Curriculum is Critical for Successful Real-World Multiagent Learning Systems\***

Blue Sky Ideas Track

Yaodong Yang <sup>†</sup> University College London Huawei R&D U.K.	Jun Luo Huawei Canada	Ying Wen Shanghai Jiao Tong University
Oliver Slumbers University College London	Daniel Graves Huawei Canada	Haitham Bou Ammar Huawei R&D U.K.
Jun Wang University College London Huawei R&D U.K.	Matthew E. Taylor University of Alberta Alberta Machine Intelligence Institute	

- Diversity matters because **the more diverse** the population pool, **the less exploitable**. Promoting diversity can help you break out of in-transitive regions faster.
- In real-world applications, you want policies to cover different skill-sets. This is a **realistic need** from **autonomous driving** and **gaming AI** applications.

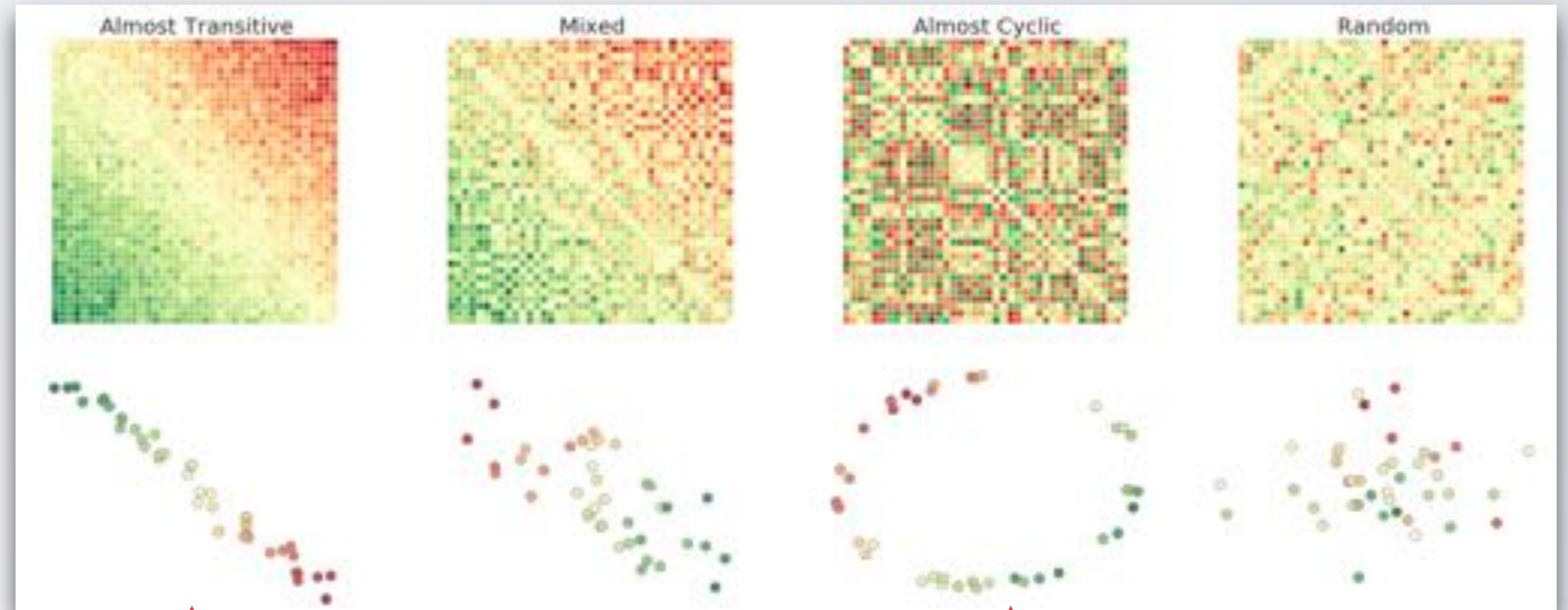
# Gamescapes

- A crucial component in characterising a population is that of the **empirical gamescape**
- Measure all ways agents can and are observed to interact with each other

Given population  $\mathfrak{P}$  of  $n$  agents with evaluation matrix  $\mathbf{A}_{\mathfrak{P}}$ , the corresponding **empirical gamescape** (EGS) is

$$\mathcal{G}_{\mathfrak{P}} := \{ \text{convex mixtures of rows of } \mathbf{A}_{\mathfrak{P}} \}.$$

- Schur Decomposition of certain payoff matrices paints an intuitive picture
- Games show an obvious gamescape structure



Obvious linear/transitive structure

Obvious cyclic/non-transitive structure

# PSRO-rN [Balduzzi et al. 2019] - Algorithm

**Definition 4.** Denote the rectifier by  $[x]_+ := x$  if  $x \geq 0$  and  $[x]_+ := 0$  otherwise. Given population  $\mathfrak{P}$ , let  $\mathbf{p}$  be a Nash equilibrium on  $\mathbf{A}_{\mathfrak{P}}$ . The effective diversity of the population is:

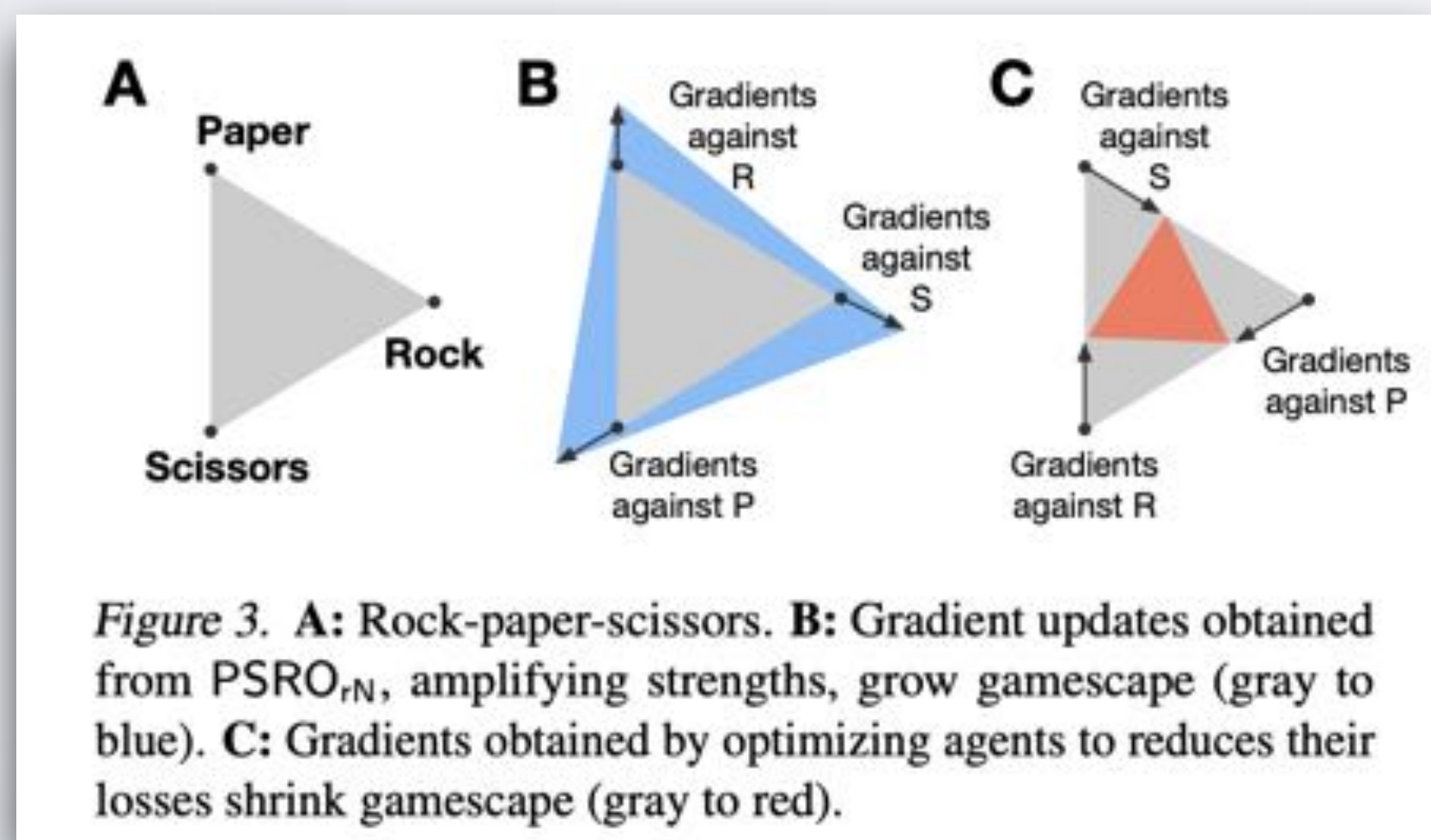
$$d(\mathfrak{P}) := \mathbf{p}^\top \cdot [\mathbf{A}_{\mathfrak{P}}]_+ \cdot \mathbf{p} = \sum_{i,j=1}^n [\phi(\mathbf{w}_i, \mathbf{w}_j)]_+ \cdot p_i p_j.$$

Effective diversity quantifies how the best agents in a population exploit each other - **Dominant Agent = 0 Diversity**

**key changes:** only selecting opponents that you already beat (i.e. rectifying the Nash)

$$\mathbf{v}_{t+1} \leftarrow \text{oracle} \left( \mathbf{v}_t, \sum_{\mathbf{w}_i \in \mathfrak{P}_t} \mathbf{p}_t[i] \cdot [\phi_{\mathbf{w}_i}(\cdot)]_+ \right)$$

**Proposition 6.** If  $\mathbf{p}$  is a Nash equilibrium on  $\mathbf{A}_{\mathfrak{P}}$  and  $\sum_i p_i \phi_{\mathbf{w}_i}(\mathbf{v}) > 0$ , then adding  $\mathbf{v}$  to  $\mathfrak{P}$  strictly enlarges the empirical gamescape:  $\mathcal{G}_{\mathfrak{P}} \subsetneq \mathcal{G}_{\mathfrak{P} \cup \{\mathbf{v}\}}$ .



Intuition: improving ones strengths allows for exploration of the strategy space

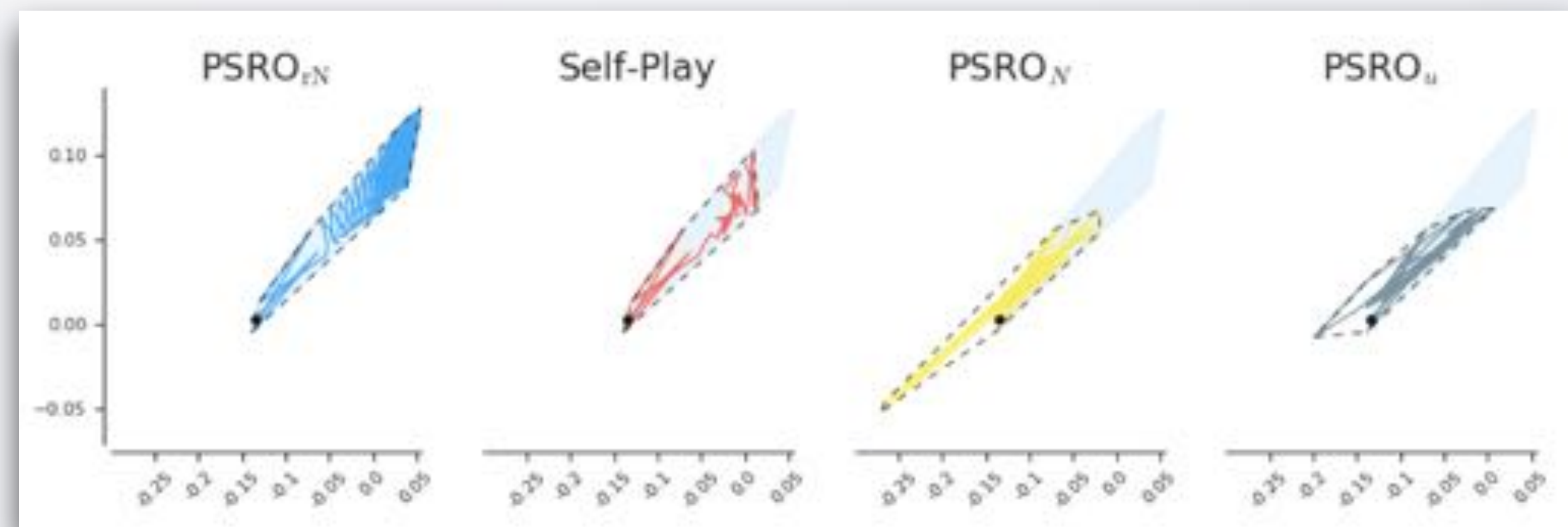
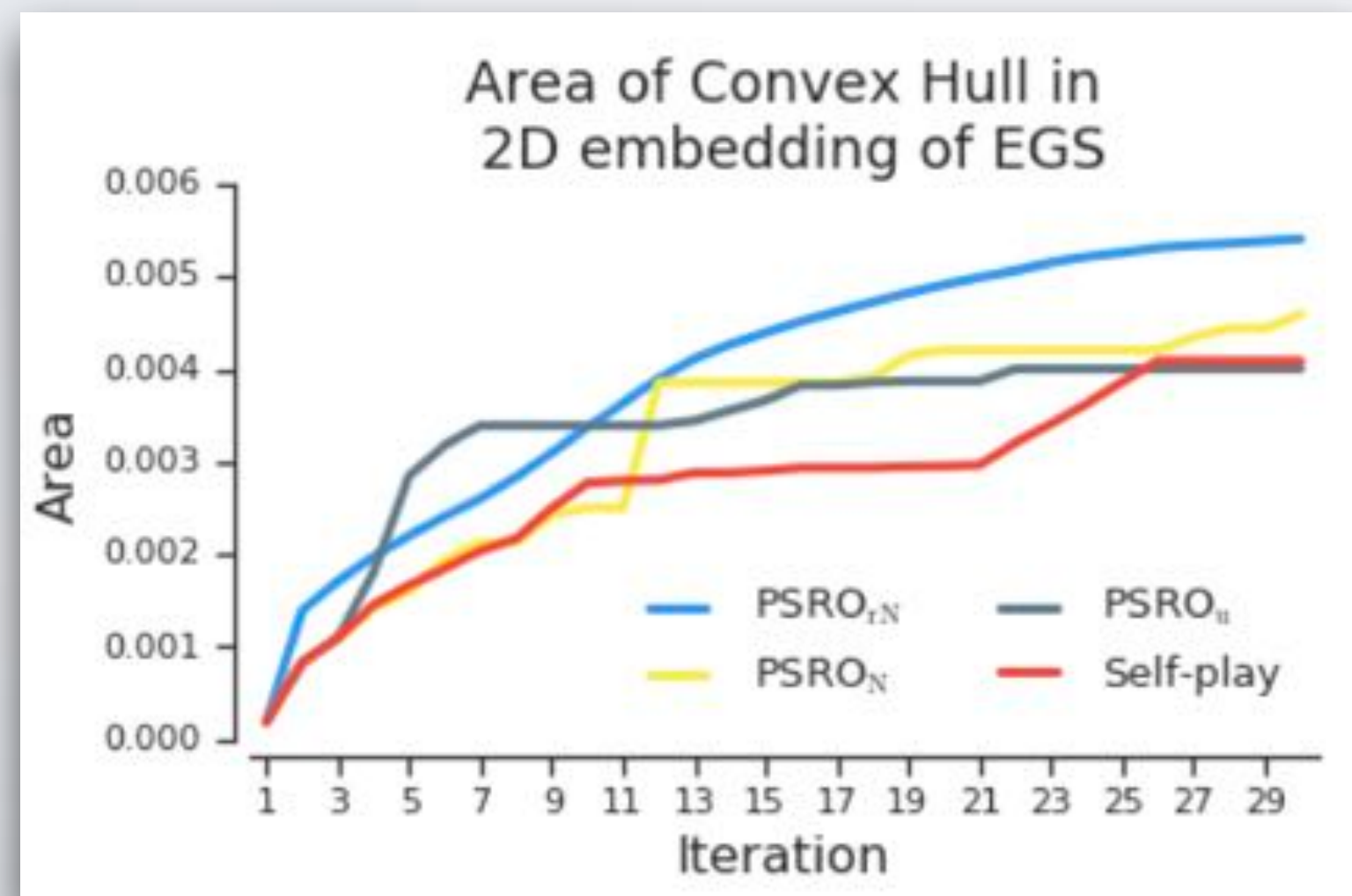
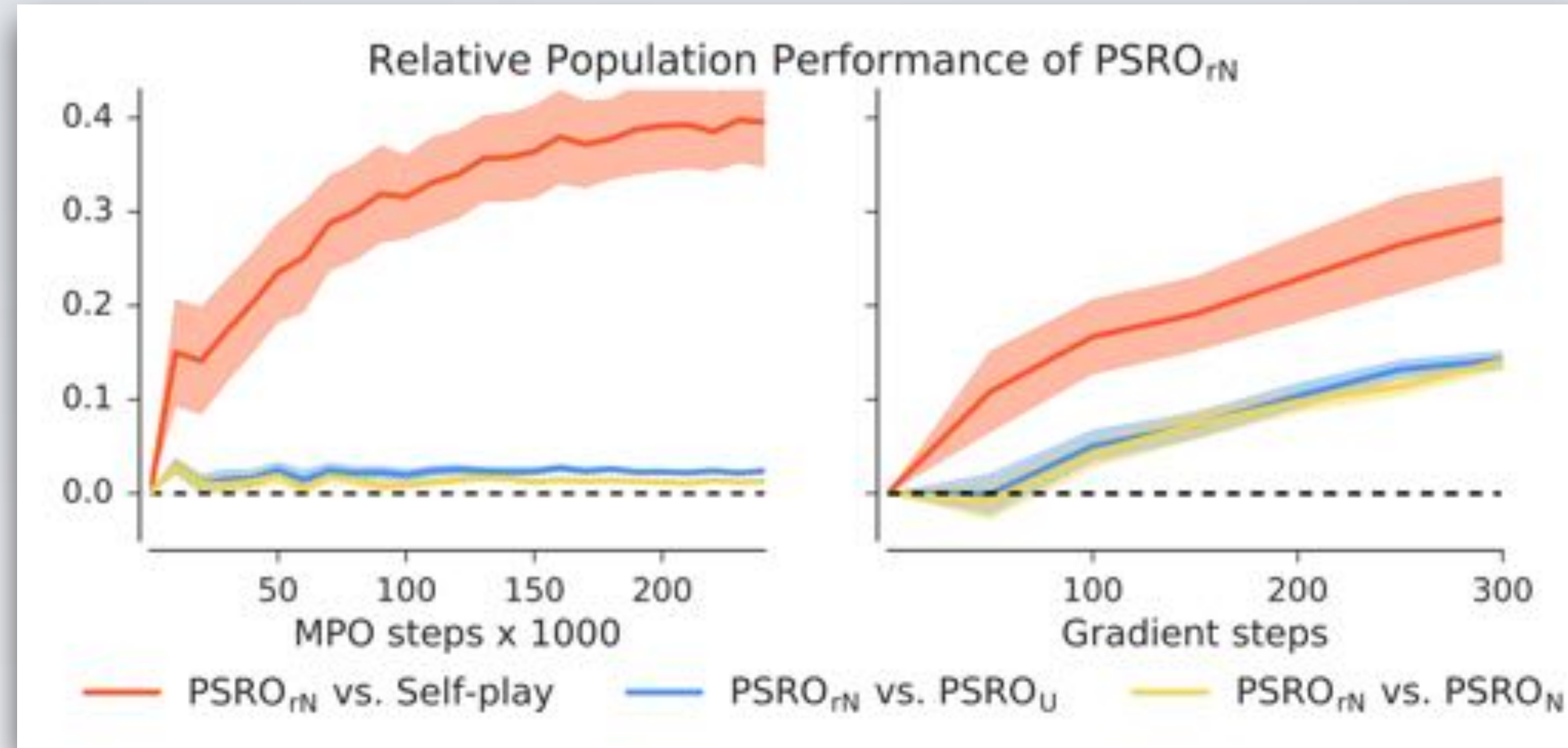
## Algorithm 4 Response to rectified Nash (PSRO<sub>rN</sub>)

```

input: population  $\mathfrak{P}_1$ 
for  $t = 1, \dots, T$  do
   $\mathbf{p}_t \leftarrow$  Nash on  $\mathbf{A}_{\mathfrak{P}_t}$ 
  for agent  $\mathbf{v}_t$  with positive mass in  $\mathbf{p}_t$  do
     $\mathbf{v}_{t+1} \leftarrow$  oracle  $(\mathbf{v}_t, \sum_{\mathbf{w}_i \in \mathfrak{P}_t} \mathbf{p}_t[i] \cdot [\phi_{\mathbf{w}_i}(\cdot)]_+)$ 
  end for
   $\mathfrak{P}_{t+1} \leftarrow \mathfrak{P}_t \cup \{\mathbf{v}_{t+1} : \text{updated above}\}$ 
end for
output:  $\mathfrak{P}_{T+1}$ 
  
```



# PSRO-rN [Balduzzi et al. 2019] - Results



**Diversity helps in exploring the strategy space more efficiently and effectively**

# Contents

- ~~Rectified Nash~~

- **Diverse-PSRO**

- Unified Behavioural + Response Diversity

- NAC

- $\alpha$ -PSRO

- Joint PSRO

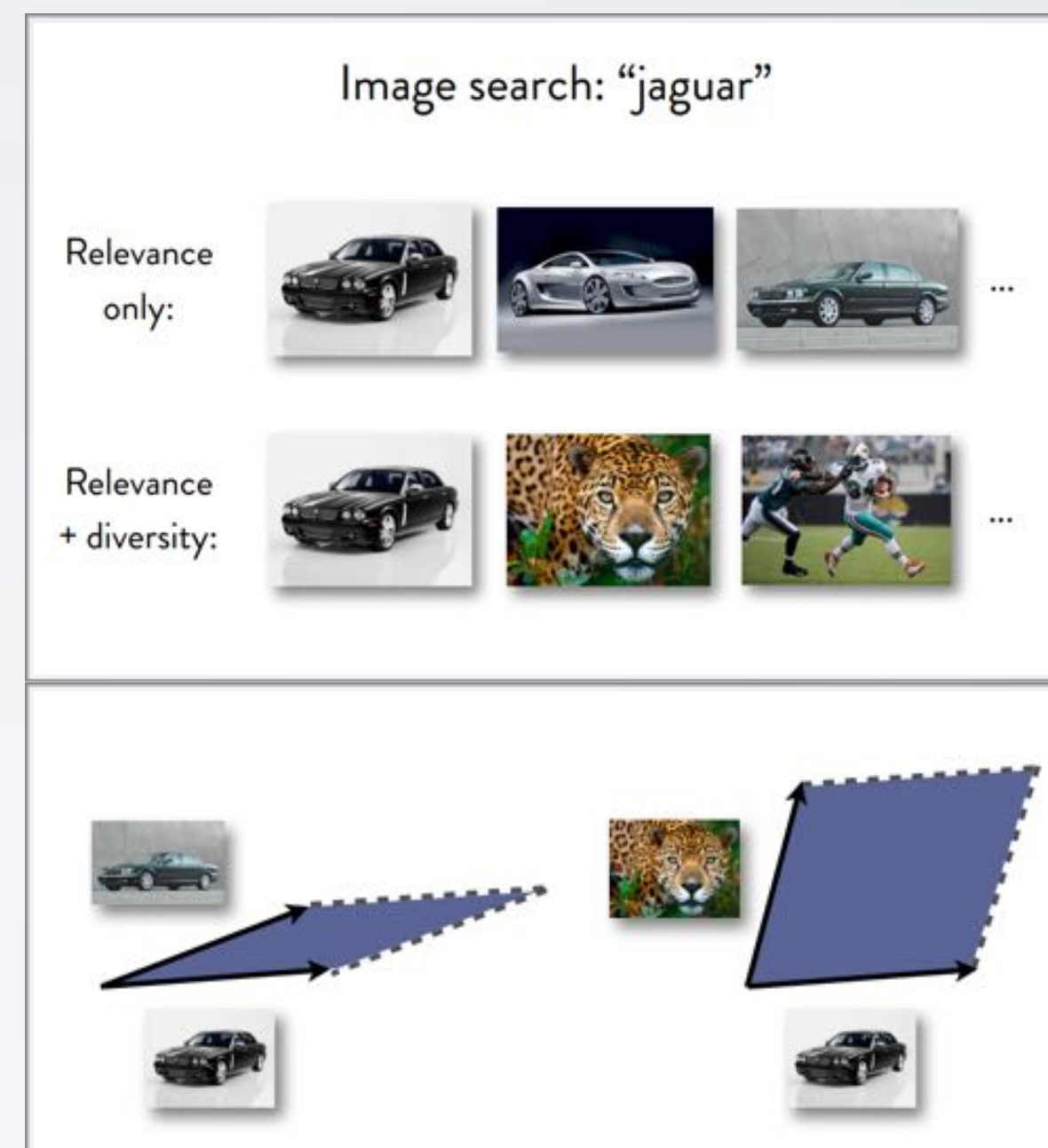
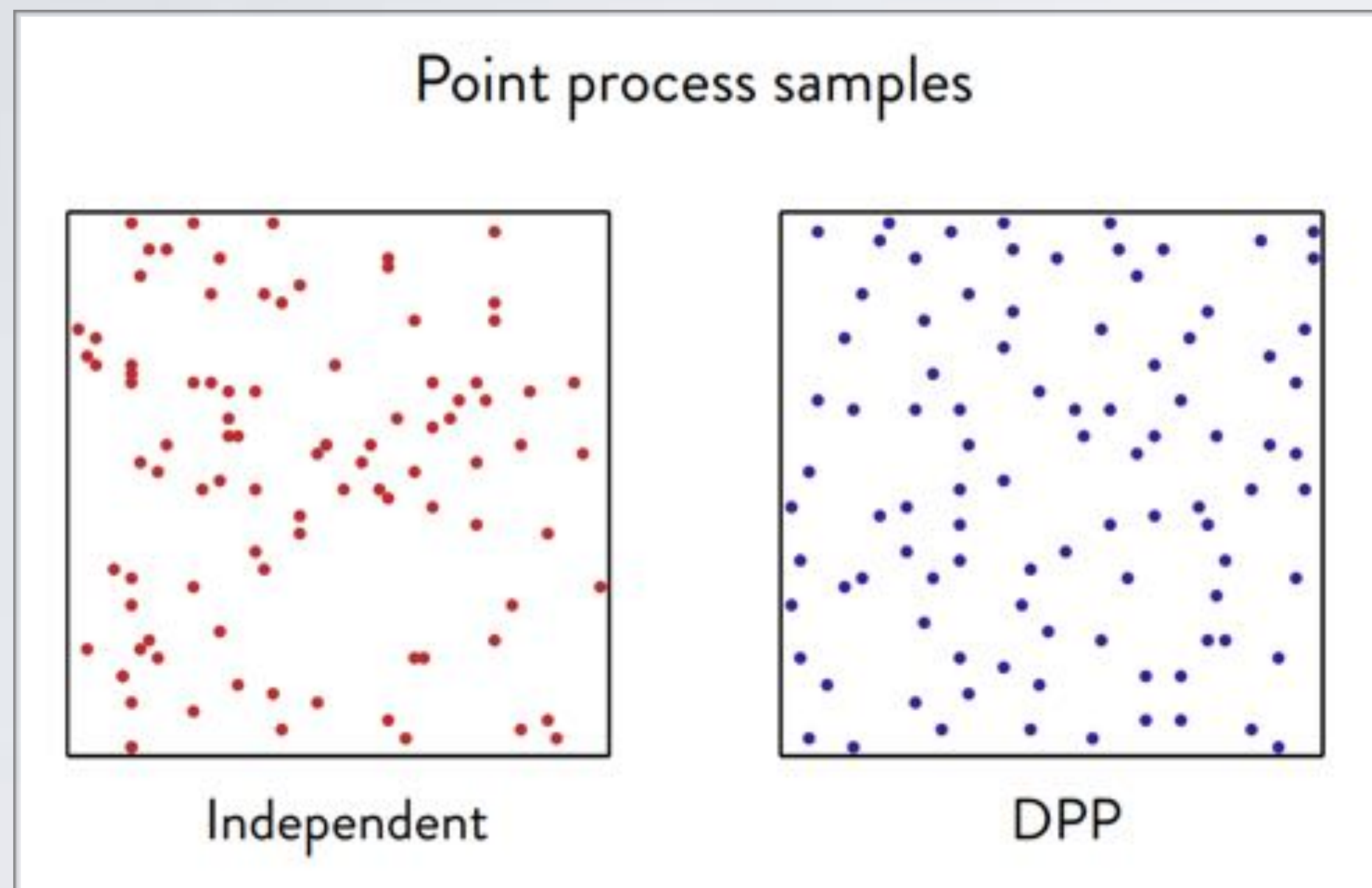
- Pipeline PSRO

- Mixed Oracles / Opponents

# Diverse-PSRO

1. Go back to first principles: **diversity should be defined in terms of orthogonality.**

- ◆ Determinantal Point Process [Alex Kulesza 2013] : a point process parameterised by a distance kernel.



Discrete point processes

- $N$  items (e.g., images or sentences):  
 $\mathcal{Y} = \{1, 2, \dots, N\}$
- $2^N$  possible subsets
- Probability measure  $\mathcal{P}$  over subsets  $Y \subseteq \mathcal{Y}$

$$\mathcal{P}(Y) \propto \det(L_Y)$$

= squared volume spanned by  $w(i), i \in Y$

$$\mathbb{P}_{\mathcal{L}}(\{i, j\}) \propto \begin{vmatrix} \mathcal{L}_{i,i} & \mathcal{L}_{i,j} \\ \mathcal{L}_{j,i} & \mathcal{L}_{j,j} \end{vmatrix} = \mathcal{L}_{i,i}\mathcal{L}_{j,j} - \mathcal{L}_{i,j}\mathcal{L}_{j,i}$$

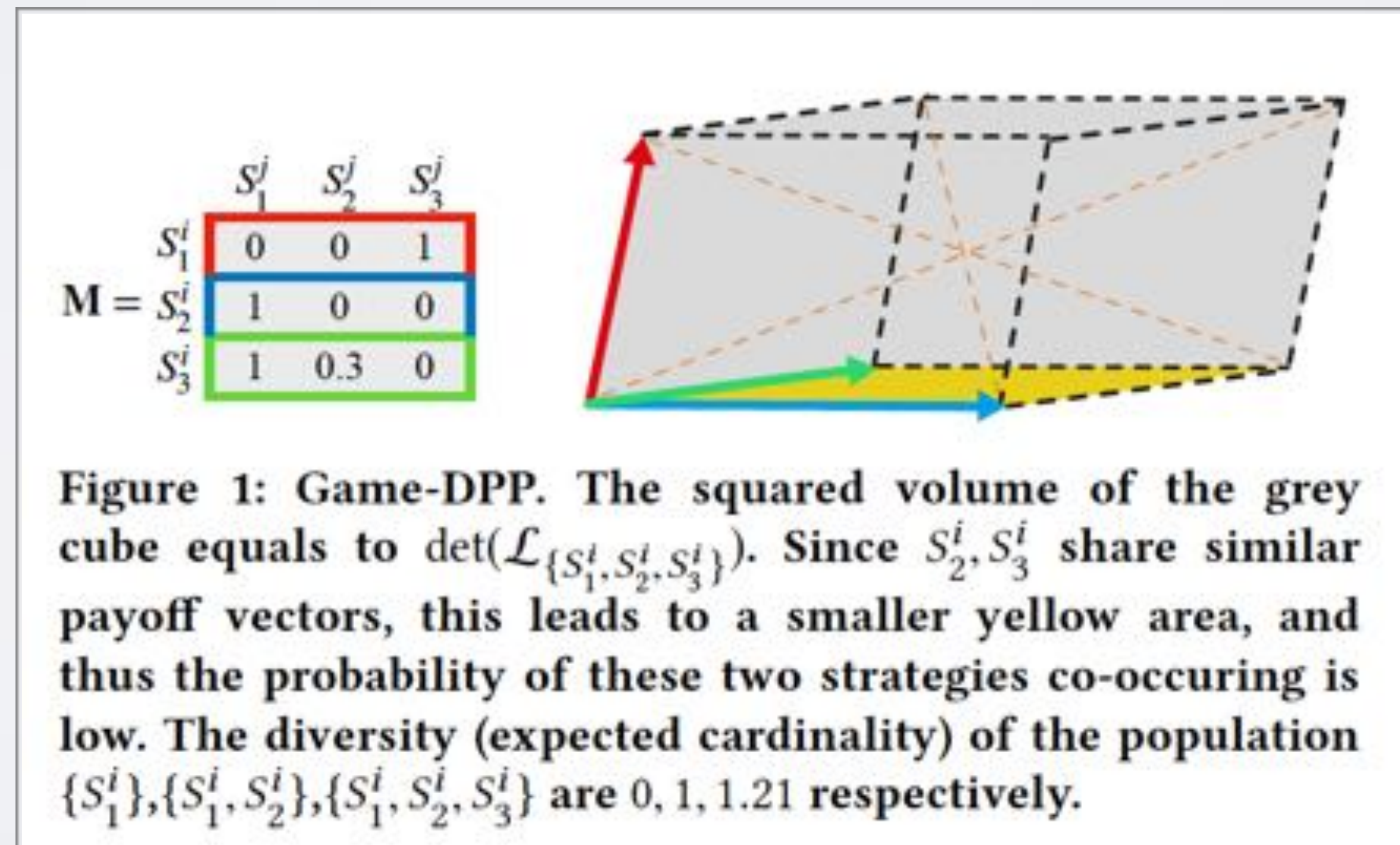
$$\text{DPP}(\mathcal{L}) := \mathbb{P}_{\mathcal{L}}(Y = Y) \propto \det(\mathcal{L}_Y) = \text{Vol}^2(\{w_i\}_{i \in Y})$$

# Diverse-PSRO

1. Go back to first principles: **diversity should be defined in terms of orthogonality.**

- ♦ Policy diversity can be measured by orthogonality of pay-off vectors, i.e.,  $\mathcal{L}_S = \mathbf{M}\mathbf{M}^\top$ .
- ♦ The expected cardinality of the DPP is the diversity metric.

$$\text{Diversity}(S) = \mathbb{E}_{Y \sim \mathbb{P}_\mathcal{L}}[|Y|] = \text{Tr} \left( \mathbf{I} - (\mathcal{L}_S + \mathbf{I})^{-1} \right)$$



# Diverse-PSRO

1. Go back to first principles: **diversity should be defined in terms of orthogonality.**

- Policy diversity can be measured by orthogonality of pay-off vectors, i.e.,  $\mathcal{L}_S = \mathbf{M}\mathbf{M}^\top$ .
- The expected cardinality of the DPP is the diversity metric.

$$\text{Diversity (S)} = \mathbb{E}_{Y \sim \mathbb{P}_L}[|Y|] = \text{Tr} \left( \mathbf{I} - (\mathcal{L}_S + \mathbf{I})^{-1} \right)$$

$$L = \begin{bmatrix} 0.0 & 0.8 & 0.1 \\ 0.9 & 0.0 & 0.7 \\ 0.9 & 0.2 & 0.7 \end{bmatrix} = \begin{bmatrix} 0.0 & 0.8 & 0.1 \\ 0.9 & 0.0 & 0.7 \\ 0.9 & 0.2 & 0.7 \end{bmatrix}$$

$$= \begin{bmatrix} 0.7 & 0.1 & 0.2 \\ 0.1 & 1.3 & 1.3 \\ 0.2 & 1.3 & 1.3 \end{bmatrix}$$

$$M = \begin{bmatrix} 0.0 & 0.8 & 0.1 \\ 0.9 & 0.0 & 0.7 \\ 0.9 & 0.2 & 0.7 \end{bmatrix} \rightarrow \mathbb{E}_{Y \sim \mathbb{P}_L}[|Y|] = 1.18$$

$$M = \begin{bmatrix} 0.0 & 0.8 & 0.1 \\ 0.9 & 0.0 & 0.7 \\ 0.9 & 0.3 & 0.7 \end{bmatrix} \rightarrow \mathbb{E}_{Y \sim \mathbb{P}_L}[|Y|] = 1.25$$

# Diverse-PSRO

- ◆ Based on the diversity metric, we can design diversity-aware PSRO

$$\text{Diversity}(\mathcal{S}) = \mathbb{E}_{Y \sim \mathbb{P}_{\mathcal{L}}} [|Y|] = \text{Tr} \left( \mathbf{I} - (\mathcal{L}_{\mathcal{S}} + \mathbf{I})^{-1} \right)$$

- ◆ Diverse PSRO

$$O^1(\pi^2) = \arg \max_{\theta \in \mathbb{R}^d} \sum_{S^2 \in \mathcal{S}^2} \pi^2(S^2) \cdot \phi(S_{\theta}, S^2) + \tau \cdot \text{Diversity}(\mathcal{S}^1 \cup \{S_{\theta}\})$$

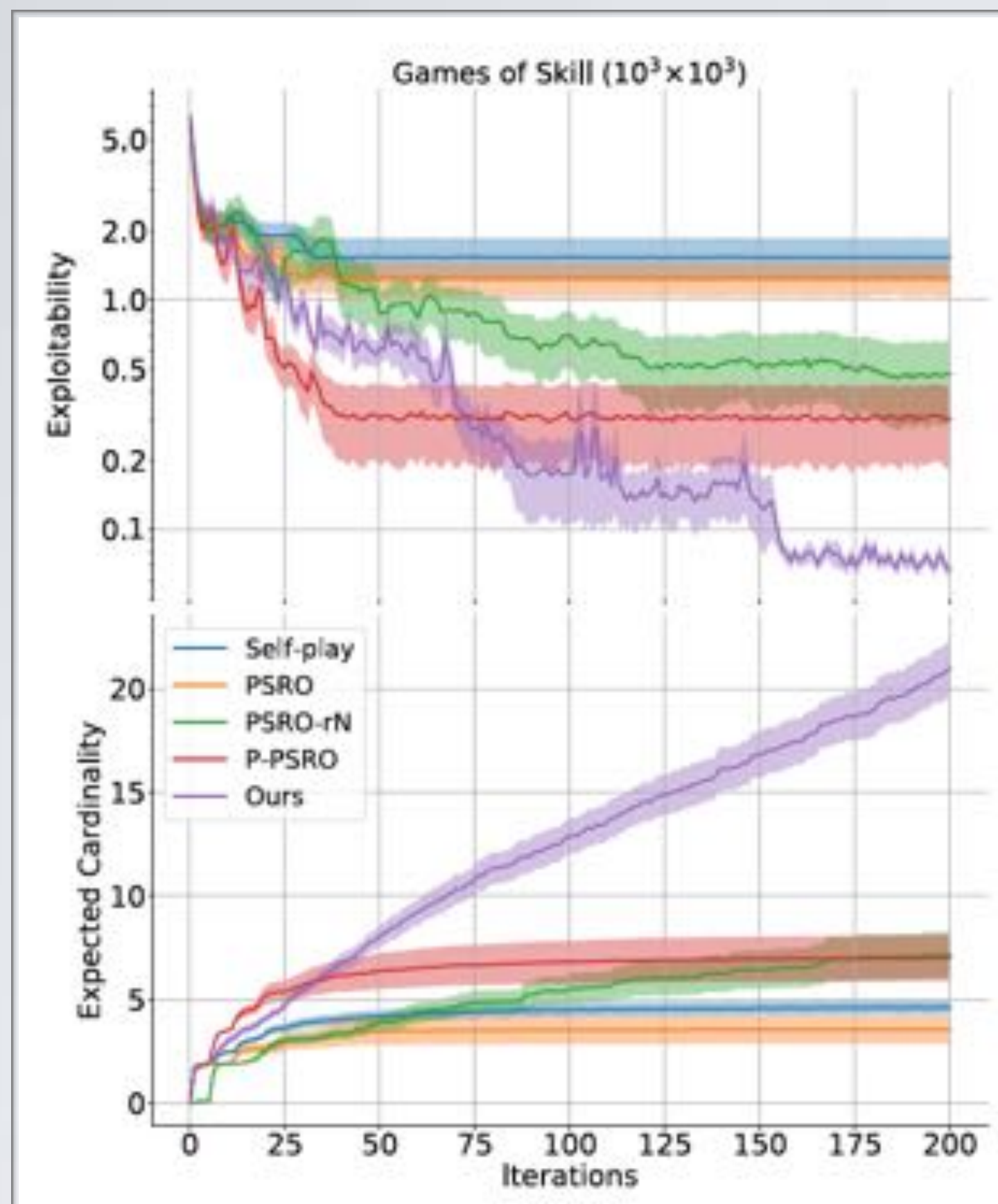
- ◆ Diverse  $\alpha$ -PSRO ( $\alpha$ -Rank as meta-solver)

$$O(\pi^2) = \arg \max_{\pi \in \Delta_{\mathcal{S}^i}} \text{Tr} \left( I - (\mathcal{L}_{\mathcal{S}^i \cup \{\pi\}} + I)^{-1} \right)$$

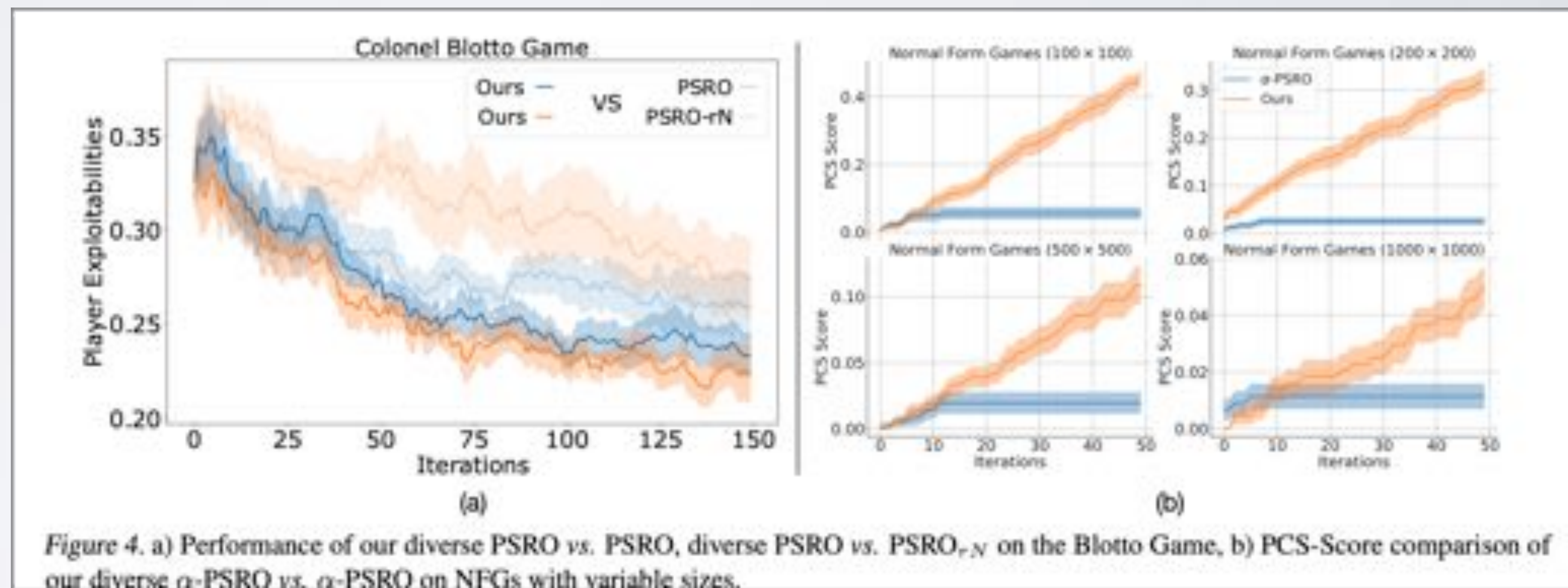
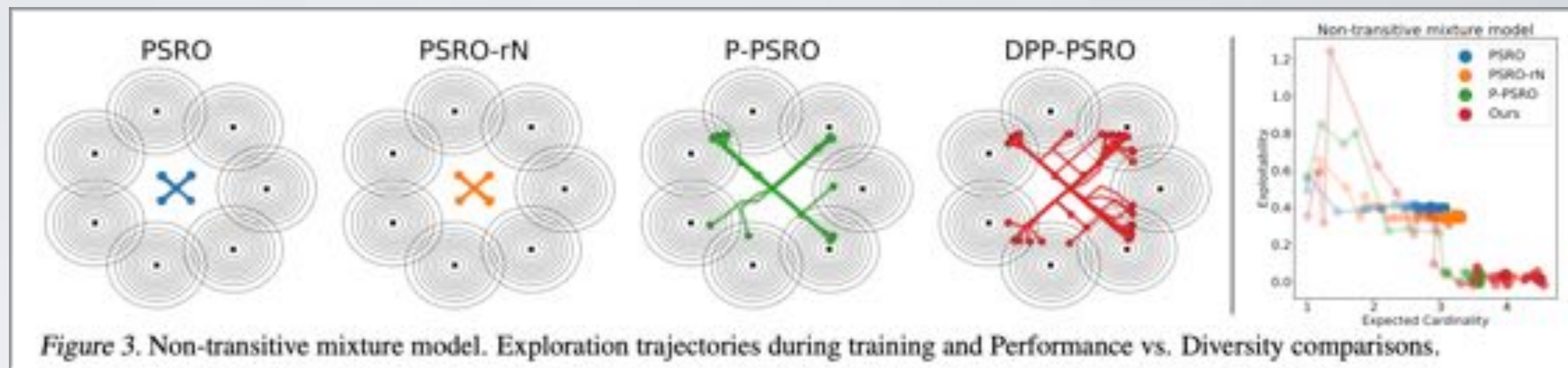
- ◆ Importantly, we prove that

$$\text{Gamescape}(\mathcal{S}) \subsetneq \text{Gamescape}(\mathcal{S} \cup \{S_{\theta}\})$$

# Diverse-PSRO



the most efficient population-based zero-sum game solver so far!



# Contents

- ~~Rectified Nash~~

- ~~Diverse-PSRO~~

- **Unified Behavioural + Response Diversity**

- NAC

- $\alpha$ -PSRO

- Joint PSRO

- Pipeline PSRO

- Mixed Oracles / Opponents



# Behavioural Diversity + Response Diversity

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

Xiangyu Liu<sup>1</sup>, Hangtian Jia<sup>2</sup>, Ying Wen<sup>1</sup>, Yaodong Yang<sup>3</sup>, Yujing Hu<sup>2</sup>,  
Yingfeng Chen<sup>2</sup>, Changjie Fan<sup>2</sup> and Zhipeng Hu<sup>2</sup>

<sup>1</sup>Shanghai Jiao Tong University, <sup>2</sup>Netease Fuxi AI Lab, <sup>3</sup>University College London

- *Rectified PSRO* & *Diverse PSRO* introduced the notion of **response diversity** (diversity of rewards)
- We want both the outcomes and the policies that lead to those outcomes to be diverse
- Diversity should include both **response diversity**, and **behavioural diversity** (diversity of the policies)

Method	Tool for Diversity	BD	RD	Game Type
DvD	Determinant	✓	×	Single-agent
PSRO <sub>N</sub>	None	×	×	n-player general-sum game
PSRO <sub>TN</sub>	$L_{1,1}$ norm	×	✓	2-player zero-sum game
DPP-PSRO	Determinantal point process	×	✓	2-player general-sum game
Our Methods	Occupancy measure & convex hull	✓	✓	n-player general-sum game

# Behavioural Diversity + Response Diversity

- **Behavioural Diversity:** Assume that we use the Nash distribution as our meta-solver,  $\pi_E = (\pi_i, \pi_{E_{-i}})$ , we want a new policy  $\pi^{M+1}$  that has a different occupancy measure  $\rho_{\pi}(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \pi)$  from  $\pi_E$ :

$$\text{Div}_{\text{occ}}(\pi_i^{M+1}) = D_f(\rho_{\pi_i^{M+1}, \pi_{E_{-i}}} \| \rho_{\pi_i, \pi_{E_{-i}}})$$

- One can train a neural network  $f_{\hat{\theta}}$  to fit  $(s, \mathbf{a}) \sim \rho_{\pi_E}$ , and then assign an intrinsic reward by encouraging the new policy to visit state-action pairs with a large prediction error (not covered by the existing occupancy measure).

$$\max R^{\text{int}}(s, a) = \left\| f_{\hat{\theta}}(s, \mathbf{a}) - f_{\theta}(s, \mathbf{a}) \right\|^2$$

# Behavioural Diversity + Response Diversity

- **Response Diversity:** we want the new policy  $\pi^{M+1}$  to expand the convex hull of the existing meta-game  $A_M$  by introducing a new payoff vector  $\mathbf{a}_{M+1} := [\phi_i(\pi_i^{M+1}, \pi_{-i}^j)]_{j=1}^N$  that

$$\text{Div}_{\text{rew}}(\pi_i^{M+1}) = \min_{\substack{\mathbf{1}^\top \boldsymbol{\beta} = 1 \\ \boldsymbol{\beta} \geq 0}} \left\| \mathbf{A}_M^\top \boldsymbol{\beta} - \mathbf{a}_{M+1} \right\|_2^2$$

- the above equation has no closed form, but we can optimise a lower bound

$$\text{Div}_{\text{rew}}(\pi_i^{M+1}) \geq \mathbf{F}(\pi_i^{M+1}) = \frac{\sigma_{\min}^2(\mathbf{A}) \left(1 - \mathbf{1}^\top (\mathbf{A}^\top)^\dagger \mathbf{a}_{n+1}\right)^2}{M} + \left\| \left(\mathbf{I} - \mathbf{A}^\top (\mathbf{A}^\top)^\dagger\right) \mathbf{a}_{n+1} \right\|_2^2$$

- However, how can we know the payoff  $\mathbf{a}_{M+1}$  before actually training the policy?

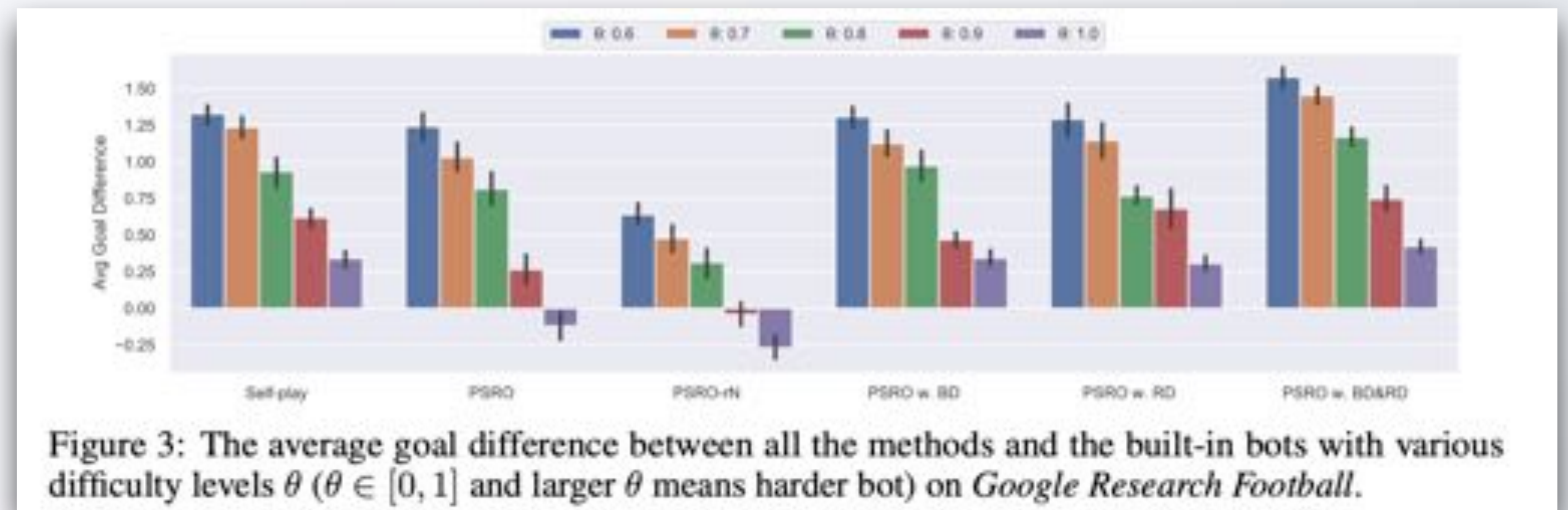
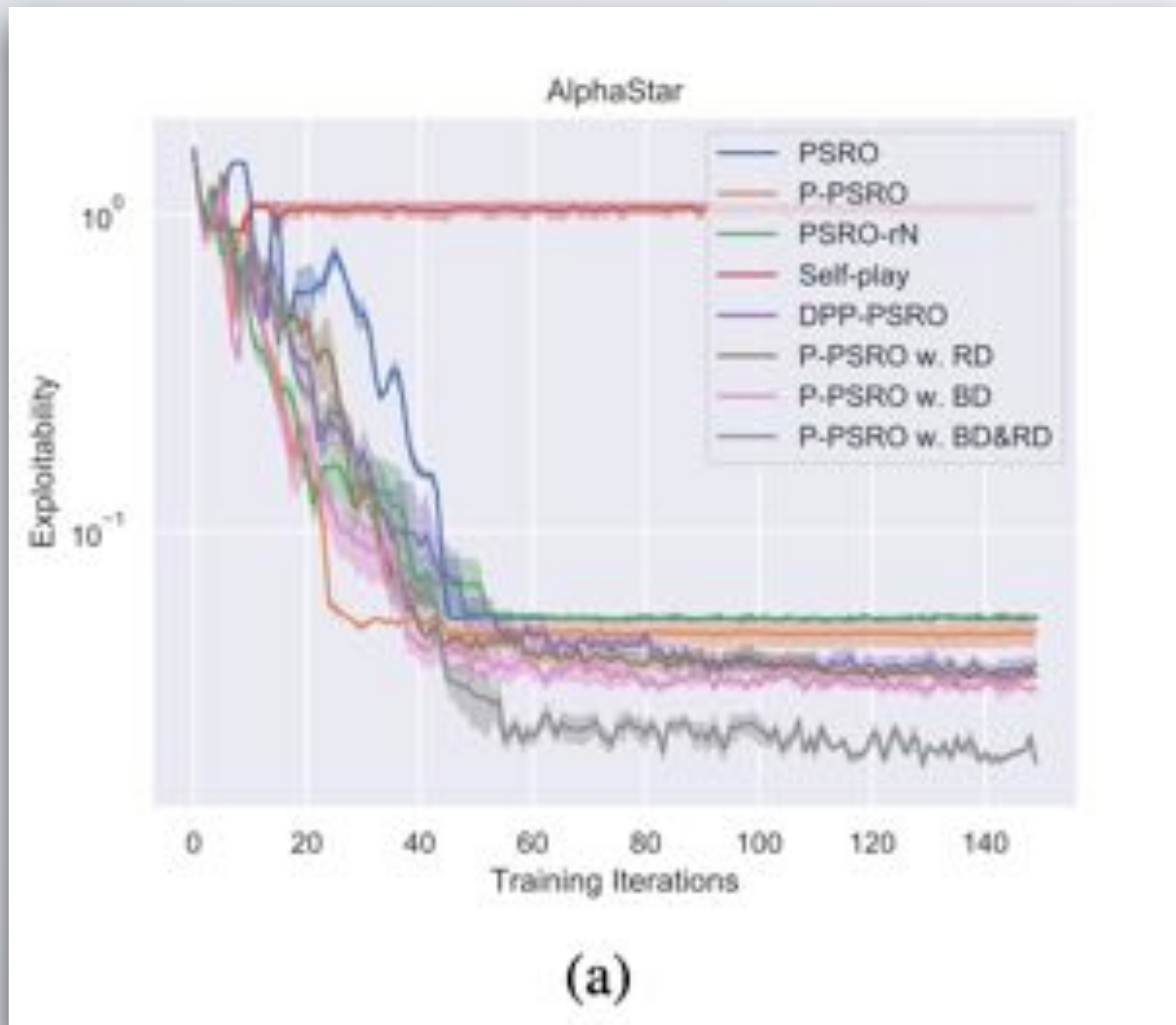
$$\frac{\partial F(\pi_i'(\theta))}{\partial \theta} = \left( \frac{\partial \phi_i(\pi_i'(\theta), \pi_{-i}^1)}{\partial \theta}, \dots, \frac{\partial \phi_i(\pi_i'(\theta), \pi_{-i}^M)}{\partial \theta} \right) \frac{\partial F}{\partial \mathbf{a}_{M+1}}$$

the answer: we can train against  $\pi_{-i}^M$  based on the weights suggested by  $\partial F / \partial \mathbf{a}_{M+1}$  !

# Behavioural Diversity + Response Diversity

- Performance when considering both Diversity terms is very impressive

$$\arg \max_{\pi'_i} \mathbb{E}_{s, \mathbf{a} \sim \rho_{\pi'_i, \pi_{E^{-i}}}} [r(s, \mathbf{a})] + \lambda_1 \text{Div}_{\text{occ}}(\pi'_i) + \lambda_2 \text{Div}_{\text{rew}}(\pi'_i)$$



# Contents

- ~~Rectified Nash~~
- ~~Diverse-PSRO~~
- ~~Unified Behavioural + Response Diversity~~
- **NAC**
- $\alpha$ -PSRO
- Joint PSRO
- Pipeline PSRO
- Mixed Oracles / Opponents

# Neural Auto-Curricula

## Discovering Multi-Agent Auto-Curricula in Two-Player Zero-Sum Games

Xidong Feng<sup>\*1</sup>, Oliver Slumbers<sup>\*1</sup>, Yaodong Yang<sup>†1</sup>,  
Ziyu Wan<sup>2</sup>, Bo Liu<sup>3</sup>, Stephen McAleer<sup>4</sup>, Ying Wen<sup>2</sup>, Jun Wang<sup>1</sup>

- Learning to learn: discover algorithm components (e.g. “who to beat” and “how to beat them”) from data.
- Is Game-Theoretic knowledge (e.g. **transitivity/non-transitivity/Nash**) needed? Learn purely from data?
- Can we learn the **auto-curricula** (i.e. the meta-solver) based on the type of game provided to the meta-learning algorithm?
- Beneficial because RL Oracles can only approximate a best-response, and using Nash may not be the best option as a meta-solver dependent on game structure & approximate best-responses.
- In single-agent RL, discovered RL methods have been shown to outperform human-designed TD learning.

## Discovering Reinforcement Learning Algorithms

Junhyuk Oh Matteo Hessel Wojciech M. Czarnecki Zhongwen Xu  
Hado van Hasselt Satinder Singh David Silver  
DeepMind

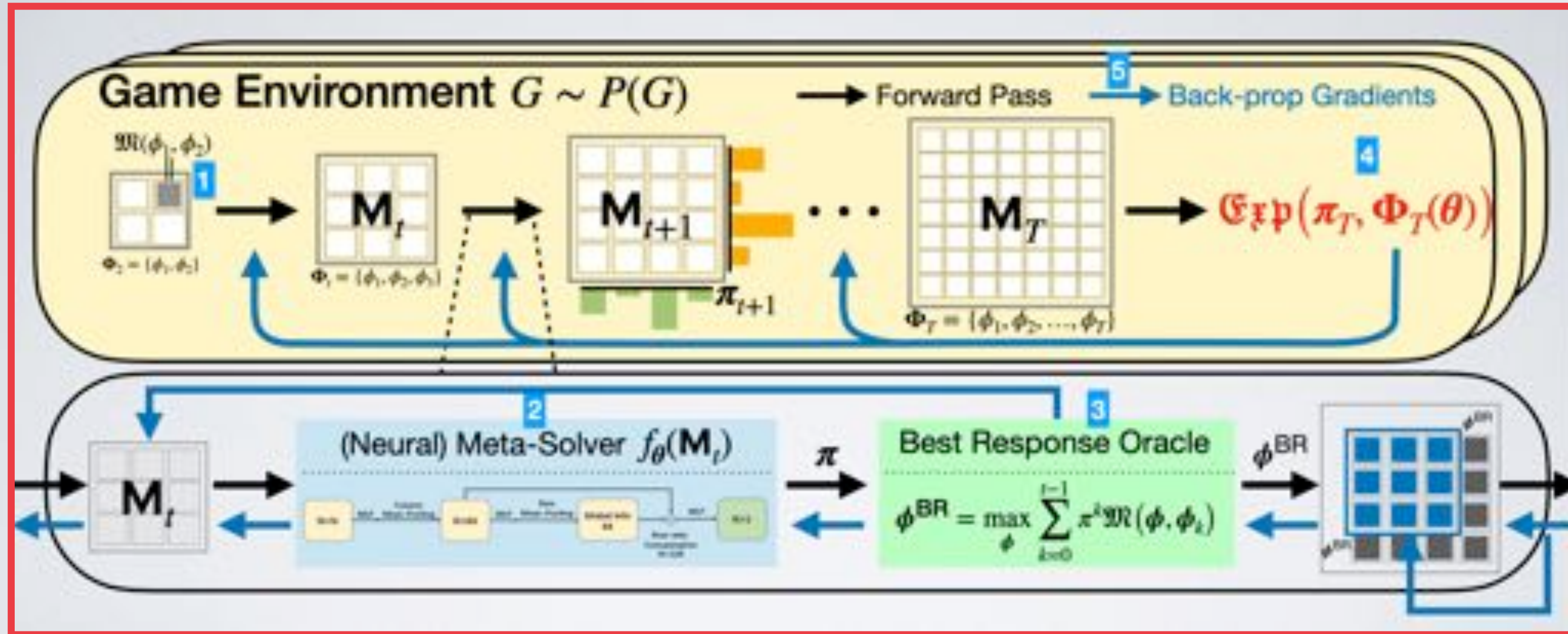
## Meta-Gradient Reinforcement Learning with an Objective Discovered Online

Zhongwen Xu, Hado van Hasselt, Matteo Hessel  
Junhyuk Oh, Satinder Singh, David Silver  
DeepMind  
{zhongwen,hado,mthss,junhyuk,baveja,davidsilver}@google.com

Algorithm	Algorithm properties	What is meta-learned?
IDBD, SMD [30, 27]	† □ →	learning rate
SGD <sup>2</sup> [1]	+++ ■ ←	optimiser
RL <sup>2</sup> , Meta-RL [9, 39]	+++ ■ X	recurrent network
MAML, REPTILE [11, 23]	+++ □ ←	initial params
Meta-Gradient [43, 46]	† □ →	$\gamma, \lambda$ , reward
Meta-Gradient [38, 44, 40]	† □ ←	auxiliary tasks, hyperparams, reward weights
ML <sup>3</sup> , MetaGenRL [2, 19]	+++ ■ ←	loss function
Evolved PG [16]	+++ ■ X	loss function
Oh et al. 2020 [24]	+++ ■ ←	target vector
This paper	† ■ ←	target

□ white box, ■ black box, † single lifetime, +++ multi-lifetime  
← backward mode, → forward mode, X no meta-gradient

# Neural Auto-Curricula Framework



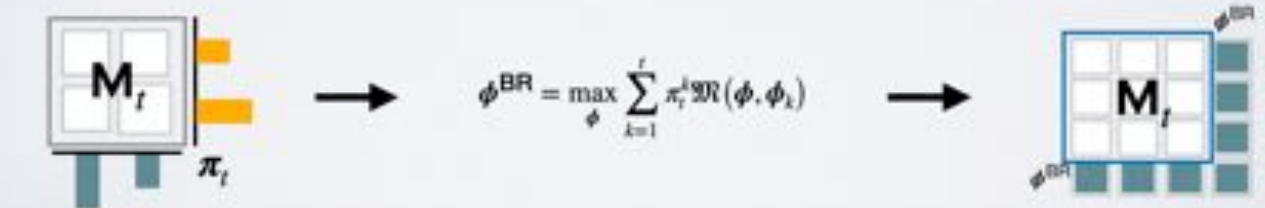
## 3 The Best-Response Oracle

Algorithm component that controls the iterative expansion of the population

Given a curriculum  $\pi_t \in \Delta_{|\Phi_t|}$  the goal becomes to solve a best-response to this distribution

Goal is the following:  $\phi_t^{BR} = \operatorname{argmax}_{\phi} \sum_{k=1}^t \pi_t^k \mathfrak{R}(\phi, \phi_k)$

Perform the optimisation in anyway desired, but this will impact the meta-gradient calculation



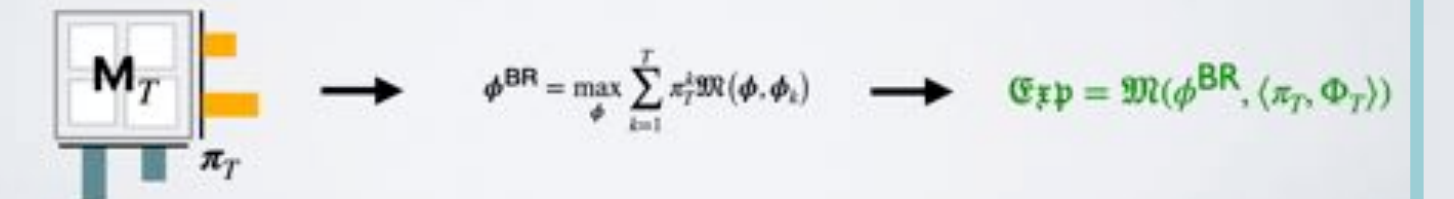
## 4 The Learning Objective

What is the goal of the iterative update procedure?

Given a curriculum  $\pi_T = f_{\theta}(M_T)$  and a population  $\Phi_T$  we want to be as close to a Nash equilibrium as possible.

Distance to Nash measured as the exploitability:  $\mathfrak{Exp} := \max_{\phi} \mathfrak{R}(\phi, \langle \pi_T, \Phi_T \rangle)$

i.e. How good is the best-response to the curriculum? If 0, it is a Nash equilibrium



## 1 The Meta-Game

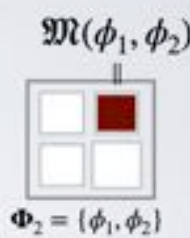
Main component of population-based methods - **The meta-game**

An agent is a mapping  $\phi : S \times A \rightarrow [0,1]$

The payoff for agent  $i$  vs. agent  $j$  is defined as  $\mathfrak{R}(\phi_i, \phi_j)$

Payoff matrix between agents in a population amenable to GT analysis

The goal of these algorithms is to expand the populations  $\Phi$  iteratively



## 2 The Meta-Solver

Algorithm component that controls the auto-curricula of who to compete with

General examples: Nash equilibrium, Uniform distribution, Last agent

Need to parameterise the process so that we can learn it

A network with parameters  $\theta$  maps  $f_{\theta} : M_t \rightarrow [0,1]^t$  so that  $\pi_t = f_{\theta}(M_t)$



## 5 Optimisation through meta-gradients

Recall the learning objective of the player:  $\mathfrak{Exp} := \max_{\phi} \mathfrak{R}(\phi, \langle \pi_T, \Phi_T \rangle)$

Also recall that  $\pi_T = f_{\theta}(M_T)$ , which allows us to define the meta-solver optimisation as:

$$\theta^* = \operatorname{argmin}_{\theta} J(\theta), \text{ where } J(\theta) = \mathbb{E}_{G \sim P(G)} [\mathfrak{Exp}(\pi, \Phi | \theta, G)]$$

What does the gradient boil down to then?

$$\nabla_{\theta} J(\theta) = \mathbb{E}_G \left[ \frac{\partial \mathfrak{R}_{T+1}}{\partial \phi_{T+1}^{BR}} \frac{\partial \phi_{T+1}^{BR}}{\partial \theta} + \frac{\partial \mathfrak{R}_{T+1}}{\partial \pi_T} \frac{\partial \pi_T}{\partial \theta} + \frac{\partial \mathfrak{R}_{T+1}}{\partial \Phi_T} \frac{\partial \Phi_T}{\partial \theta} \right]$$

Gradient of most interest decomposes to

$$\frac{\partial \phi_{T+1}^{BR}}{\partial \theta} = \frac{\partial \phi_{T+1}^{BR}}{\partial \pi_T} \frac{\partial \pi_T}{\partial \theta} + \frac{\partial \phi_{T+1}^{BR}}{\partial \Phi_T} \frac{\partial \Phi_T}{\partial \theta}$$

# 1 The Meta-Game

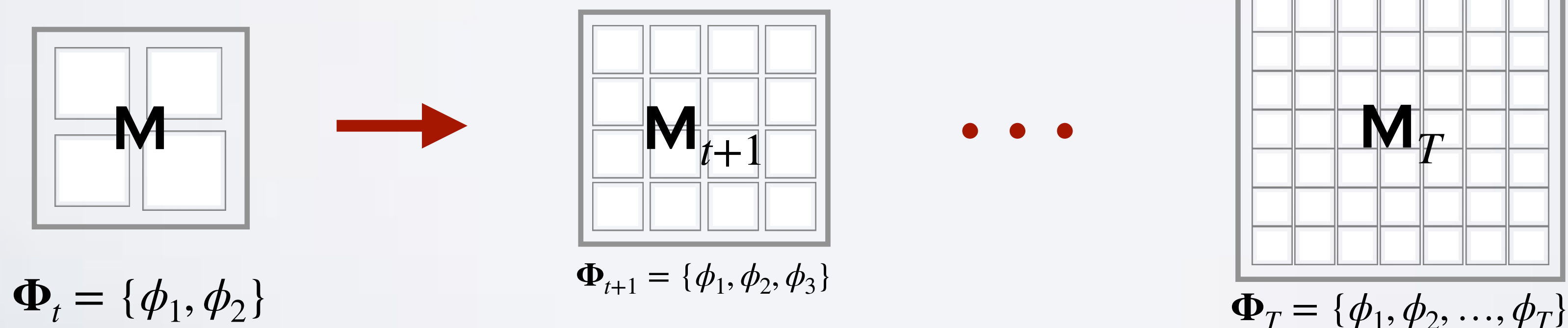
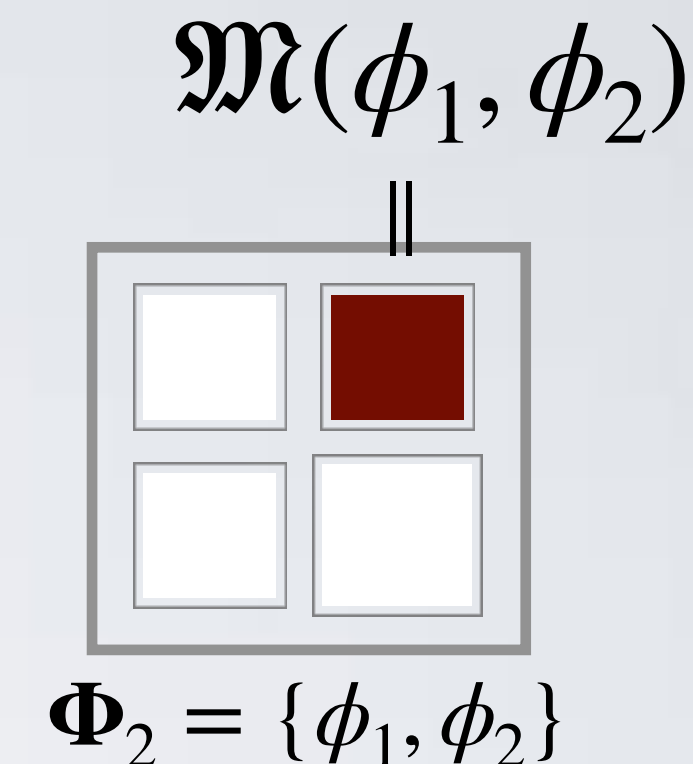
- Main component of population-based methods - **The meta-game**

- An agent is a mapping  $\phi : \mathcal{S} \times \mathcal{A} \rightarrow [0,1]$

- The payoff for agent  $i$  vs. agent  $j$  is defined as  $\mathfrak{M}(\phi_i, \phi_j)$

- Payoff matrix between *agents* in a population amenable to GT analysis

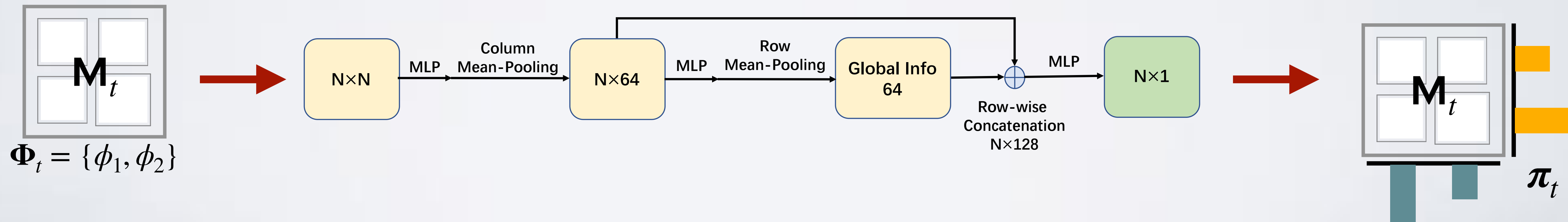
- The goal of these algorithms is to expand the populations  $\Phi$  iteratively





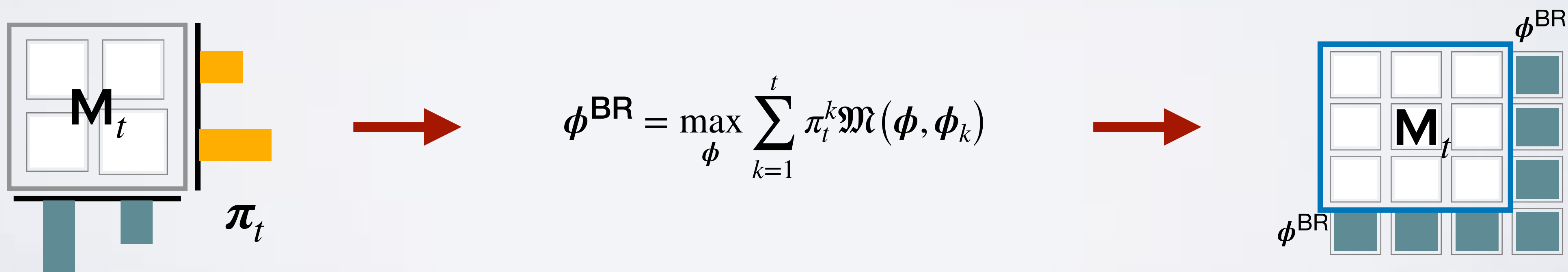
## 2 The Meta-Solver

- Algorithm component that controls the auto-curricula of *who to compete with*
  - General examples: Nash equilibrium, Uniform distribution, Last agent
  - Need to parameterise the process so that we can learn it
    - A network with parameters  $\theta$  maps  $f_\theta : \mathbf{M}_t \rightarrow [0,1]^t$  so that  $\pi_t = f_\theta(\mathbf{M}_t)$



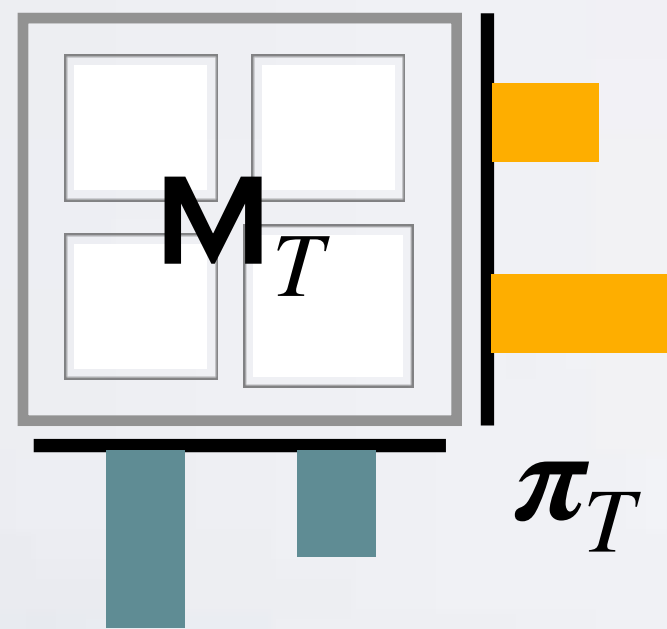
### 3 The Best-Response Oracle

- Algorithm component that controls the iterative expansion of the population
  - Given a curriculum  $\pi_t \in \Delta_{|\Phi_t|}$  the goal becomes to solve a best-response to this distribution
  - Goal is the following:
$$\phi_t^{\text{BR}} = \operatorname{argmax}_{\phi} \sum_{k=1}^t \pi_t^k \mathfrak{M}(\phi, \phi_k)$$
  - Perform the optimisation in anyway desired, but this will impact the meta-gradient calculation



## 4 The Learning Objective

- What is the goal of the iterative update procedure?
  - Given a curriculum  $\pi_T = f_\theta(\mathbf{M}_T)$  and a population  $\Phi_T$  we want to be as close to a Nash equilibrium as possible.
  - Distance to Nash measured as the *exploitability*:
$$\mathfrak{Exp} := \max_{\phi} \mathfrak{M}(\phi, \langle \pi_T, \Phi_T \rangle)$$
    - i.e. How good is the best-response to the curriculum? If 0, it is a Nash equilibrium



$$\phi^{\text{BR}} = \max_{\phi} \sum_{k=1}^T \pi_T^k \mathfrak{M}(\phi, \phi_k)$$

$$\mathfrak{Exp} = \mathfrak{M}(\phi^{\text{BR}}, \langle \pi_T, \Phi_T \rangle)$$

## 5 Optimisation through meta-gradients

• Recall the learning objective of the player:  $\mathfrak{Exp} := \max_{\phi} \mathfrak{M}(\phi, \langle \pi_T, \Phi_T \rangle)$

• Also recall that  $\pi_T = f_{\theta}(\mathbf{M}_T)$ , which allows us to define the meta-solver optimisation as:

$$\theta^* = \operatorname{argmin}_{\theta} J(\theta), \text{ where } J(\theta) = \mathbb{E}_{G \sim P(G)} \left[ \mathfrak{Exp}(\pi, \Phi \mid \theta, G) \right]$$

• What does the gradient boil down to then?

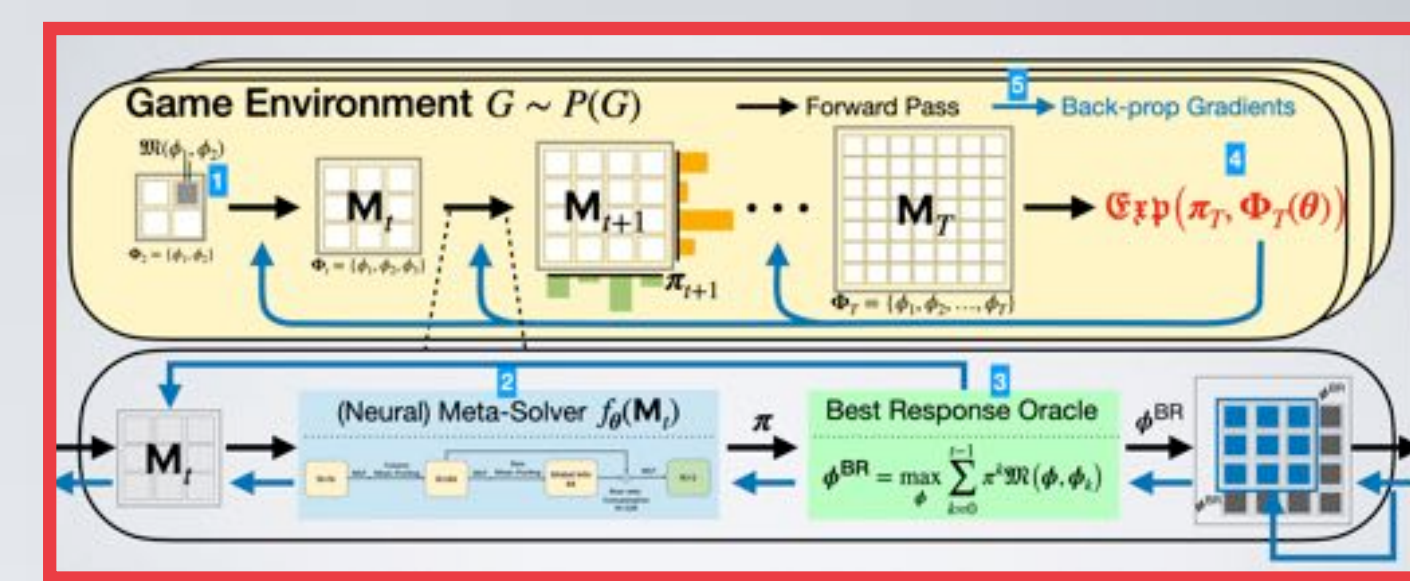
$$\nabla_{\theta} J(\theta) = \mathbb{E}_G \left[ \frac{\partial \mathfrak{M}_{T+1}}{\partial \phi_{T+1}^{\text{BR}}} \frac{\partial \phi_{T+1}^{\text{BR}}}{\partial \theta} + \frac{\partial \mathfrak{M}_{T+1}}{\partial \pi_T} \frac{\partial \pi_T}{\partial \theta} + \frac{\partial \mathfrak{M}_{T+1}}{\partial \Phi_T} \frac{\partial \Phi_T}{\partial \theta} \right]$$

Gradient of most interest decomposes to

$$\frac{\partial \phi_{T+1}^{\text{BR}}}{\partial \theta} = \frac{\partial \phi_{T+1}^{\text{BR}}}{\partial \pi_T} \frac{\partial \pi_T}{\partial \theta} + \frac{\partial \phi_{T+1}^{\text{BR}}}{\partial \Phi_T} \frac{\partial \Phi_T}{\partial \theta}$$

# Neural Auto-Curricula Recap

- The objective is given by:



The goal of LMAC is to find an auto-curricula that after  $T$  best-response iterations returns a meta-strategy and population,  $\langle \pi_T, \Phi_T \rangle$ , that helps minimise the exploitability, written as:

$$\min_{\theta} \mathcal{E}_{xp}(\pi_T(\theta), \Phi_T(\theta)), \text{ where } \mathcal{E}_{xp} := \max_{\phi} \mathfrak{M}(\phi, \langle \pi_T, \Phi_T \rangle), \quad (3)$$

$$\pi_T = f_{\theta}(\mathbf{M}_T), \Phi_T = \{\phi_T^{\text{BR}}(\theta), \phi_{T-1}^{\text{BR}}(\theta), \dots, \phi_1^{\text{BR}}(\theta)\}. \quad (4)$$

Based on the *Player's* learning objectives in Eq. (3), we can optimise the meta-solver as follows:

$$\theta^* = \arg \min_{\theta} J(\theta), \text{ where } J(\theta) = \mathbb{E}_{G \sim P(G)} [\mathcal{E}_{xp}(\pi, \Phi | \theta, G)]. \quad (5)$$

- When optimising the meta-solver  $\theta$ , **the type of best-response oracle** matters due to back-propagation!

- one-step gradient descent oracle 
$$\phi_{t+1}^{\text{BR}} = \phi_0 + \alpha \frac{\partial \mathfrak{M}(\phi_0, \langle \pi_t, \Phi_t \rangle)}{\partial \phi_0}, \frac{\partial \phi_{t+1}^{\text{BR}}}{\partial \pi_t} = \alpha \frac{\partial^2 \mathfrak{M}(\phi_0, \langle \pi_t, \Phi_t \rangle)}{\partial \phi_0 \partial \pi_t}, \frac{\partial \phi_{t+1}^{\text{BR}}}{\partial \Phi_t} = \alpha \frac{\partial^2 \mathfrak{M}(\phi_0, \langle \pi_t, \Phi_t \rangle)}{\partial \phi_0 \partial \Phi_t}.$$

- N-step gradient descent oracle (via **implicit gradient**) 
$$\frac{\partial \phi_{t+1}^{\text{BR}}}{\partial \Phi_t} = - \left[ \frac{\partial^2 \mathfrak{M}(\phi_{t+1}^{\text{BR}}, \langle \pi_t, \Phi_t \rangle)}{\partial \phi_{t+1}^{\text{BR}} \partial \phi_{t+1}^{\text{BR}T}} \right]^{-1} \frac{\partial^2 \mathfrak{M}(\phi_{t+1}^{\text{BR}}, \langle \pi_t, \Phi_t \rangle)}{\partial \phi_{t+1}^{\text{BR}} \partial \Phi_t}$$

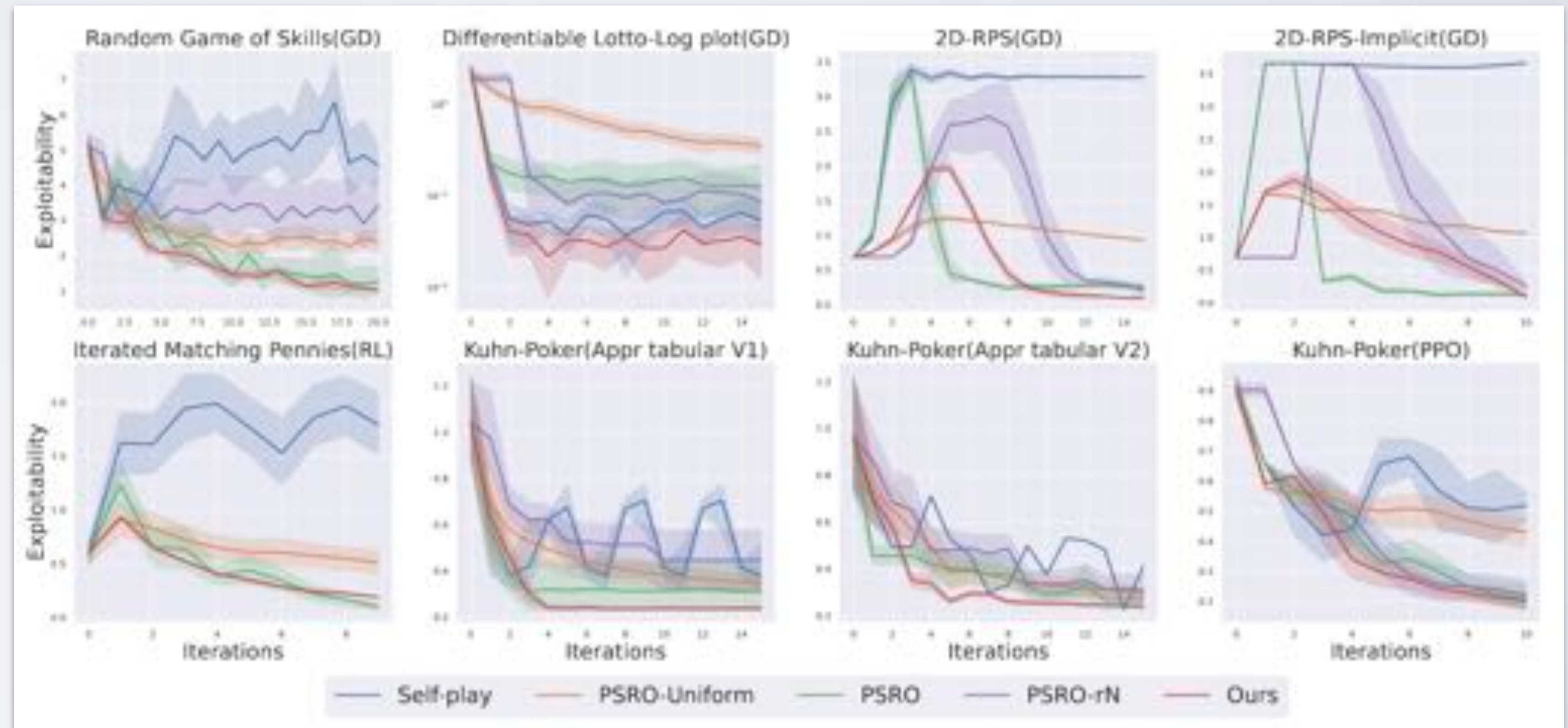
- policy-gradient based oracle (via **DICE**) 
$$\phi_1 = \phi_0 + \alpha \frac{\partial \mathcal{J}^{\text{DICE}}}{\partial \phi_0}, \text{ where } \mathcal{J}^{\text{DICE}} = \sum_{k=0}^{H-1} \left( \prod_{k'=0}^k \frac{\pi_{\phi_1}(a_{k'}^1 | s_{k'}^1) \pi_{\phi_2}(a_{k'}^2 | s_{k'}^2)}{\perp (\pi_{\phi_1}(a_{k'}^1 | s_{k'}^1) \pi_{\phi_2}(a_{k'}^2 | s_{k'}^2))} \right) r_k^1$$

- general type of oracle (via **ES**) 
$$\nabla_{\theta} \hat{J}_{\sigma}(\theta) = \mathbb{E}_{G \sim P(G), \epsilon \sim \mathcal{N}(0, I)} \left[ \frac{1}{\sigma} (\mathcal{E}_{xp}_T(\pi_T, \Phi_T) | \theta + \epsilon, G) \epsilon \right]$$

# Neural Auto-Curricula Results

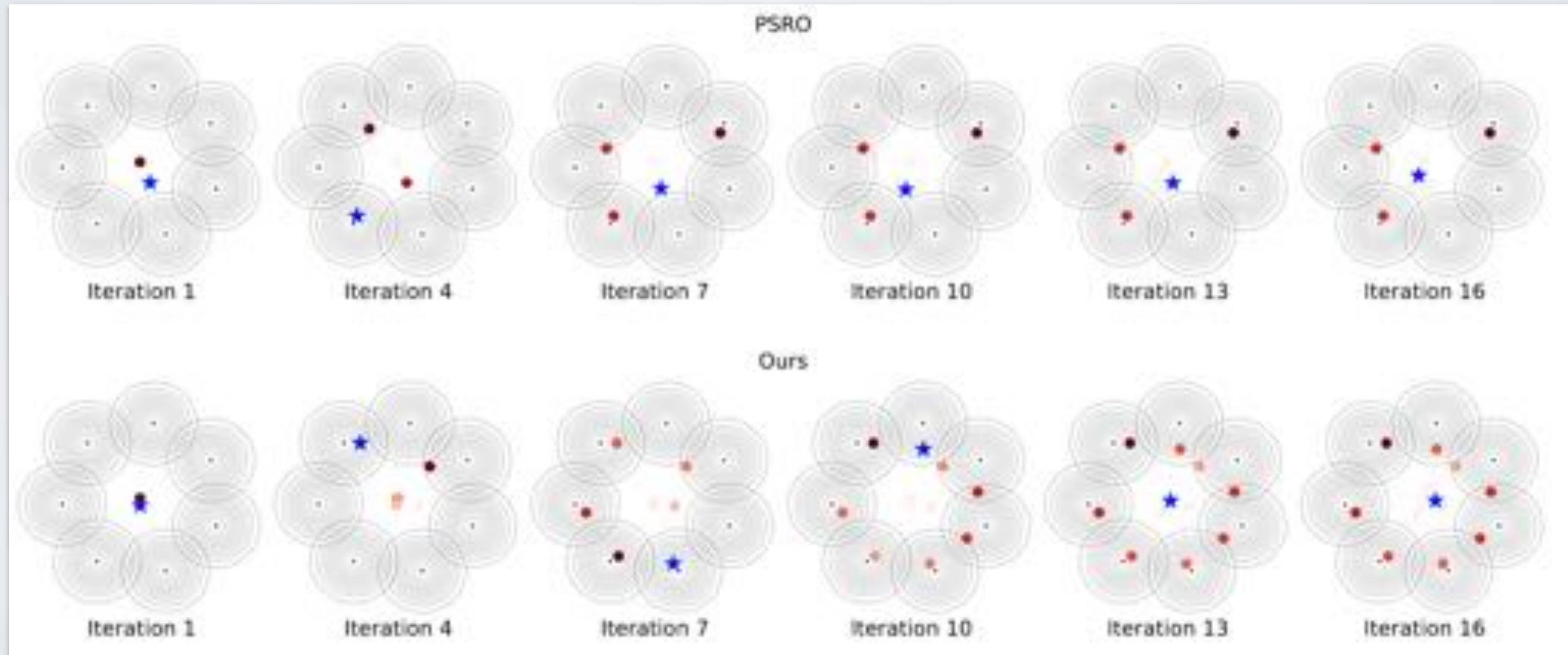
- Is our method any good on the environments where it is trained?
  - Due to long-trajectory issues, we also focus on the **approximate** best-response setting

- Performance *at least* as good as baseline measures
- Outperforms PSRO in multiple settings



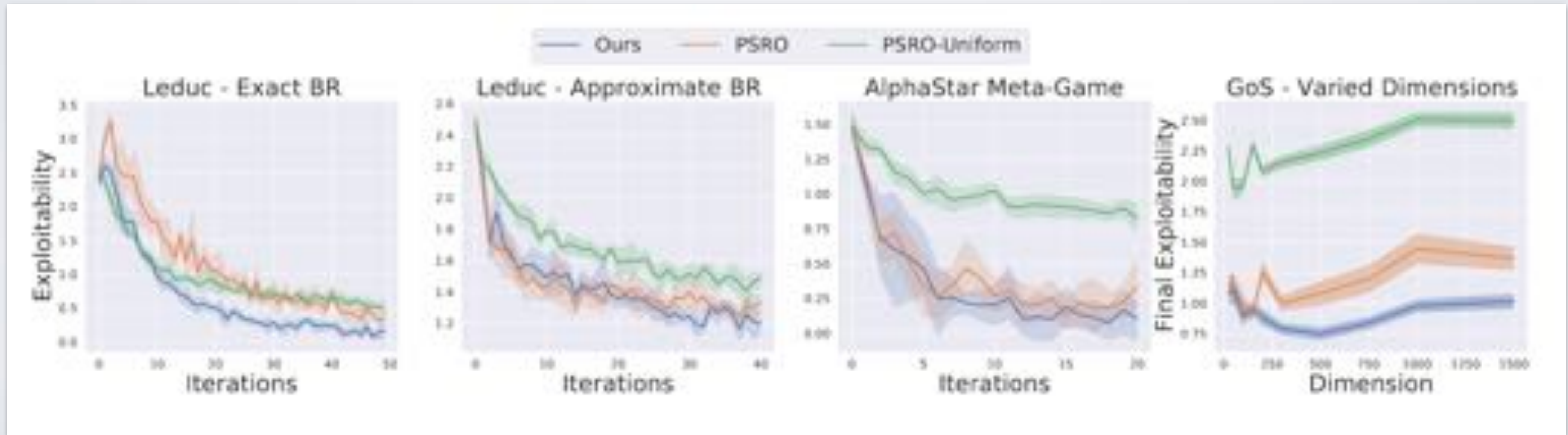
# Neural Auto-Curricula Results

- What is the learned **auto-curricula**?
- Compare agents found and their respective densities in the meta-distribution



# Neural Auto-Curricula Results

- Can the learned solver **generalise** over different games?
  - the most promising and striking aspect of NAC - Train on small games and generalise to large games, e.g., train on Kuhn Poker and test on Leduc Poker





# Contents

● ~~Rectified Nash~~

● ~~Diverse-PSRO~~

● ~~Unified Behavioural + Response Diversity~~

● ~~NAC~~

●  $\alpha$ -PSRO

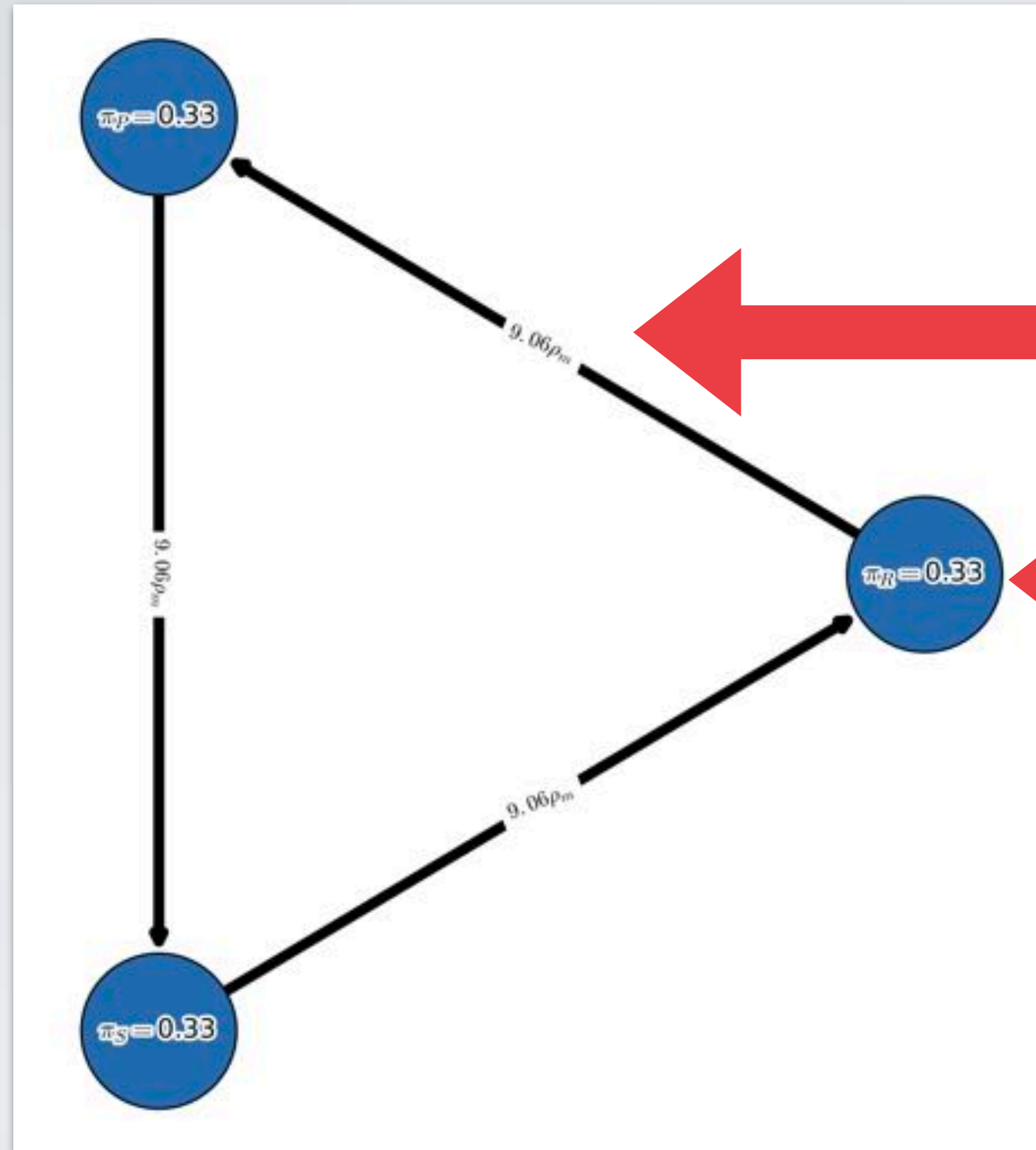
● Joint PSRO

● Pipeline PSRO

● Mixed Oracles / Opponents

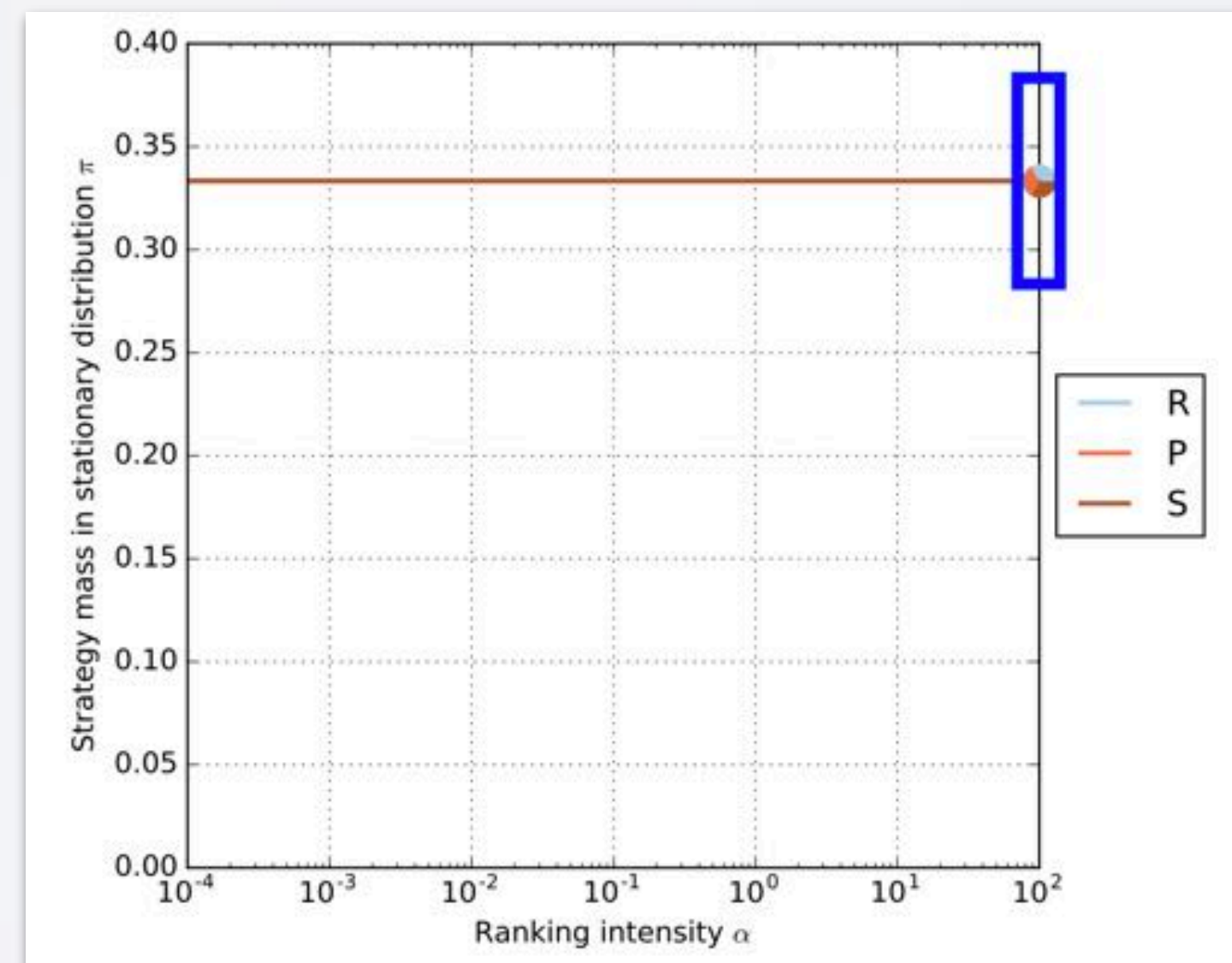
# $\alpha$ -Rank

## Applies to K-Player General-Sum Games



Edge direction corresponds to flow of population from one action to the other

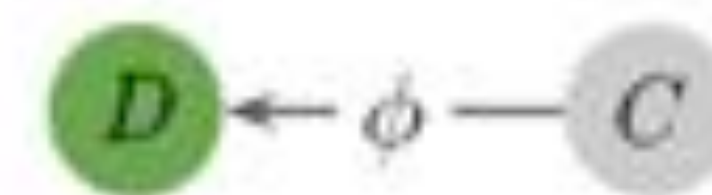
Amount of time a population spends as each action



Agent	Rank	Score
<i>R</i>	1	0.33
<i>P</i>	1	0.33
<i>S</i>	1	0.33

# $\alpha$ -PSRO [Muller et al. 2020] - Example

		Player 2				
		<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>X</i>
Player 1	<i>A</i>	0	$-\phi$	1	$\phi$	$-\epsilon$
	<i>B</i>	$\phi$	0	$-\phi^2$	1	$-\epsilon$
	<i>C</i>	-1	$\phi^2$	0	$-\phi$	$-\epsilon$
	<i>D</i>	$-\phi$	-1	$\phi$	0	$-\epsilon$
	<i>X</i>	$\epsilon$	$\epsilon$	$\epsilon$	$\epsilon$	0

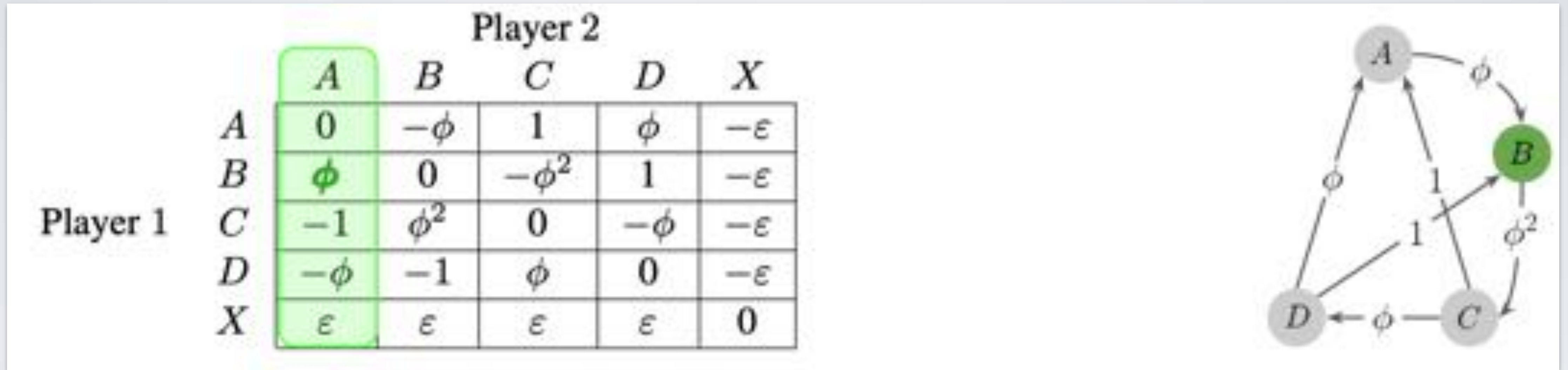


# $\alpha$ -PSRO [Muller et al. 2020] - Example

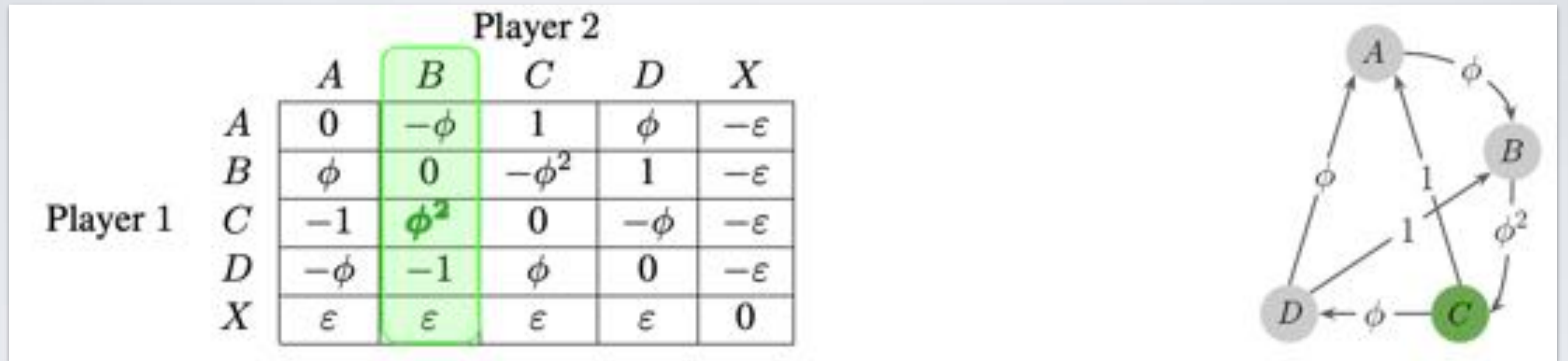
		Player 2				
		A	B	C	D	X
Player 1	A	0	$-\phi$	1	$\phi$	$-\epsilon$
	B	$\phi$	0	$-\phi^2$	1	$-\epsilon$
	C	-1	$\phi^2$	0	$-\phi$	$-\epsilon$
	D	$-\phi$	-1	$\phi$	0	$-\epsilon$
	X	$\epsilon$	$\epsilon$	$\epsilon$	$\epsilon$	0



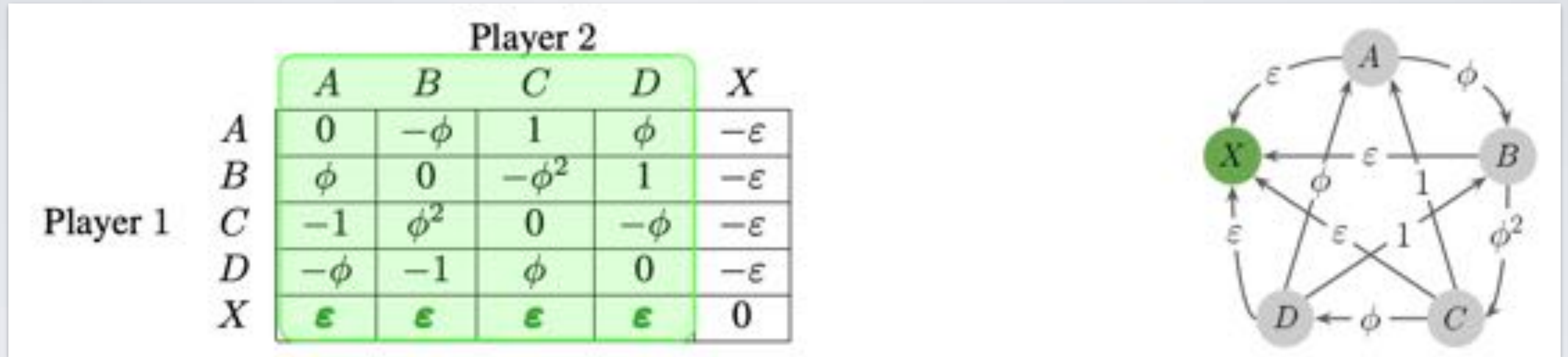
# $\alpha$ -PSRO [Muller et al. 2020] - Example



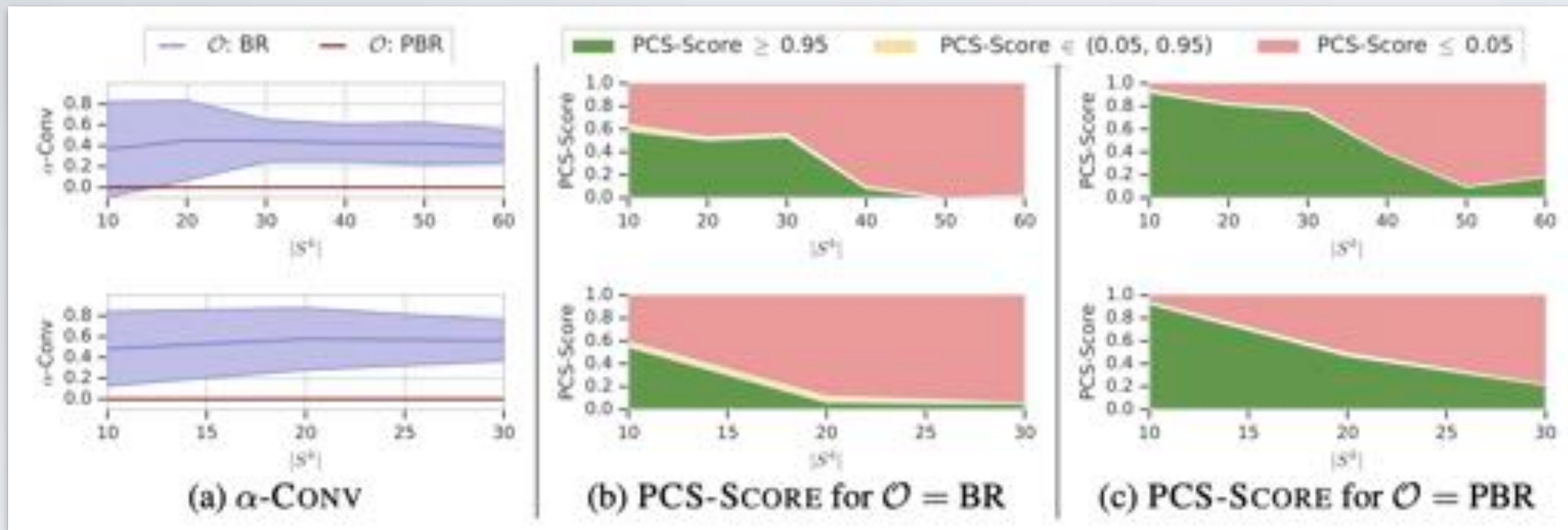
# $\alpha$ -PSRO [Muller et al. 2020] - Example



# $\alpha$ -PSRO [Muller et al. 2020] - Example

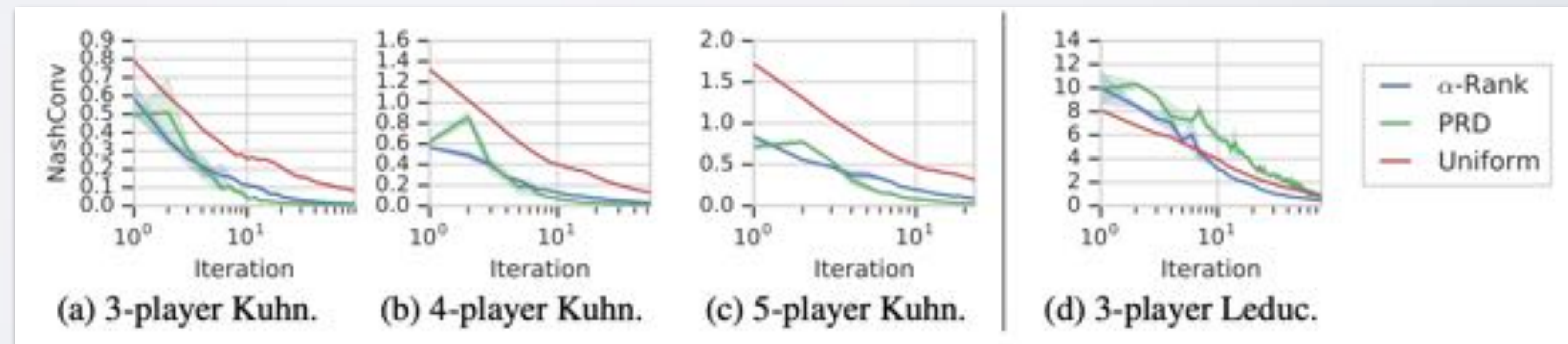


# $\alpha$ -PSRO [Muller et al. 2020] - Results



- Randomly generated K-Player General-Sum games with increasing  $|S^k|$
- Largest game considered has 24 million strategy profiles
- PBR outperforms in terms of  $\alpha$ -Rank support found

- $>2$  Player Poker games
- Similar / more competitive convergence performance





# Contents

- ~~Rectified Nash~~

- ~~Diverse-PSRO~~

- ~~Unified Behavioural + Response Diversity~~

- ~~NAC~~

- ~~$\alpha$ -PSRO~~

- **Joint PSRO**

- Pipeline PSRO

- Mixed Oracles / Opponents

# Joint PSRO [Marris et al. 2021]

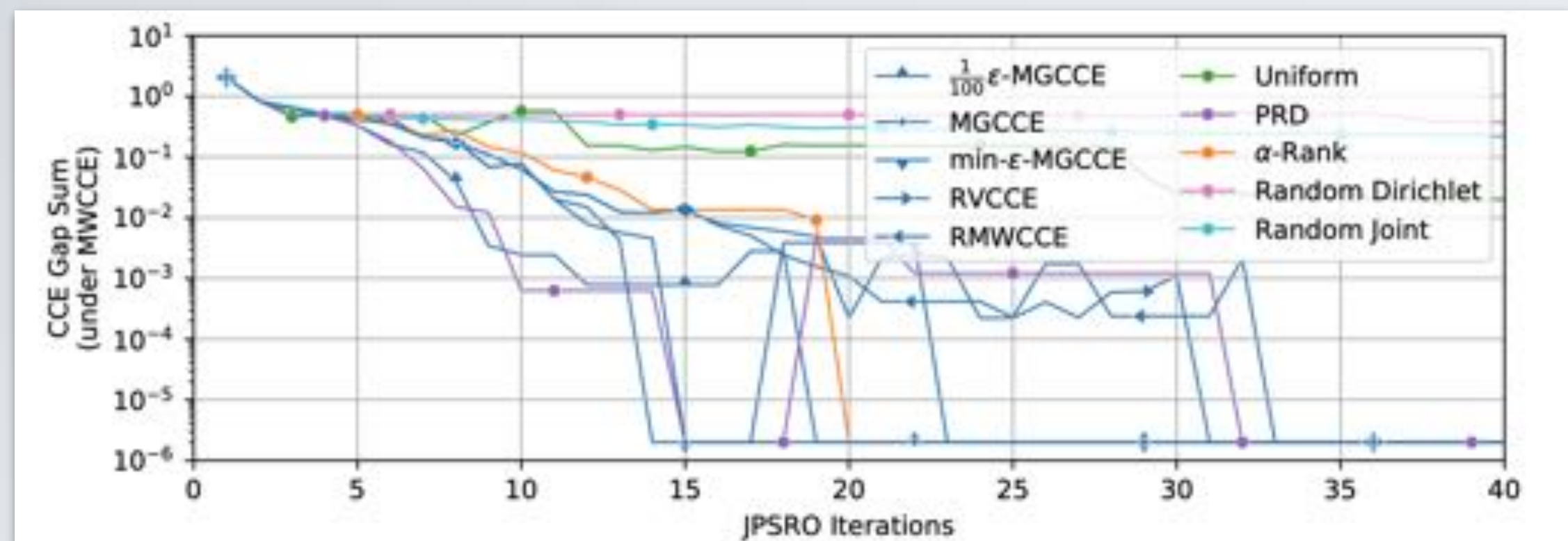
- Developed for n-player general-sum extensive-form games
- Maximum Gini Correlated Equilibrium as meta-solver

Maximised for a perfectly uniform mixed-strategy

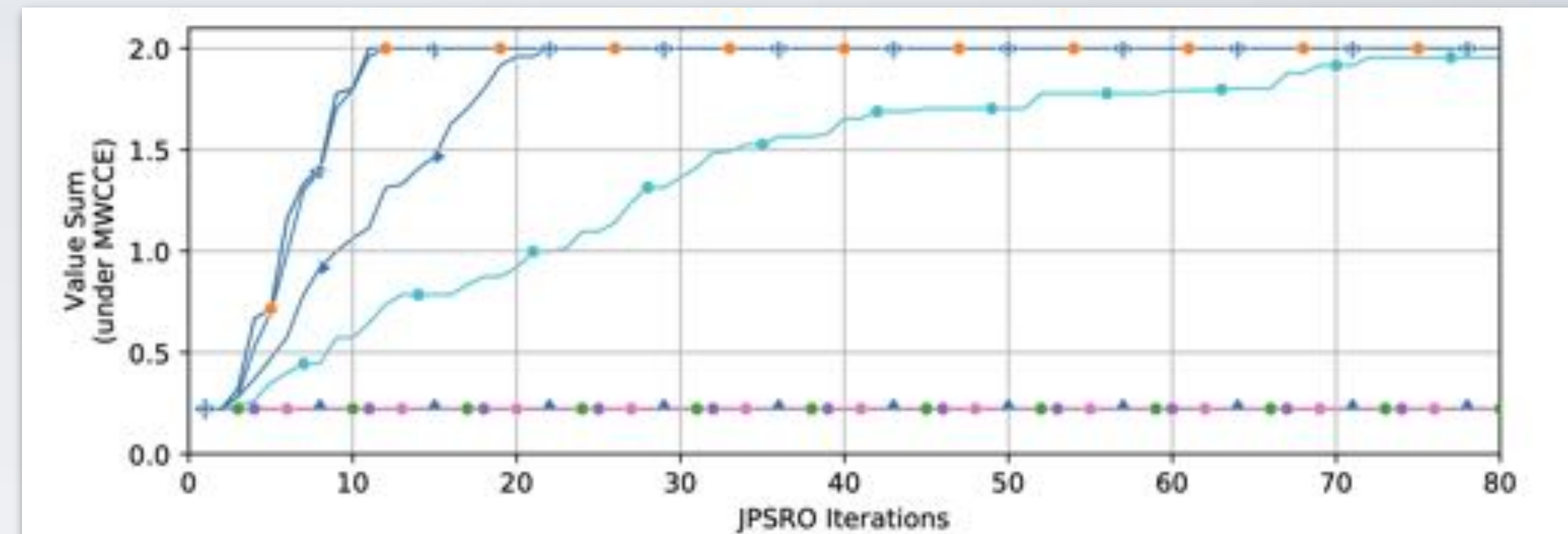
Gini objective:	$\max_{\sigma} -\frac{1}{2}\sigma^T\sigma$	s.t.
(C)CE constraints:	$A_p\sigma \leq \epsilon$	$\forall p$
Probability constraints:	$\sigma \geq 0$	$e^T\sigma = 1$

Correlated equilibrium is a joint mixed strategy where no player gains from a unilateral deviation

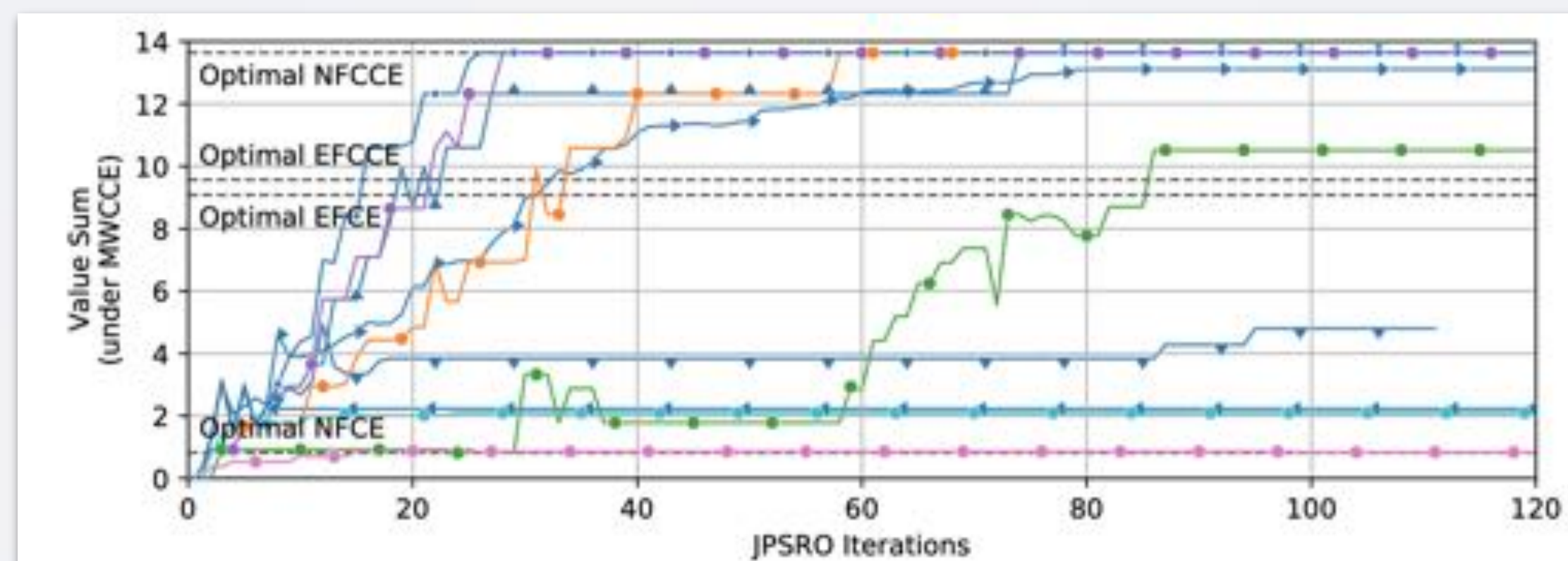
# Joint PSRO [Marris et al. 2021] - Results



(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.

# Contents

- ~~Rectified Nash~~

- ~~Diverse-PSRO~~

- ~~Unified Behavioural + Response Diversity~~

- ~~NAC~~

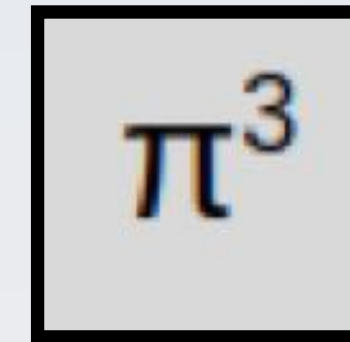
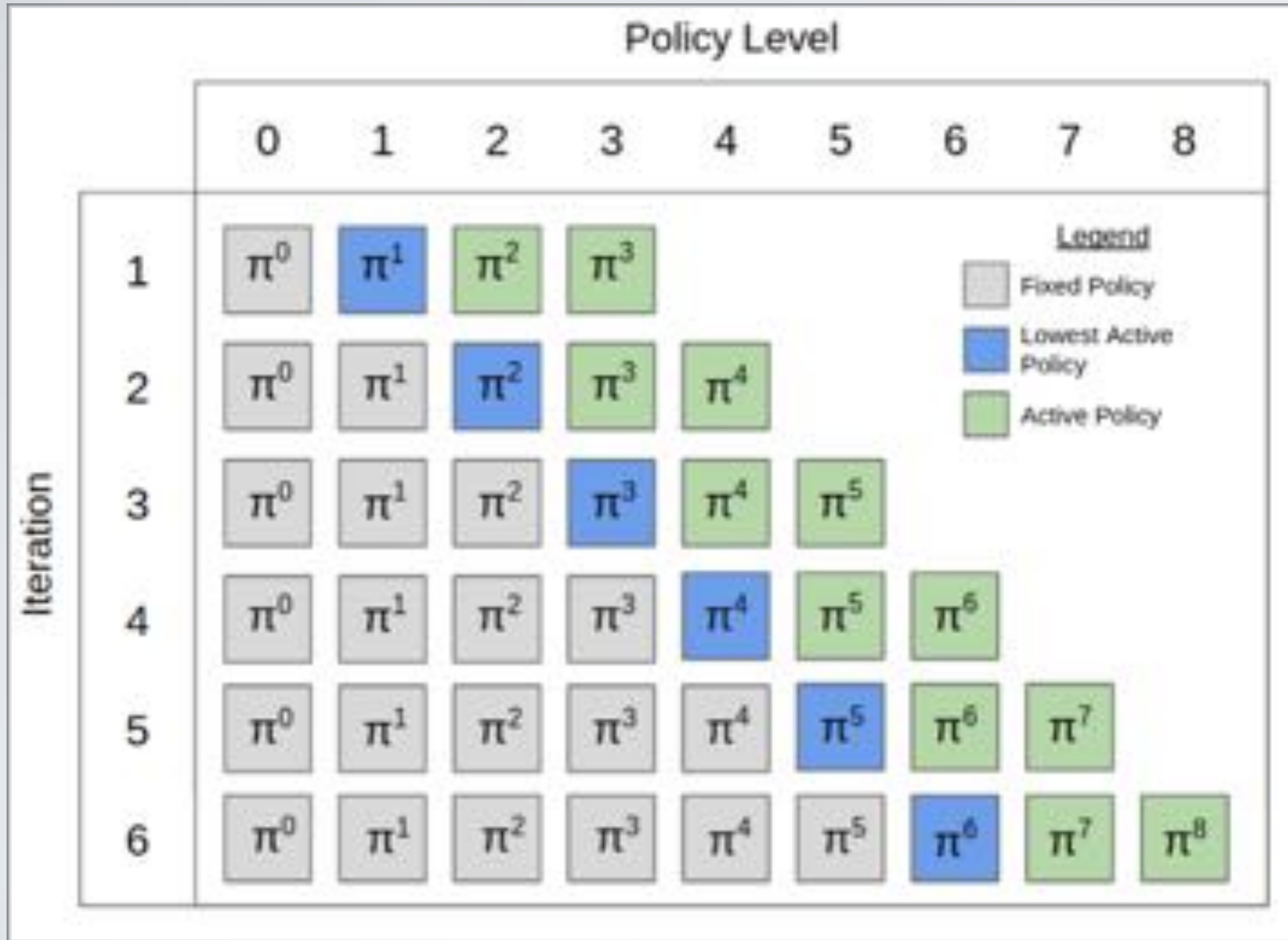
- ~~$\alpha$ -PSRO~~

- ~~Joint PSRO~~

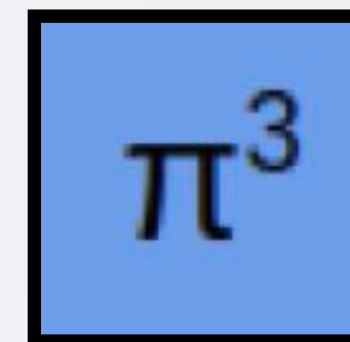
- **Pipeline PSRO**

- Mixed Oracles / Opponents

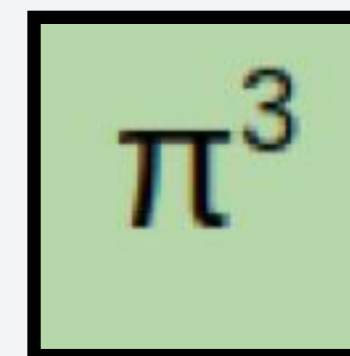
# Pipeline PSRO [McAleer 2020] - Algorithm



Fixed policies do not train anymore and remain within the fixed population.

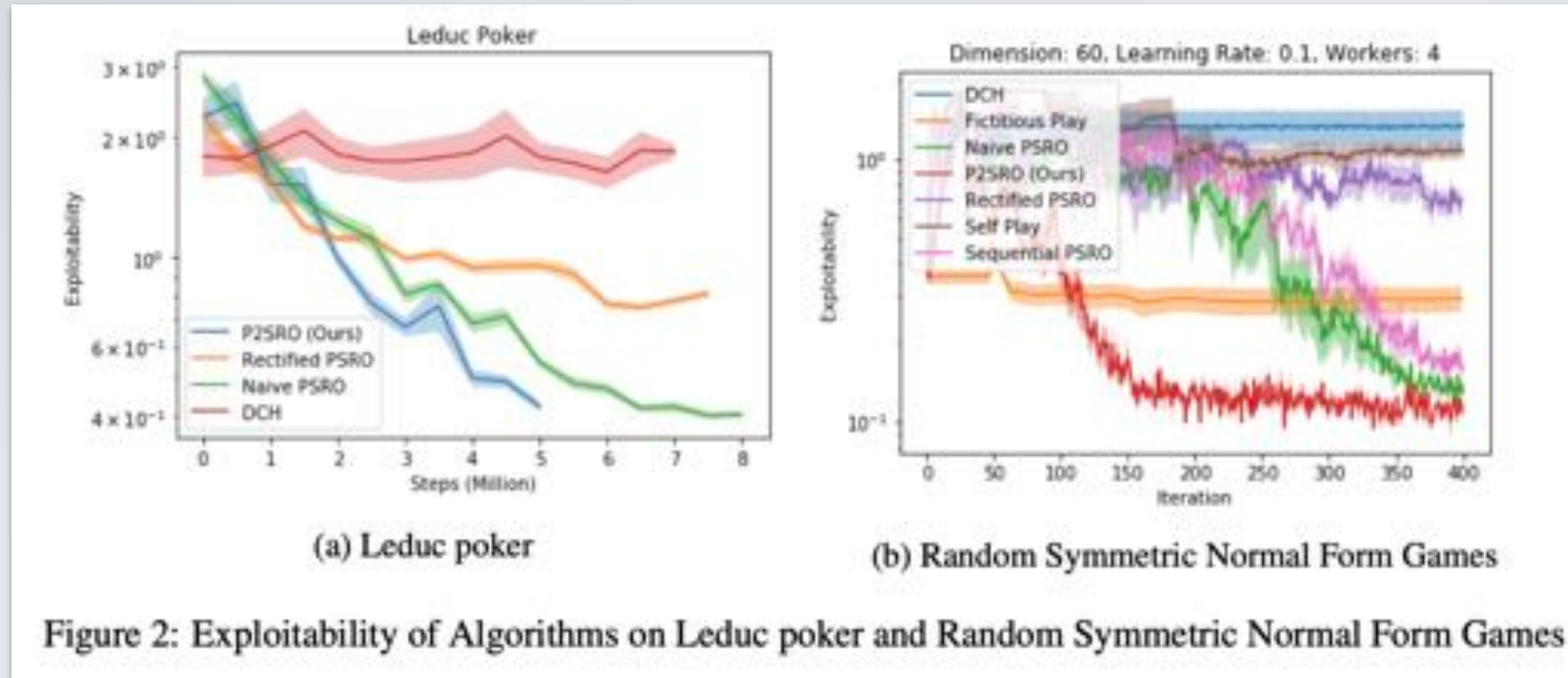


Lowest active policy trains against the meta-distribution defined by the fixed population



Active policies train against the meta-distribution defined by the population of agents below them in the pipeline

# Pipeline PSRO [McAleer 2020] - Results



- Pipeline PSRO reaches an approximate Nash equilibrium far quicker than other algorithms in Random Symmetric NFGs
- In Leduc Poker reaches low exploitability almost twice as quick than Naive PSRO - other algorithms do not reach low exploitability

- Barrage Stratego is a Two-Player Zero-Sum imperfect information game
- Game-tree complexity of  $10^{50}$
- Comparison vs. All existing bots for the game

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

# Contents

● ~~Rectified Nash~~

● ~~Diverse-PSRO~~

● ~~Unified Behavioural + Response Diversity~~

● ~~NAC~~

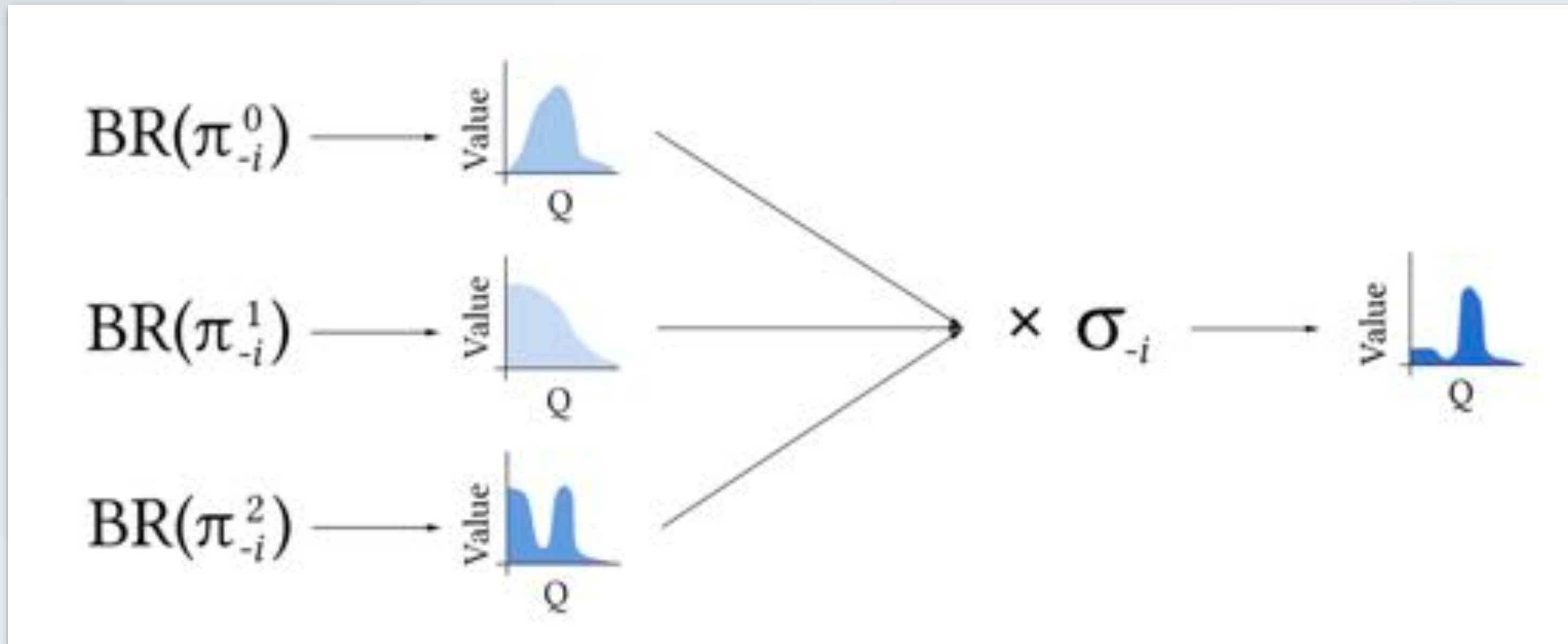
●  ~~$\alpha$ -PSRO~~

● ~~Joint PSRO~~

● ~~Pipeline PSRO~~

● **Mixed Oracles / Opponents**

# Q-Mixing [Smith et al. 2020]



- Learn best-responses to different policies  $\pi_{-i}^t$
- Transfer knowledge against opponent mixture by weighting Q-values according to current belief of opponent's policy

$$Q_i(o_i, a_i | \sigma_{-i}) = \sum_{\pi_{-i}} \psi_i(\pi_{-i} | o_i, \sigma_{-i}) Q_{\pi_i}(o_i, a_i | \pi_{-i})$$



Current belief about opponents' policy



# Mixed Oracles [Smith et al. 2021]

- During PSRO how can we transfer experience across iterations?
- Now maintain two populations

$$\Pi_t = \{\pi_1, \pi_2, \dots, \pi_t\} \quad \Lambda_t = \{\lambda_1, \lambda_2, \dots, \lambda_t\}$$

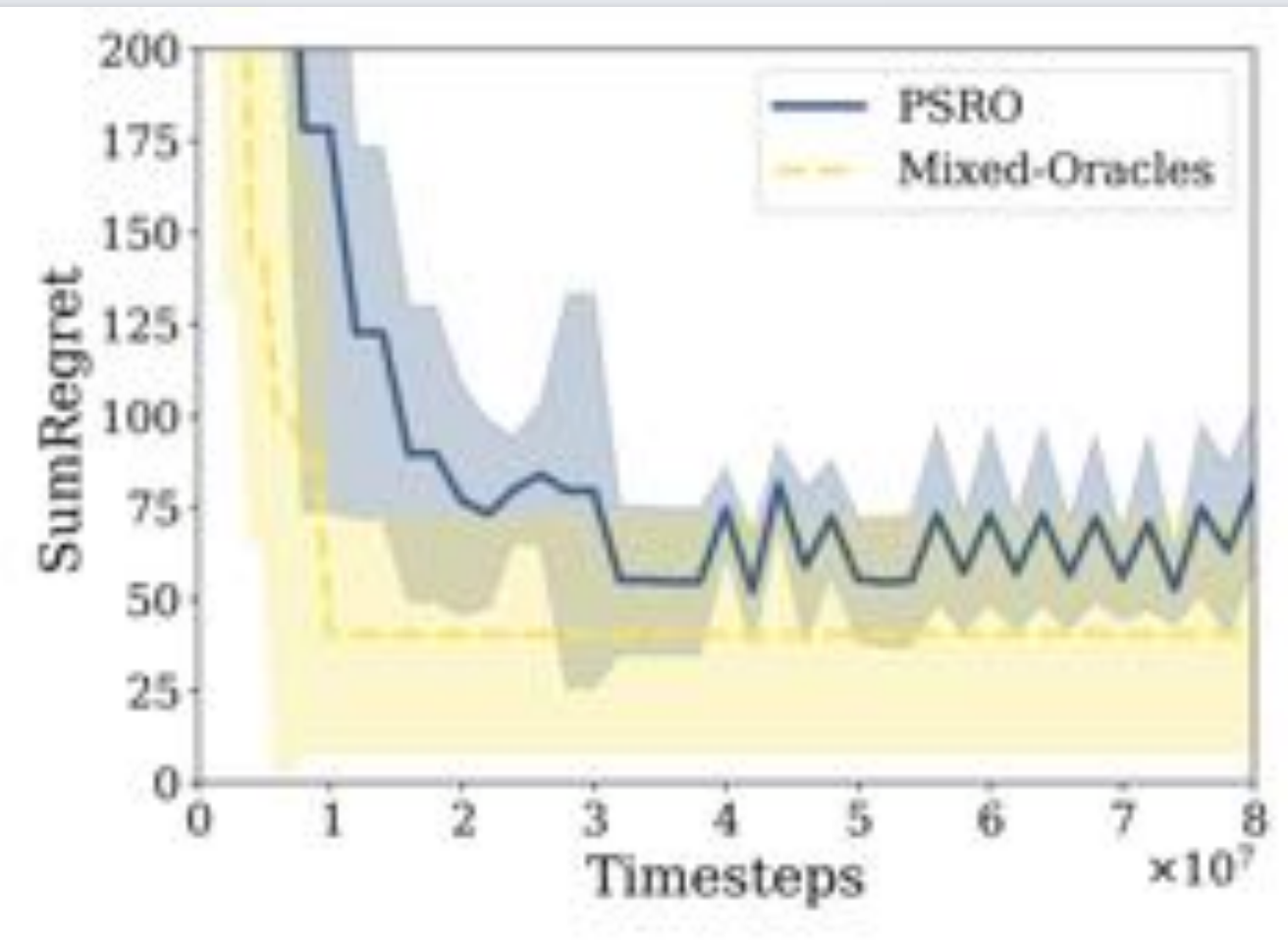
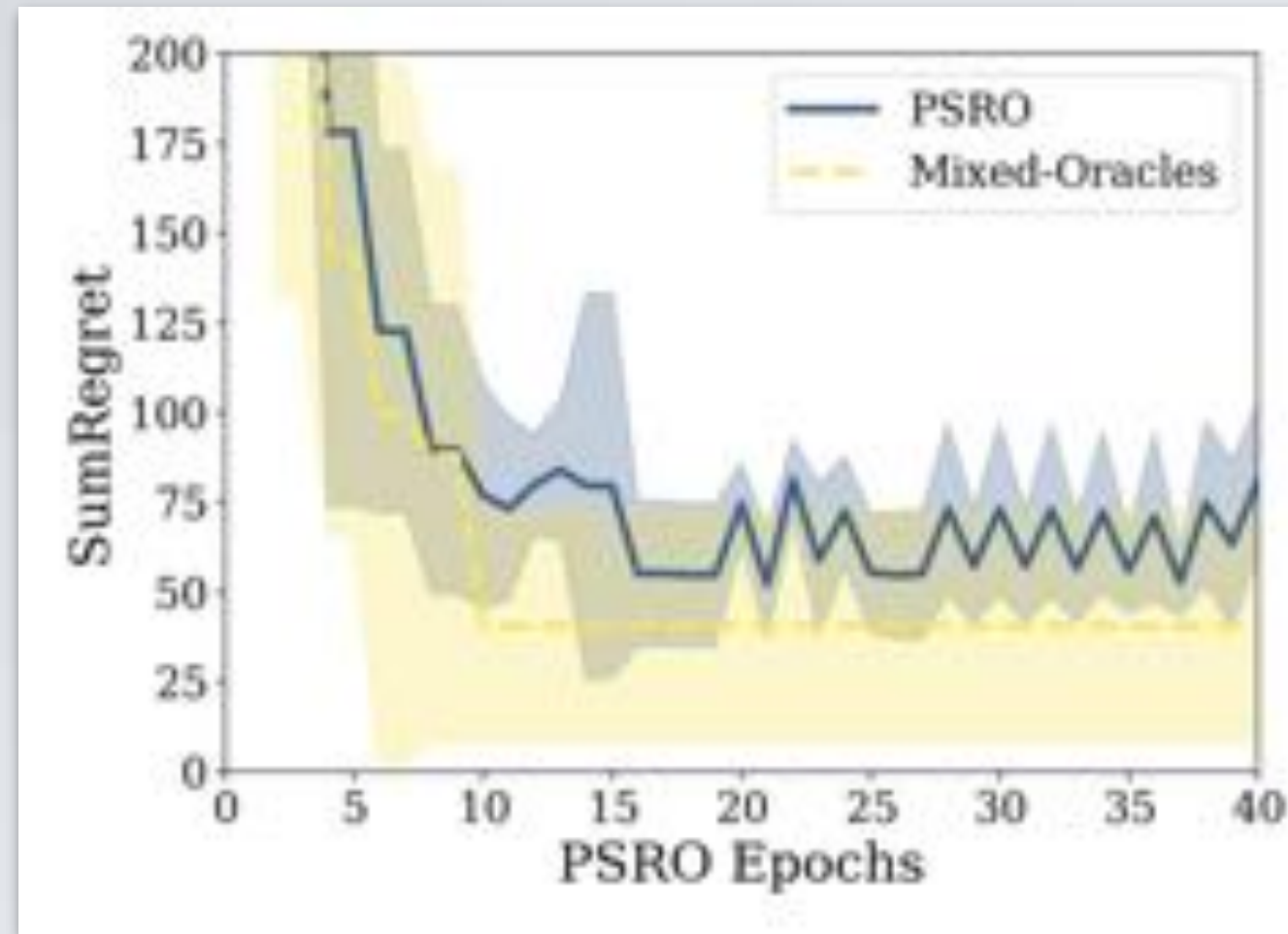
- Where  $\lambda_t$  is a learned best-response to  $\pi_t$  at every iteration, rather than against the meta-distribution
- We now have best-response experience against all policies in the population
- Use Q-mixing to find the new population policy

$$Q_i(o_i, a_i | \sigma_{-i}) = \sum_{\pi_{-i}} \psi_i(\pi_{-i} | o_i, \sigma_{-i}) Q_i(o_i, a_i | \pi_{-i})$$



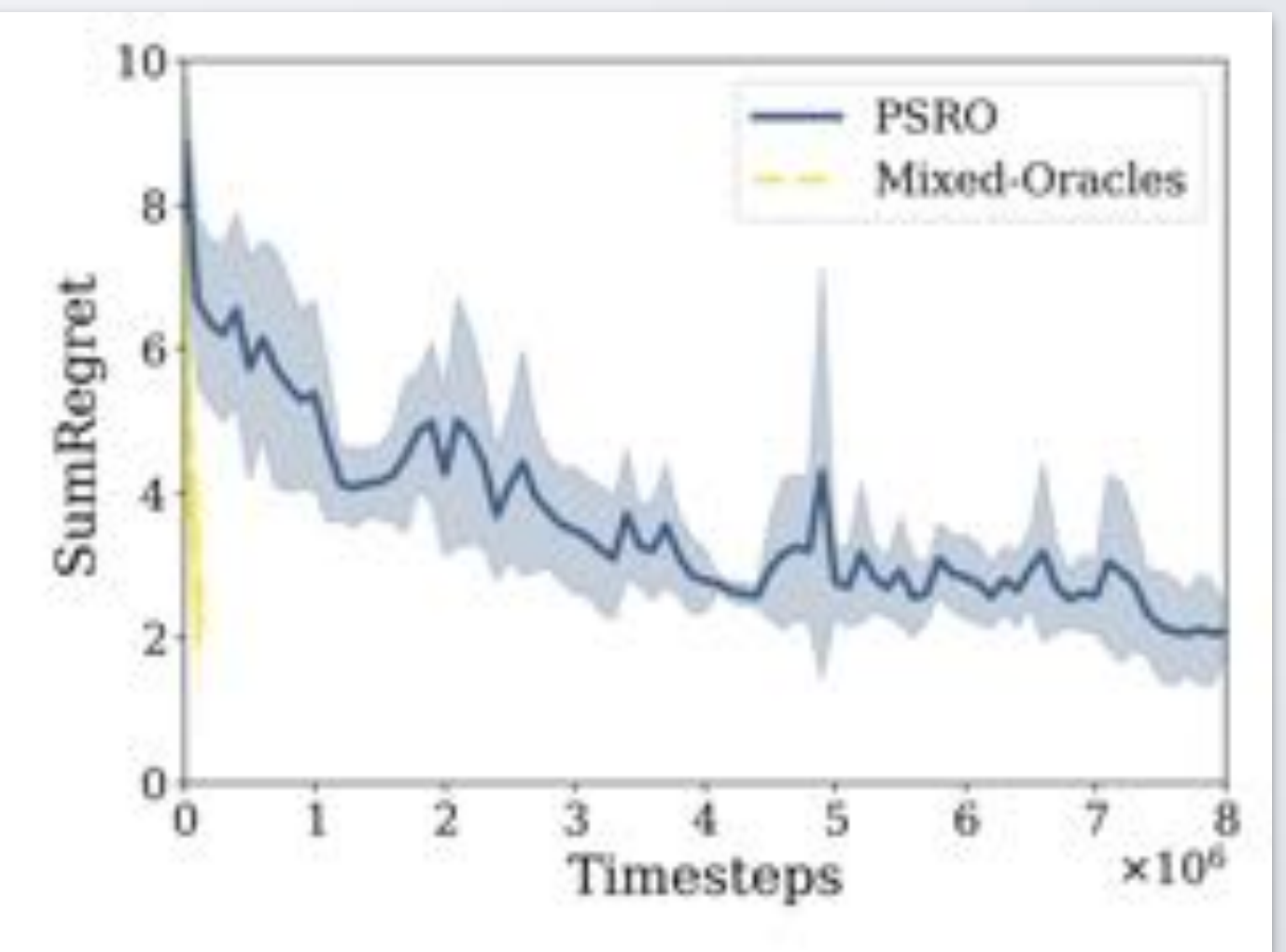
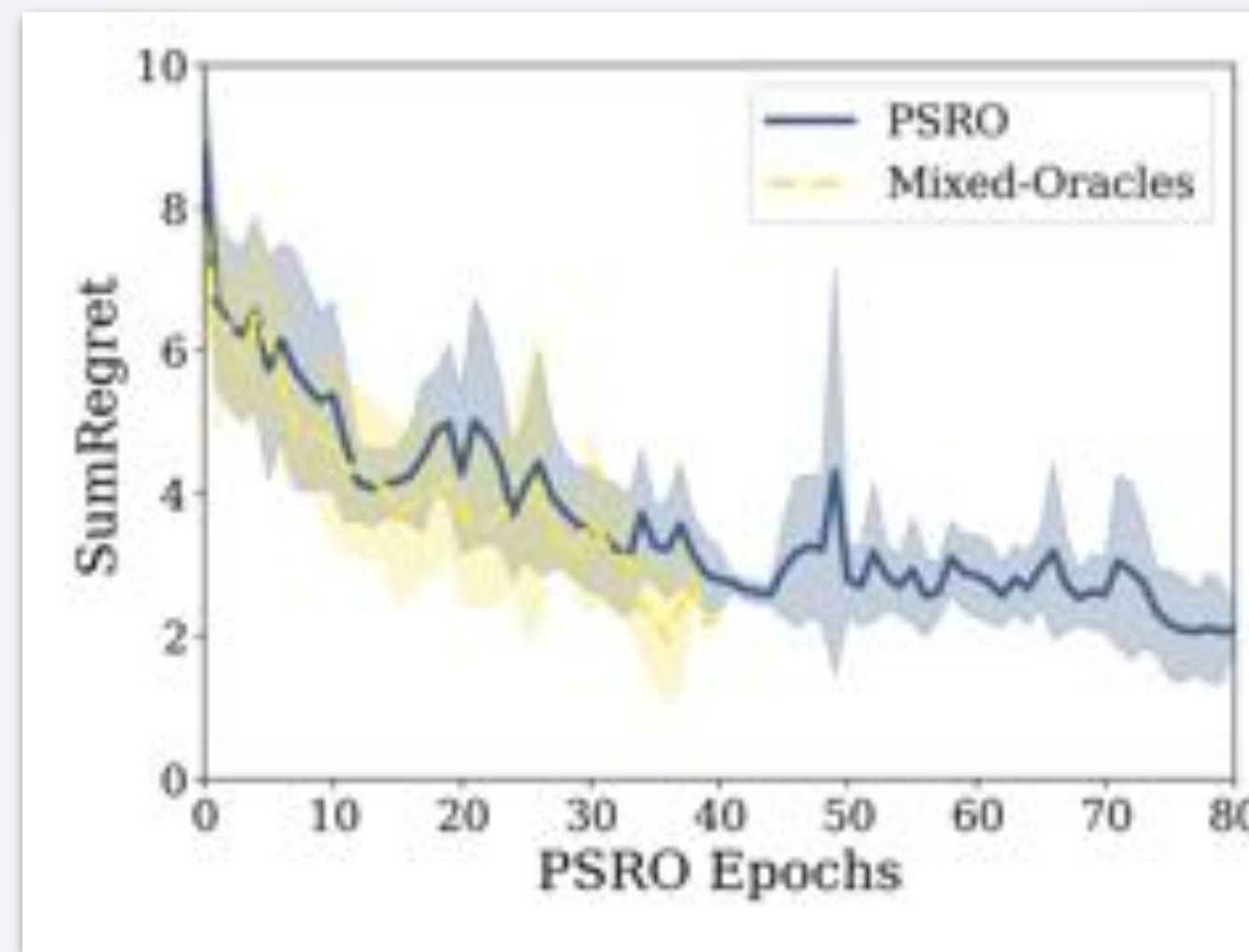
Probability of playing opponent  $\pi_{-i}$

# Mixed Oracles [Smith et al. 2021] - Results

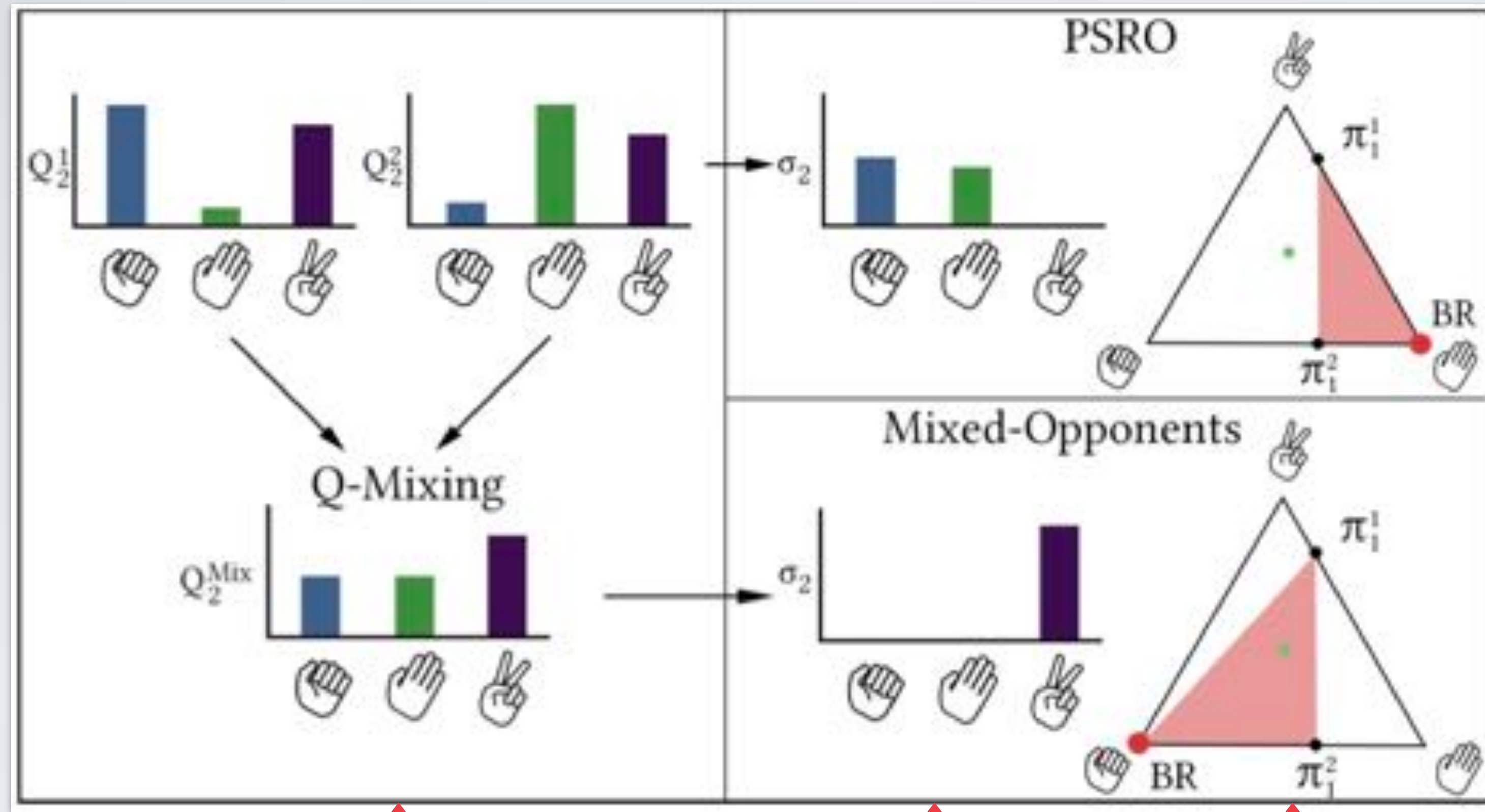


- General-Sum Tragedy of the Commons style game where individual interest is in conflict with the group interest
- Mixed-oracles converges in half the number of PSRO epochs.
- Utilises quarter the number of simulations

- Leduc Poker
- Mixed-oracles reaches similar performance in half the number of PSRO epochs.
- Drastically fewer number of time steps for a comparable solution



# Mixed Opponents [Smith et al. 2021]



BR includes old strategy  
Paper

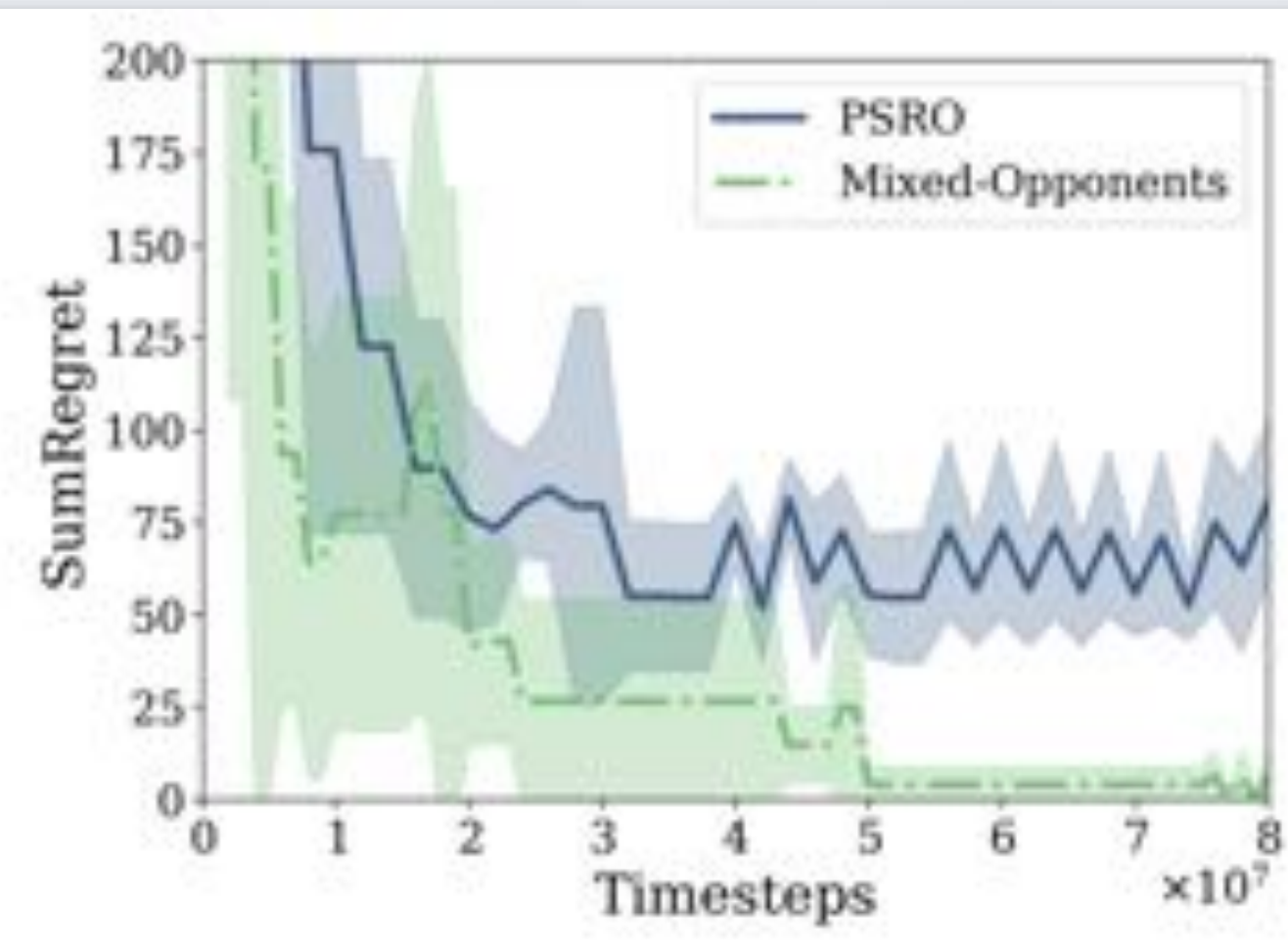
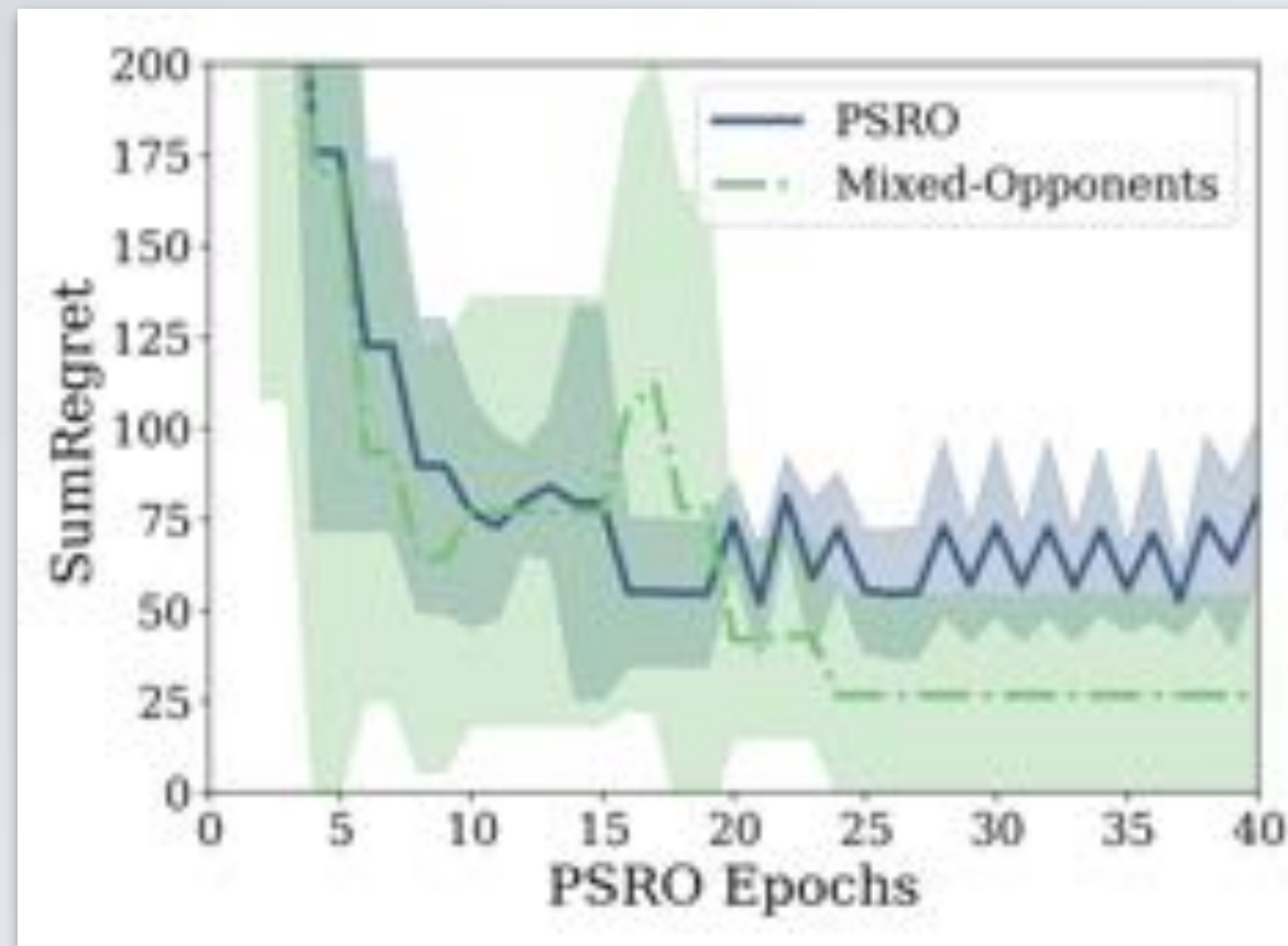
BR includes new strategy  
Rock

Opponent  
Policy Q-Values

Opponent  
Strategy

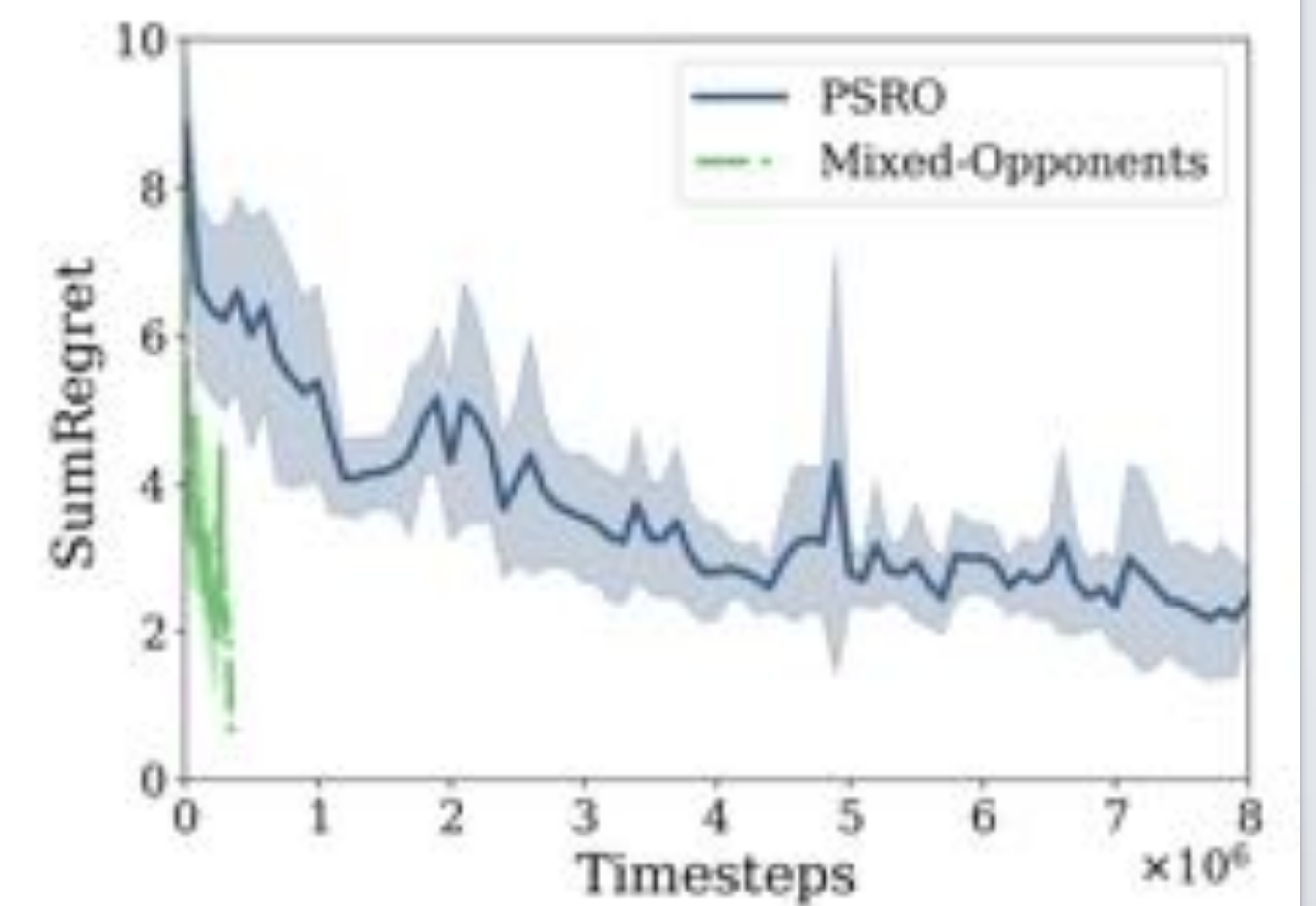
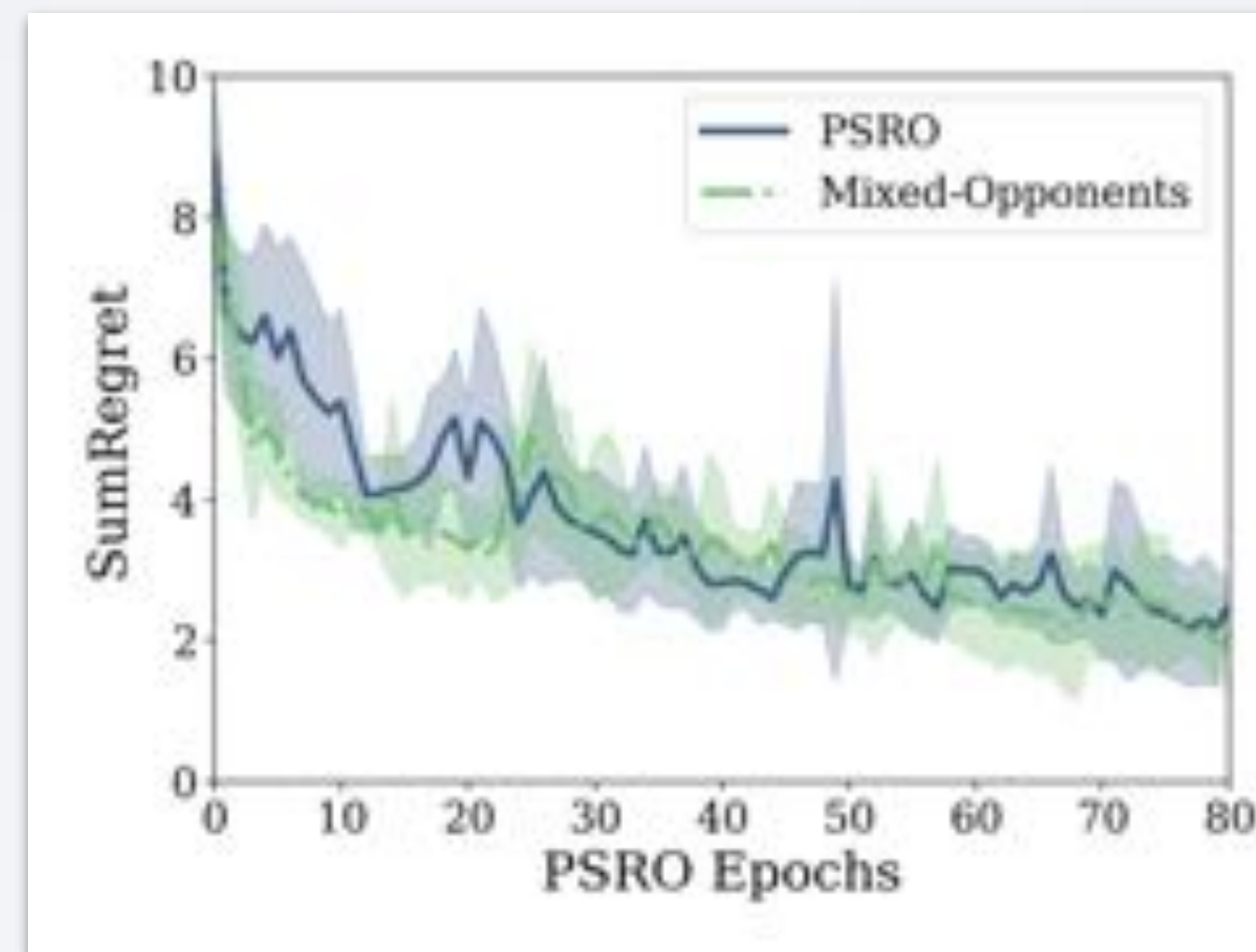
BR to  
Opponent  
Strategy

# Mixed Opponents [Smith et al. 2021] - Results



- General-Sum Tragedy of the Commons style game where individual interest is in conflict with the group interest
- Mixed-oracles appears to converge in a similar number of PSRO epochs.
- Whilst PSRO converges, mixed-opponents continually improves and nearly solves the game

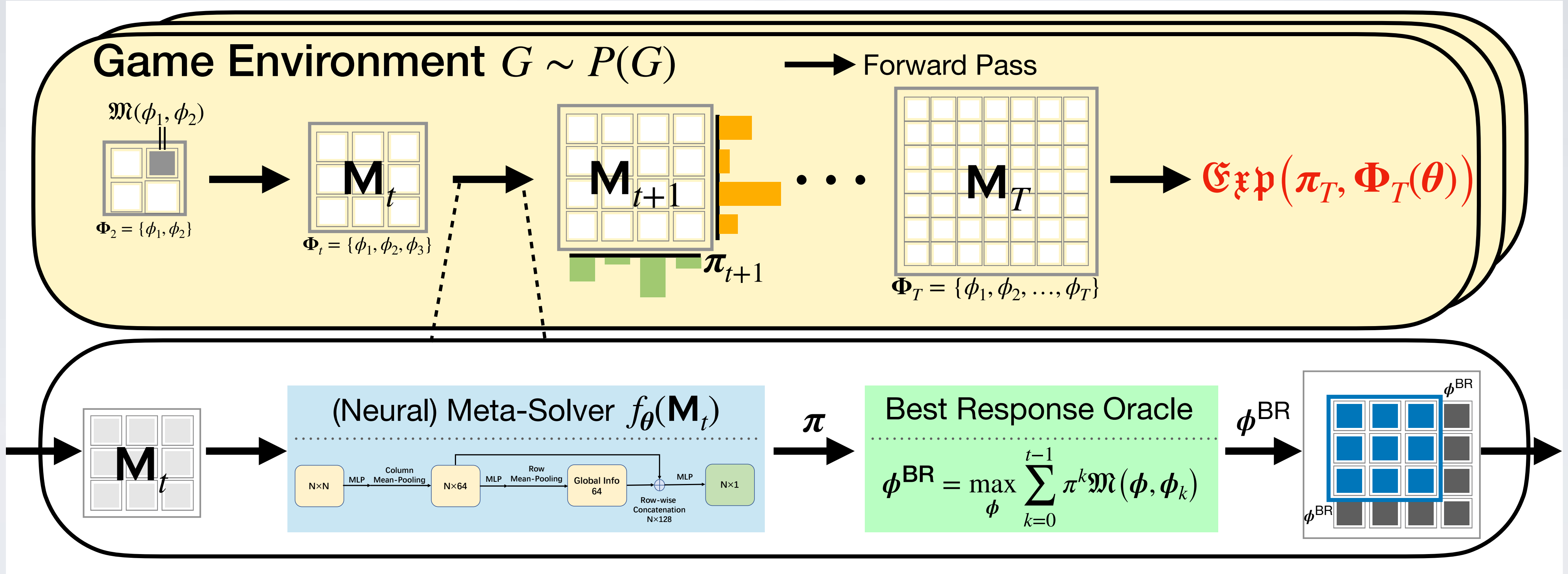
- Leduc Poker
- Mixed-oracles reaches similar performance in similar number of PSRO epochs to PSRO.
- Drastically fewer number of time steps for a comparable solution



# **PART III: Recent Advances & System Level Thinking**

Yaodong Yang

# Summary of Part II: PSRO Incorporate Many Variants



**Elo rating**  
**Correlated equilibrium**  
**Nash equilibrium**  
**Replicator dynamics**  
 $\alpha$ -Rank/ $\alpha^\alpha$ -Rank

**naive self-play**  
**fictitious play**  
**double oracle/PSRO**  
**PSRO-Nash**  
 $\alpha$ -PSRO  
**JPSRO**

# Two Mainstreams of Multi-Agent Learning algorithms

## Population-based methods:

- Fictitious play, double oracle, PSRO series, ...
- Regard the opponents fixed and seek for best responses.
- Easily and nicely integrated with RL methods (e.g., NFSP, PSRO)
- Work effectively in potential and zero-sum games, but limited in general-sum games.
- Average policy have convergence guarantee but generally no last-iteration convergence

## No-regret methods

- MWU, Follow the Regularised/Perturbed leader, CFR and all kinds of CFR variants, MCTS, ...
- Work in a self-play settings, no best-response step but a no-regret step.
- Often requires to know the model (the game tree, utility function/strategies of opponents, etc)
- A portal to the arsenal of online learning tools
- Have nice convergence guarantee to Nash zero-sum games, and CE/CCE in general-sum games.

# Two Mainstreams of Multi-Agent Learning algorithms

## Actor-Critic Policy Optimization in Partially Observable Multiagent Environments

Sriram Srinivasan<sup>\*1</sup>  
srsrinivasan@

Karl Tuyls<sup>1</sup>  
karltuyls@

...@google.com

### Abstract

Optimization of parameterized policies is a difficult and challenging problem. In this paper, we propose actor-critic algorithms in multiagent environments. We apply our method to multiagent sequential decision problems in several domains, showing performance comparable to state-of-the-art algorithms. Our method approximates Nash equilibria in self-play with rates similar to or better than a baseline model-free algorithm for zero-sum games, without any domain-specific state space reductions.

Under review as a conference paper at ICLR 2021

## THE ADVANTAGE OF REGRET-MATCHING ACTOR-CRITIC

Actor-Critic (ARMAC). Rather than saving past state-action data, ARMAC saves a buffer of *past policies*, replaying through them to reconstruct

Texas Hold'em.

They are somehow equivalent:  
 $\text{Regret} \approx \text{Value} !$

Can we design methods that  
take the merits from both side?



# Online Learning and No-Regret

- The settings of online learning:

- ♦ The algorithm picks a strategy  $p^t \in \Delta_{|A|}$  at time step  $t$
- ♦ The adversary/nature picks cost vector  $c^t : A \rightarrow [0,1]$
- ♦ An action  $a^t$  is drawn from  $p^t$ , and the algorithm incurs cost of  $c^t(a^t)$
- ♦ **Full-information settings:** observe the entire cost vector  $c^t$ . **Bandit settings:** only observe selected  $c^t(a^t)$
- ♦ **Oblivious adversary:**  $c^t$  only depends on  $t$ . **Adaptive adversary:**  $c^t$  depends on  $\{t, (p^1, \dots, p^t), (a^1, \dots, a^{t-1})\}$
- ♦ **Goal:** we try to learn how should we adapt our algorithms, **learn from mistakes [remember Alan Turing]!**

- The (external) regret of a sequence of actions w.r.t action  $a \in A$ :

$$R_T(a) = \frac{1}{T} \left( \sum_{t=1}^T c^t(a^t) - \sum_{t=1}^T c^t(a) \right)$$

- A no-regret algorithm  $\mathcal{A}$  is said to be **no-regret** (also **Hannan consistent**) if:

$$\lim_{T \rightarrow \infty} \mathbf{E} [R_T^{\mathcal{A}}(a)] = \frac{1}{T} \left( \sum_{i=1}^T \mathbf{E}_{a^i \sim p^i} [c^i(a^i)] - \sum_{t=1}^T c^t(a) \right) = 0, \forall a \in A$$

# No-Regret Learning in Zero-Sum Games

- If all players play no-regret methods, they can reach a CCE in general-sum games.
- Here we can show that no-regret players will reach the Nash in zero-sum games.
  - For better clarity. Let's assume row player chooses an action  $I_t \in \{1, \dots, N\}$ , mixed strategy  $\mathbf{p}_t = (p_{1,t}, \dots, p_{N,t})$
  - Column player instead of deciding  $c^t$ , let's assume they choose an action  $J_t \in \{1, \dots, M\}$  and  $\mathbf{q}_t = (q_{1,t}, \dots, q_{M,t})$
  - Assuming both players adopt no-regret algorithm regardless of what the opponent does, such that

$$\limsup_{T \rightarrow \infty} \left( \frac{1}{T} \sum_{t=1}^n \ell(I_t, J_t) - \min_{i=1, \dots, N} \frac{1}{T} \sum_{t=1}^n \ell(i, J_t) \right) \leq 0$$

- Recall that the Nash value of a two-player zero-sum game is  $\max_{\mathbf{q}} \min_{\mathbf{p}} \sum_{i=1}^N \sum_{j=1}^M p_i q_j \ell(i, j)$
- We have the main theorem that

**Theorem: assuming that in a two-player zero-sum game, if both players play no-regret**

**methods, then**  $\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \ell(I_t, J_t) = V = \max_{\mathbf{q}} \min_{\mathbf{p}} \sum_{i=1}^N \sum_{j=1}^M p_i q_j \ell(i, j)$  **almost surely.**

# No-Regret Learning in Zero-Sum Games

**Theorem: assuming that in a two-player zero-sum game, if both players play no-regret**

**methods, then**  $\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \ell(I_t, J_t) = V = \max_{\mathbf{q}} \min_{\mathbf{p}} \sum_{i=1}^N \sum_{j=1}^M p_i q_j \ell(i, j)$  **almost surely.**

**Proof:** 1) we first should that regardless of what column player plays, if the row player plays no-regret method, his loss will be no more than maximin value (i.e., worst case scenario to row player)  $\limsup_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \ell(I_t, J_t) \leq V$ .

Since the row player adopts no-regret method, we only need to show:

$$\begin{aligned} \min_{i=1, \dots, N} \frac{1}{T} \sum_{t=1}^T \ell(i, J_t) &= \min_{\mathbf{p}} \frac{1}{T} \sum_{t=1}^T \left( \sum_{i=1}^N p_i \ell(i, j) \right) \\ &= \min_{\mathbf{p}} \frac{1}{T} \sum_{t=1}^T \bar{\ell}(\mathbf{p}, J_t) \\ &= \min_{\mathbf{p}} \sum_{j=1}^M \left( \frac{1}{T} \sum_{t=1}^T \mathbf{1}_{\{J_t=j\}} \bar{\ell}(\mathbf{p}, j) \right) \\ &= \min_{\mathbf{p}} \bar{\ell}(\mathbf{p}, \hat{\mathbf{q}}_T) \\ &\leq \max_{\mathbf{q}} \min_{\mathbf{p}} \bar{\ell}(\mathbf{p}, \mathbf{q}) = V \end{aligned}$$

pure strategy is a special mixed strategy

change of notation

empirical mean on column player's action

expectation over the empirical mean of column player

done!

# No-Regret Learning in Zero-Sum Games

**Theorem: assuming that in a two-player zero-sum game, if both players play no-regret methods, then**  $\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \ell(I_t, J_t) = V = \max_{\mathbf{q}} \min_{\mathbf{p}} \sum_{i=1}^N \sum_{j=1}^M p_i q_j \ell(i, j)$  **almost surely.**

**Proof:** 2) we proved that  $\min_{i=1, \dots, N} \frac{1}{T} \sum_{t=1}^T \ell(i, J_t) \leq V$ , and since row player plays no-regret

$$\limsup_{T \rightarrow \infty} \left( \frac{1}{T} \sum_{t=1}^T \ell(I_t, J_t) - \min_{i=1, \dots, N} \frac{1}{T} \sum_{t=1}^T \ell(i, J_t) \right) \leq 0$$

we can know that

$$\limsup_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \ell(I_t, J_t) \leq V$$

for row player, we can do the same for the column player

$$\liminf_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \ell(I_t, J_t) \geq \min_{\mathbf{p}} \max_{\mathbf{q}} \bar{\ell}(\mathbf{p}, \mathbf{q}) = V$$

By von Neumann's minimax theorem, we prove that

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \ell(I_t, J_t) = V$$

This mean no-regret players self-play will reach to Nash equilibrium in two-player zero-sum games.

# No-Regret Learning in Zero-Sum Games

- We have showed that no-regret players will reach Nash equilibrium value.
- Furthermore, we can show that they **reach to the Nash strategy**.

**Theorem: In a two-player zero-sum game, if both players play no-regret methods, then**  
 $\hat{p}_{i,T} = \frac{1}{T} \sum_{t=1}^T \mathbf{1}_{\{I_t=i\}}, \quad \hat{q}_{j,T} = \frac{1}{T} \sum_{t=1}^T \mathbf{1}_{\{J_t=j\}}$  **almost surely converge to the set of Nash equilibrium.**

**Proof:** in the previous proof, we have shown that

$$\min_{\mathbf{p}} \bar{\ell}(\mathbf{p}, \hat{\mathbf{q}}_T) \leq V, \quad \max_{\mathbf{q}} \bar{\ell}(\hat{\mathbf{p}}_T, \mathbf{q}) \geq V$$

and due to the uniqueness of  $V$  value in zero-sum games

$$\min_{\mathbf{p}} \bar{\ell}(\mathbf{p}, \hat{\mathbf{q}}_T) = \max_{\mathbf{q}} \bar{\ell}(\hat{\mathbf{p}}_T, \mathbf{q}) = V$$

and because of

$$\bar{\ell}(\hat{\mathbf{p}}_T, \hat{\mathbf{q}}_T) \geq \min_{\mathbf{p}} \bar{\ell}(\mathbf{p}, \hat{\mathbf{q}}_T), \quad \max_{\mathbf{q}} \bar{\ell}(\hat{\mathbf{p}}_T, \mathbf{q}) \geq \bar{\ell}(\hat{\mathbf{p}}_T, \hat{\mathbf{q}}_T)$$

finally, we prove that

$$\bar{\ell}(\hat{\mathbf{p}}_T, \hat{\mathbf{q}}_T) = V$$

This means that the empirical mean of  $\hat{\mathbf{p}}_T, \hat{\mathbf{q}}_T$  is the Nash

# Fictitious Play is Not No-Regret

- No-regret can lead to NE in two-player zero-sum.
- But, what exactly is a no-regret algorithm? How can we behave to achieve no-regret?

$$\limsup_{T \rightarrow \infty} \left( \frac{1}{T} \sum_{t=1}^n \ell(I_t, J_t) - \min_{i=1, \dots, N} \frac{1}{T} \sum_{t=1}^n \ell(i, J_t) \right) \leq 0$$

- **Surprisingly, Fictitious play is not no-regret !**

- ◆ Recall that fictitious play, also known as **Follow-the-Leader**, is to take the best response to the average cumulative loss.

$$I_t = \operatorname{argmin}_{i=1, \dots, N} \left( \frac{1}{t-1} \sum_{s=1}^{t-1} \ell(i, J_s) \right)$$

- ◆ Consider  $N = 2$  actions, let  $J_t$  be chosen such that  $\ell(1, J_t) = (1/2, 0, 1, 0, 1, \dots)$  and  $\ell(2, J_t) = (1/2, 1, 0, 1, 0, \dots)$
- ◆ Then the accumulative loss for both actions is **T/2**, and the FP will suffer a loss of **T**, thus has constant regret.
- ◆ There are many variants that make FP no-regret, for example **Follow-the-Perturbed-Leader**:

$$I_t = \operatorname{argmin}_{i=1, \dots, N} \left( \frac{1}{t-1} \sum_{s=1}^{t-1} \ell(i, J_s) + Z_{i,t} \right), \mathbf{Z}_t \text{ any i.i.d random vectors of size } N$$

# Adversarial Bandit — MWU

- Fictitious play is not no-regret !
- Let's introduce a true no-regret algorithm: **Multiplicative Weight Update [Freund 1999]**.
  - ♦ MWU has many names: *Hedge, Exponential Weights Algorithms, Randomised Weighted Majority*

$$p_{i,t} = \frac{\exp\left(-\eta \sum_{s=1}^{t-1} \ell(i, J_s)\right)}{\sum_{k=1}^{|A|} \exp\left(-\eta \sum_{s=1}^{t-1} \ell(k, J_s)\right)}$$

- ♦ Equivalently, one can think of the following iterative process:

$$p_t = \frac{w^t}{\sum_{a \in A} w^t(a)}, \quad w^{t+1}(a) = w^t(a) \cdot \exp\left(-\eta \ell(a, J_t)\right), \quad w^1(a) = \mathbf{1} \quad \forall a \in A$$

- Large  $\eta$  means more exploitation, small  $\eta$  means more exploration.
- ♦ Equivalently, one can get MWU by the following maximum entropy framework (a common trick in RL).

$$\arg \min_{p \in \Delta_{|A|}} \bar{\ell}(p, J_s) + 1/\eta \cdot \sum_i p_i \log p_i$$

similar to soft Q-learning !

# Adversarial Bandit — MWU

- Let's now prove MWU is indeed a no-regret method

◆ Let  $\mathbf{opt} = \min_{a \in A} \sum_{t=1}^n \ell(a, J_t)$  and  $\mathbf{v}^t = \sum_{a \in |A|} p_t(a) \cdot \ell(a, J_t)$ , thus the regret-bound is  $\lim_{T \rightarrow \infty} \frac{1}{T} \left( \sum_{t=1}^T \mathbf{v}^t - \mathbf{opt} \right)$

◆ We first bound the denominator  $\sum_{a \in A} w^t(a) \geq w^t(a^*) = \exp(-\eta \cdot \mathbf{opt})$

multiplication rule of exp function

$$\geq (1 - \epsilon)^{\mathbf{opt}}$$

assume  $1 - \epsilon = e^{-\eta}$

$$\sum_a w^{t+1}(a) = \sum_a w^t(a) \cdot \exp(-\eta \ell(a, J_t)) := \sum_a w^t(a) \cdot (1 - \epsilon)^{\ell(a, J_t)}$$

assume  $1 - \epsilon = e^{-\eta}$

$$\leq \sum_a w^t(a) \cdot (1 - \epsilon \ell(a, J_t)) = \sum_a w^t(a) \cdot (1 - \epsilon \mathbf{v}^t)$$

definition of  $\mathbf{v}^t = \sum_{a \in |A|} \frac{w^t(a)}{\sum_a w^t(a)} \cdot \ell(a, J_t)$

◆ Merge the upper and lower bound  $(1 - \epsilon)^{\mathbf{opt}} \leq \sum_{a \in A} w^t(a) \leq \sum_a w^1(a) \prod_{t=1}^T (1 - \epsilon \mathbf{v}^t) = |A| \prod_{t=1}^T (1 - \epsilon \mathbf{v}^t)$

$$\mathbf{opt} \cdot (-\epsilon - \epsilon^2) \leq \ln |A| + \sum_{t=1}^T (-\epsilon \mathbf{v}^t)$$

Taylor expansion  $\ln(1 - x) = -x - x^2/2$

◆ Set  $\epsilon = \sqrt{\ln |A| / T}$  we conclude the proof  $\frac{1}{T} \left( \sum_{t=1}^T \mathbf{v}^t - \mathbf{opt} \right) \leq \frac{1}{T} \left( \epsilon T + \frac{1}{\epsilon} \ln |A| \right) \leq 2\sqrt{\frac{\ln |A|}{T}} \rightarrow 0$

Note: the log term is great for many real-world problems!



# Online Double Oracle: Merits from both worlds

## Online Double Oracle

Le Cong Dinh<sup>\*,1,2</sup>, Yaodong Yang<sup>\*,1,4</sup>, Nicolas Perez-Nieves<sup>3</sup>, Oliver Slumbers<sup>4</sup>,

Zheng Tian<sup>4</sup>, David Henry Mguni<sup>1</sup>, Haitham Bou Ammar<sup>1</sup>, Jun Wang<sup>1,4</sup>

1. Nash is unexploitable, but when a player always plays Rock, you should play Paper rather than (1/3, 1/3, 1/3).
2. Double Oracle/PSRO assumes both players play the worst-case scenario, can be too **pessimistic** during training.
3. Online learning provides a framework about how to exploit opponents through minimising regret.

### Algorithm 1 Double Oracle (McMahan et al., 2003)

```

1: Input: A set  $\Pi, C$  strategy set of players
2:  $\Pi_0, C_0$ : initial set of strategies
3: for  $t = 1$  to  $\infty$  do
4:   if  $\Pi_t \neq \Pi_{t-1}$  or  $C_t \neq C_{t-1}$  then
5:     Solve the NE of the subgame  $G_t$ :
6:      $(\pi_t^*, c_t^*) = \arg \min_{\pi \in \Delta_{\Pi_t}} \arg \max_{c \in \Delta_{C_t}} \pi^\top A c$ 
7:     Find the best response  $a_{t+1}$  and  $c_{t+1}$  to  $(\pi_t^*, c_t^*)$ :
8:      $a_{t+1} = \arg \min_{a \in \Pi} a^\top A c_t^*$ 
9:      $c_{t+1} = \arg \max_{c \in C} \pi_t^{*\top} A c$ 
10:    Update  $\Pi_{t+1} = \Pi_t \cup \{a_{t+1}\}, C_{t+1} = C_t \cup \{c_{t+1}\}$ 
11:  else if  $\Pi_t = \Pi_{t-1}$  and  $C_t = C_{t-1}$  then
12:    Terminate
13:  end if
14: end for

```

### What we want:

if opponents play  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_T$ , we want the player to have  $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \dots, \boldsymbol{\pi}_T$  s.t.

$$\lim_{T \rightarrow \infty} \frac{R_T}{T} = 0, \quad R_T = \max_{\pi \in \Delta_{\Pi}} \sum_{t=1}^T (\pi_t^\top A c_t - \pi^\top A c_t)$$

### What we know:

hedge algorithm/multiplicative weight update can achieve no-regret property if one follows the below update

$$\pi_{t+1}(i) = \pi_t(i) \frac{\exp(-\mu_t a^{i\top} A c_t)}{\sum_{i=1}^n \pi_t(i) \exp(-\mu_t a^{i\top} A c_t)}, \quad \forall i \in [n]$$

the regret of MWU is  $\mathcal{O}(\sqrt{\log(n)/T})$

# Online Double Oracle: Merits from both worlds

---

**Algorithm 1** Double Oracle (McMahan et al., 2003)

---

```
1: Input: A set  $\Pi, C$  strategy set of players
2:  $\Pi_0, C_0$ : initial set of strategies
3: for  $t = 1$  to  $\infty$  do
4:   if  $\Pi_t \neq \Pi_{t-1}$  or  $C_t \neq C_{t-1}$  then
5:     Solve the NE of the subgame  $G_t$ :
6:      $(\pi_t^*, c_t^*) = \arg \min_{\pi \in \Delta_{\Pi_t}} \arg \max_{c \in \Delta_{C_t}} \pi^\top A c$ 
7:     Find the best response  $a_{t+1}$  and  $c_{t+1}$  to  $(\pi_t^*, c_t^*)$ :
8:      $a_{t+1} = \arg \min_{a \in \Pi} a^\top A c_t^*$ 
9:      $c_{t+1} = \arg \max_{c \in C} \pi_t^{*\top} A c$ 
10:    Update  $\Pi_{t+1} = \Pi_t \cup \{a_{t+1}\}, C_{t+1} = C_t \cup \{c_{t+1}\}$ 
11:   else if  $\Pi_t = \Pi_{t-1}$  and  $C_t = C_{t-1}$  then
12:     Terminate
13:   end if
14: end for
```

---

---

**Algorithm 2:** Online Single Oracle Algorithm

---

```
1: Input: Player's pure strategy set  $\Pi$ 
2: Init. effective strategies set:  $\Pi_0 = \Pi_1 = \{a^j\}, a^j \in \Pi$ 
3: for  $t = 1$  to  $T$  do
4:   if  $\Pi_t = \Pi_{t-1}$  then
5:     Compute  $\pi_t$  by the MWU in Equation (5)
6:   else if  $\Pi_t \neq \Pi_{t-1}$  then
7:     Start a new time window  $T_{i+1}$  and
8:     Reset  $\pi_t = [1/|\Pi_t|, \dots, 1/|\Pi_t|], \bar{l} = \mathbf{0}$ 
9:   end if
10:  Observe  $l_t$  and update the average loss in  $T_i$ :
11:   $\bar{l} = \sum_{t \in T_i} l_t / |T_i|$ 
12:  Calculate the best response:  $a_t = \arg \min_{\pi \in \Pi} \langle \pi, \bar{l} \rangle$ 
13:  Update the set of strategies:  $\Pi_{t+1} = \Pi_t \cup \{a_t\}$ 
14: end for
15: Output:  $\pi_T, \Pi_T$ 
```

---

**Intuition:** maintain a time window  $T_i$  to track opponent's strategy, if no new best response can be found, then keep exploiting, otherwise refresh the time window to catch up with the latest change

# Online Double Oracle: Merits from both worlds

1. OSO is a no-regret algorithm.

**Theorem 4 (Regret Bound of OSO).** *Let  $l_1, l_2, \dots, l_T$  be a sequence of loss vectors played by an adversary, and  $\langle \cdot, \cdot \rangle$  be the dot product, OSO in Algorithm 2 is a no-regret algorithm with*

$$\frac{1}{T} \left( \sum_{t=1}^T \langle \pi_t, l_t \rangle - \min_{\pi \in \Pi} \sum_{t=1}^T \langle \pi, l_t \rangle \right) \leq \frac{\sqrt{k \log(k)}}{\sqrt{2T}},$$

where  $k = |\Pi_T|$  is the size of effective strategy set in the final time window.

## Algorithm 2: Online Single Oracle Algorithm

```

1: Input: Player's pure strategy set  $\Pi$ 
2: Init. effective strategies set:  $\Pi_0 = \Pi_1 = \{a^j\}, a^j \in \Pi$ 
3: for  $t = 1$  to  $T$  do
4:   if  $\Pi_t = \Pi_{t-1}$  then
5:     Compute  $\pi_t$  by the MWU in Equation (5)
6:   else if  $\Pi_t \neq \Pi_{t-1}$  then
7:     Start a new time window  $T_{i+1}$  and
       Reset  $\pi_t = [1/|\Pi_t|, \dots, 1/|\Pi_t|], \bar{l} = \mathbf{0}$ 
8:   end if
9:   Observe  $l_t$  and update the average loss in  $T_i$ :
        $\bar{l} = \sum_{t \in T_i} l_t / |T_i|$ 
10:  Calculate the best response:  $a_t = \arg \min_{\pi \in \Pi} \langle \pi, \bar{l} \rangle$ 
11:  Update the set of strategies:  $\Pi_{t+1} = \Pi_t \cup \{a_t\}$ 
12: end for
13: Output:  $\pi_T, \Pi_T$ 

```

2. Putting OSO into self-play settings, we get Online Double Oracle which can solve Nash.

- ◆ Recall that in two-player zero-sum game, if two no-regret methods self play, the outcome will leads to a Nash equilibrium!
- ◆ We just prove that.

## Algorithm 3: Online Double Oracle Algorithm

```

1: Input: Full pure strategy set  $\Pi, C$ 
2: Init. effective strategies set:  $\Pi_0 = \Pi_1, C_0 = C_1$ 
3: for  $t = 1$  to  $T$  do
4:   Each player follows the OSO in Algorithm 2 with
       their respective effective strategy sets  $\Pi_t, C_t$ 
5: end for
6: Output:  $\pi_T, \Pi_T, c_T, C_T$ 

```

**Theorem 5.** *Suppose both players apply OSO. Let  $k_1, k_2$  denote the size of effective strategy set for each player. Then, the average strategies of both players converge to the NE with the rate:*

$$\epsilon_T = \sqrt{\frac{k_1 \log(k_1)}{2T}} + \sqrt{\frac{k_2 \log(k_2)}{2T}}.$$

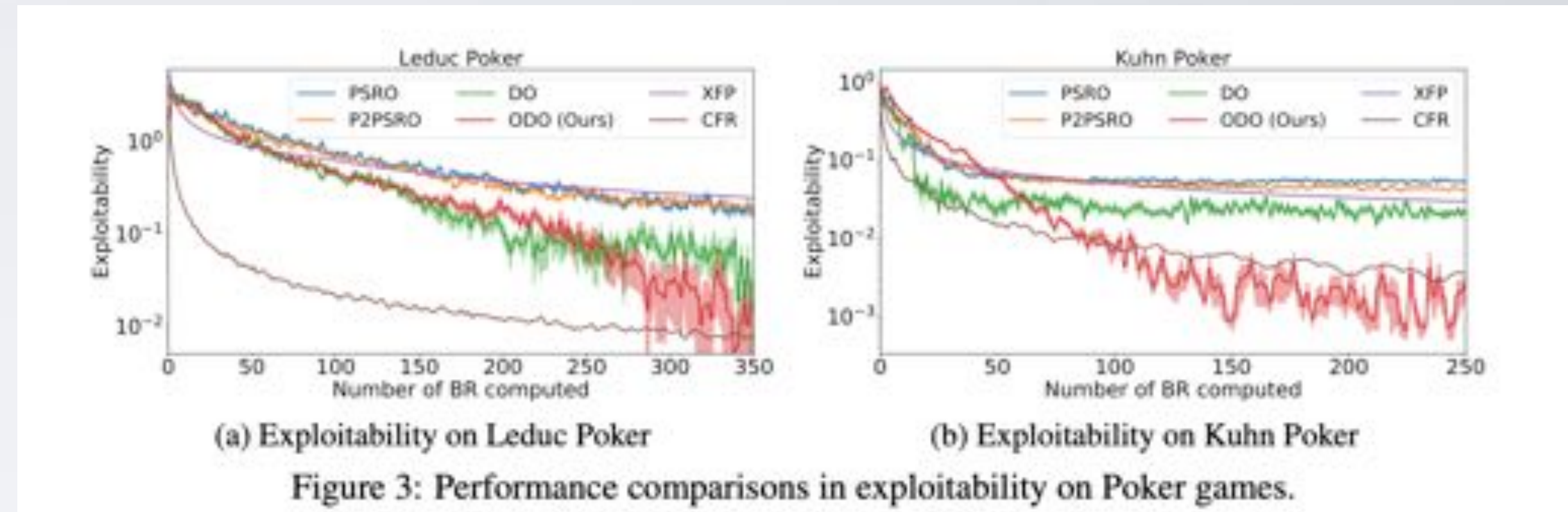
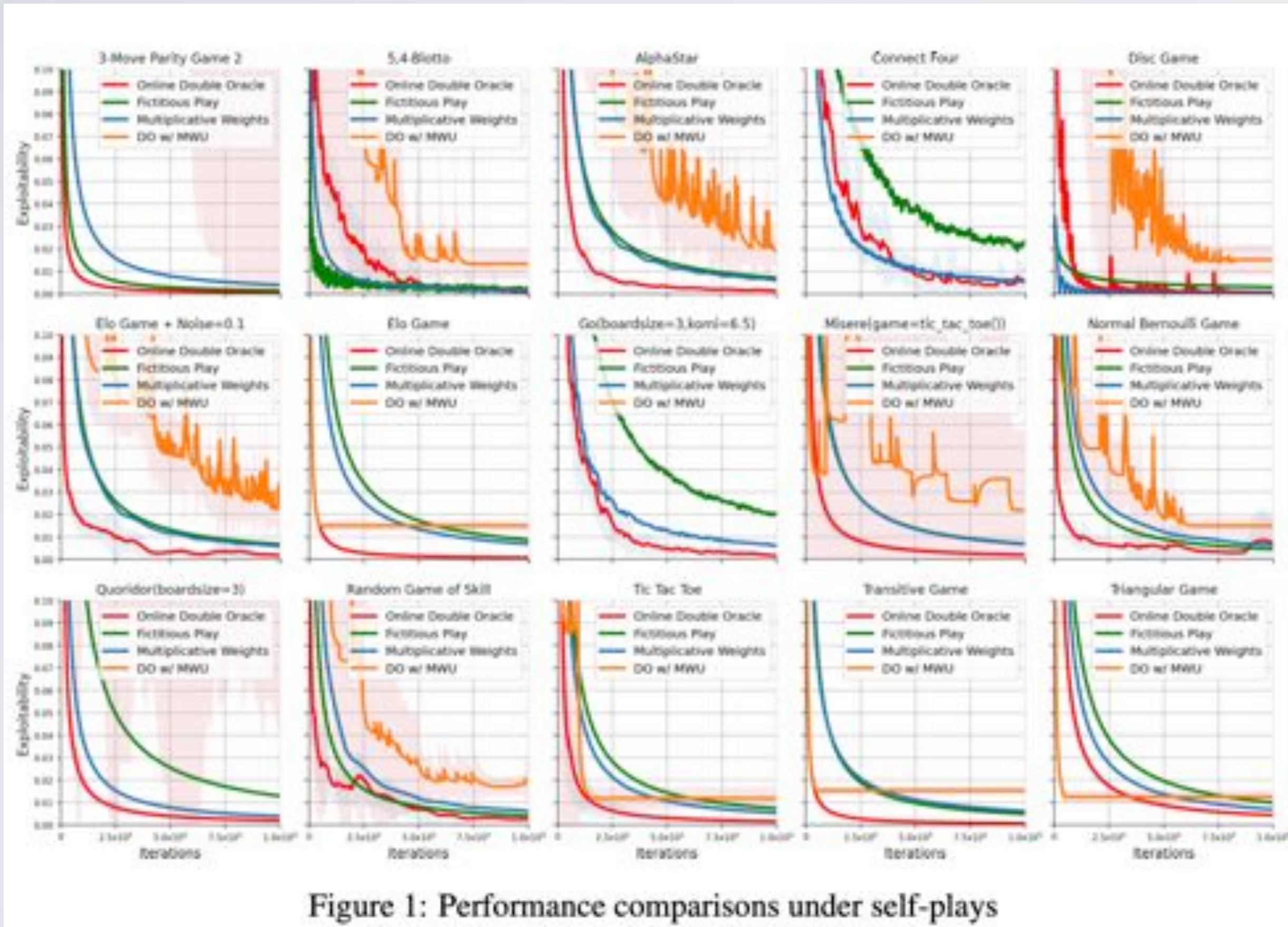
*In situation where both players follow OSO with Less-Frequent Best Response in Equation (6) and  $\alpha_{t-|T_i|}^i = \sqrt{t - |T_i|}$ , the convergence rate to NE will be*

$$\epsilon_T = \sqrt{\frac{k_1 \log(k_1)}{2T}} + \sqrt{\frac{k_2 \log(k_2)}{2T}} + \frac{\sqrt{k_1} + \sqrt{k_2}}{\sqrt{T}}.$$

# Online Double Oracle: Merits from both worlds

Exploitability on the Spinning Top games

Exploitability on Poker



Play with an imperfect opponent



# XODO: ODO can be extended to solve Extensive-form Games

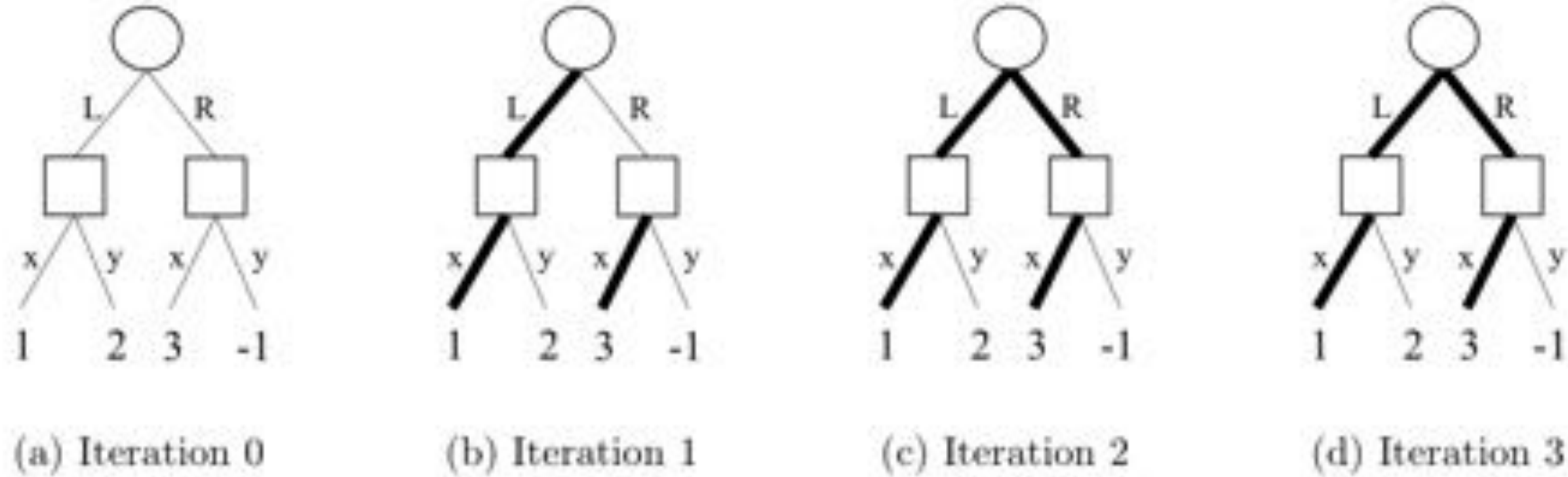
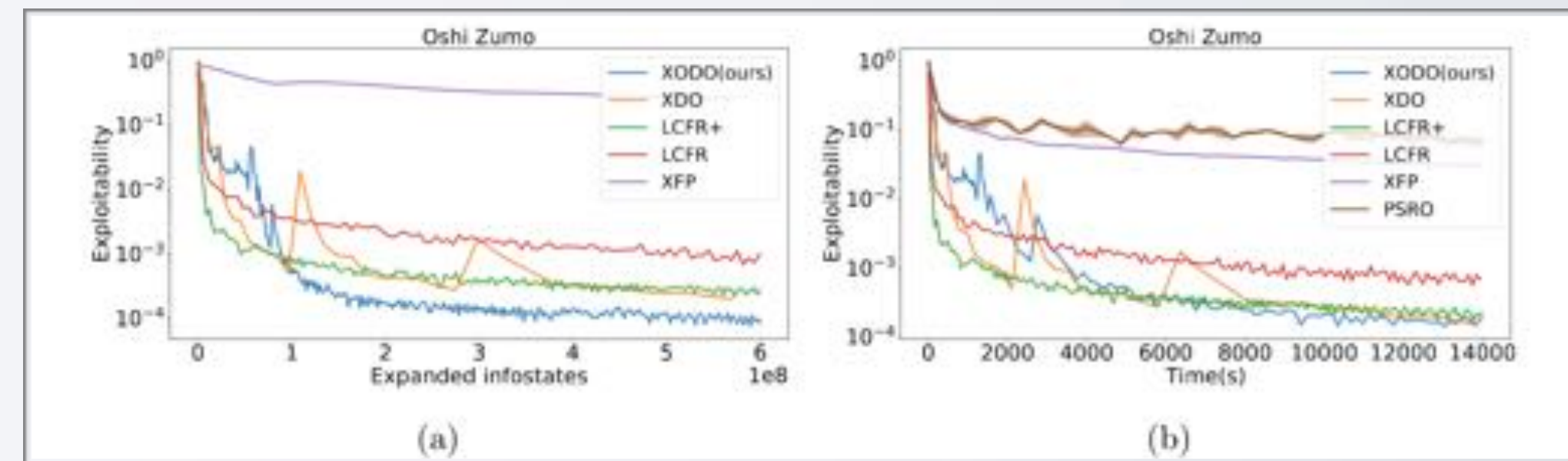
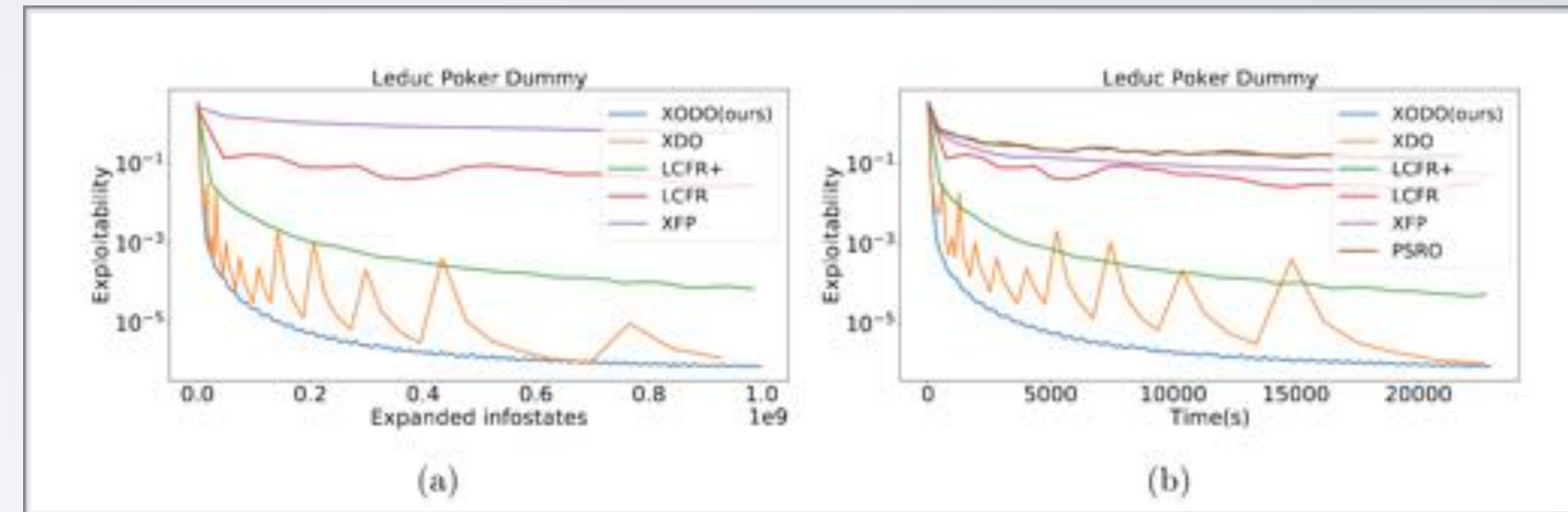
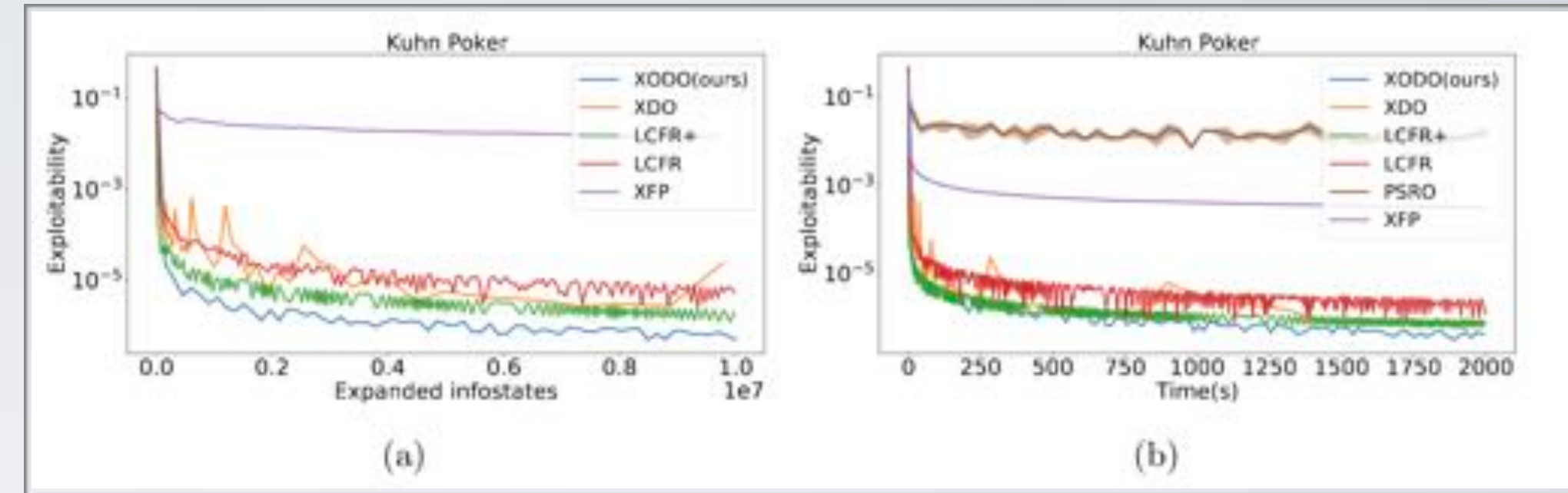


Figure 1: Example of three iterations of **XDO** in Zero-sum Extensive-Form Game. The square does not know what circle chose. The values at the leafs are circle's payoff, since it is zero-sum game, the square's payoffs are the opposite; At the beginning in graph (a), both circle and square have empty population and uniform random strategy. In graph(b), circle's BR to opponent is action  $L$ . Square's BR is action  $x$ . They are added into the population(thick line). Since both players only have one action, so the NE in the restricted game is choosing them. Then in graph(c), action  $R$  of circle is BR to square's action  $x$ , so it will be added into the population. We then compute the meta-NE of the restricted game constructed by the thick lines and repeat the above steps. Since all the best responses to the meta-NE are already in the population, the restricted game won't change in graph (d).



# Summary of Online Double Oracle Results

- The best achievable regret in bandit setting is  $\mathcal{O}(\sqrt{T|A|})$ , see [Audibert, Bubeck 2010, JMLR]

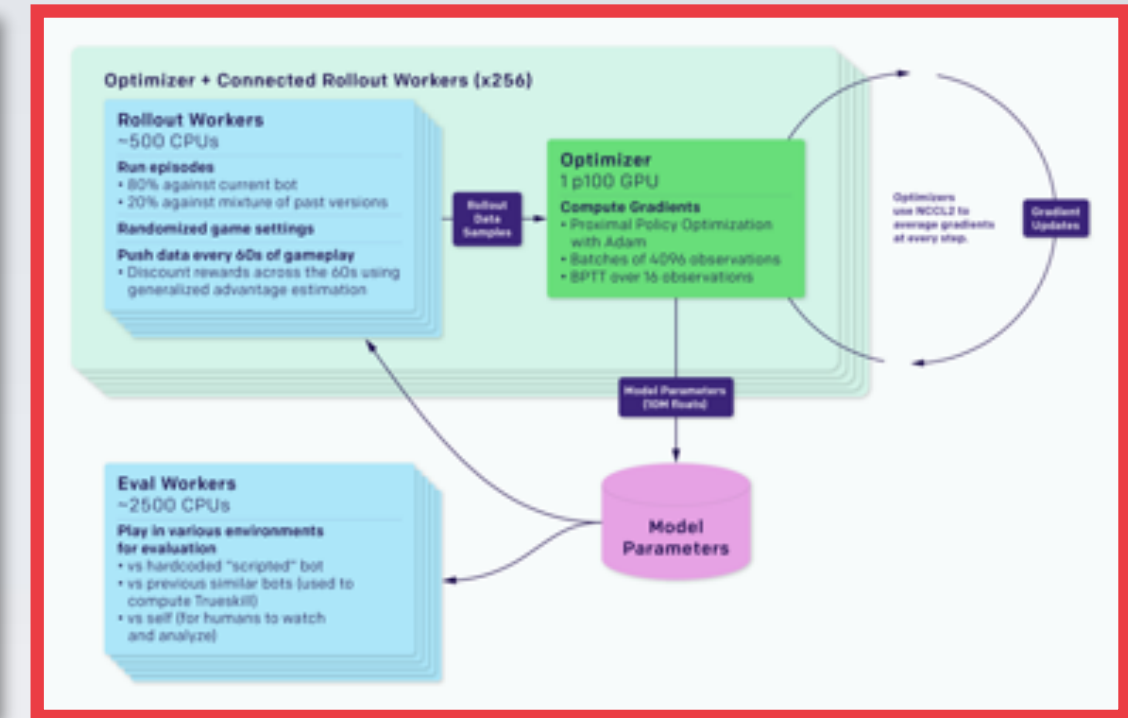
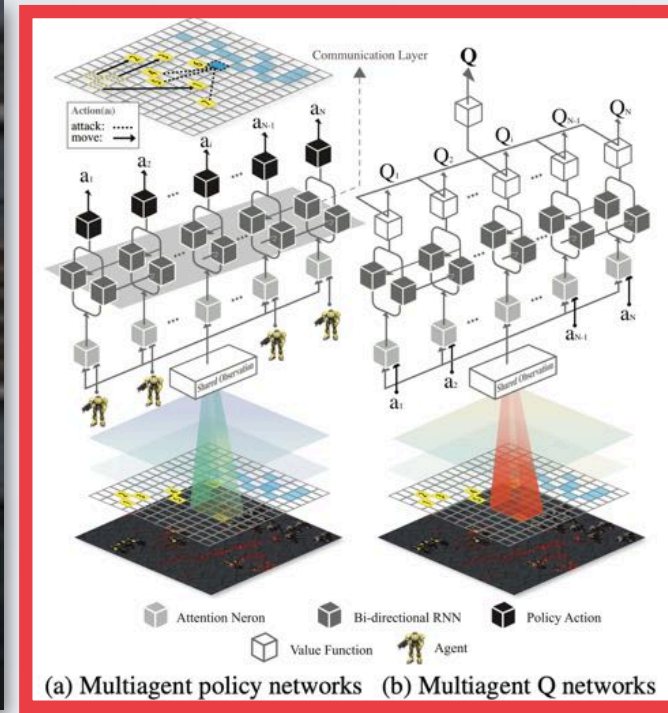
Table 1: Properties of existing solvers on two-player zero-sum games  $A_{n \times m}$ . \*:DO and XDO in the worst case has to solve all sub-games till reaching the full game, so the time complexity is one order magnitude larger than LP (van den Brand, 2020) and CFR (Zinkevich et al., 2007). †: Since PSRO uses approximate best-response, the total time complexity is unknown. ‡ Note that the regret bound of ODO can not be directly compared with the time complexity of DO, which are two different notions.

Method	Rational (No-regret)	Allow $\epsilon$ -Best Response	Convergence Rate ( $\tilde{\mathcal{O}}$ )	Regret Bound ( $\mathcal{O}$ )	Large Games
Linear Programming			$\tilde{\mathcal{O}}(n \exp(-T/n^{2.38}))$		
(Generalised) Fictitious Play		✓	$\tilde{\mathcal{O}}(T^{-1/(n+m-2)})$		
Multiplicative Weight Update	✓			$\mathcal{O}(\sqrt{\log(n)/T})$	
Double Oracle			$\tilde{\mathcal{O}}(n \exp(-T/n^{3.38}))^*$		✓
Counterfactual Regret Minimization	✓			$\mathcal{O}(\Delta I \sqrt{T A })$	✓
Extensive-Form Double Oracle		✓	$\tilde{\mathcal{O}}(\Delta I \sqrt{ A ^2/T})^*$		✓
Policy Space Response Oracle		✓			✓
<b>Online Double Oracle</b>	✓	✓		$\mathcal{O}(\sqrt{k \log(k)/T})^\dagger$	✓
<b>Extensive-Form Online Double Oracle</b>	✓	✓		$\mathcal{O}(\Delta S_i^* \sqrt{k A_i^* /T})$	✓

# Contents

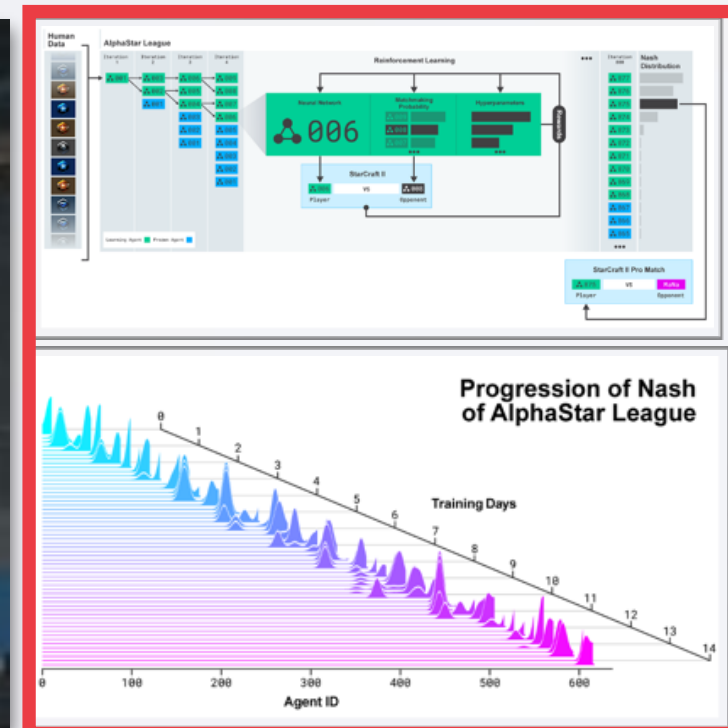
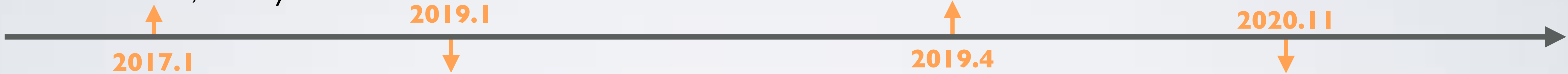
- **Formulation & Challenges of Training A Population of RL Agents**
- **Training A Population of RL Agents on Fully-Cooperative Games**
- **Training A Population of RL Agents on Zero-Sum Games**
- **Some System Level Thinkings**
- **Conclusions**

# Training Population of RL Agents Require Powerful AI Systems

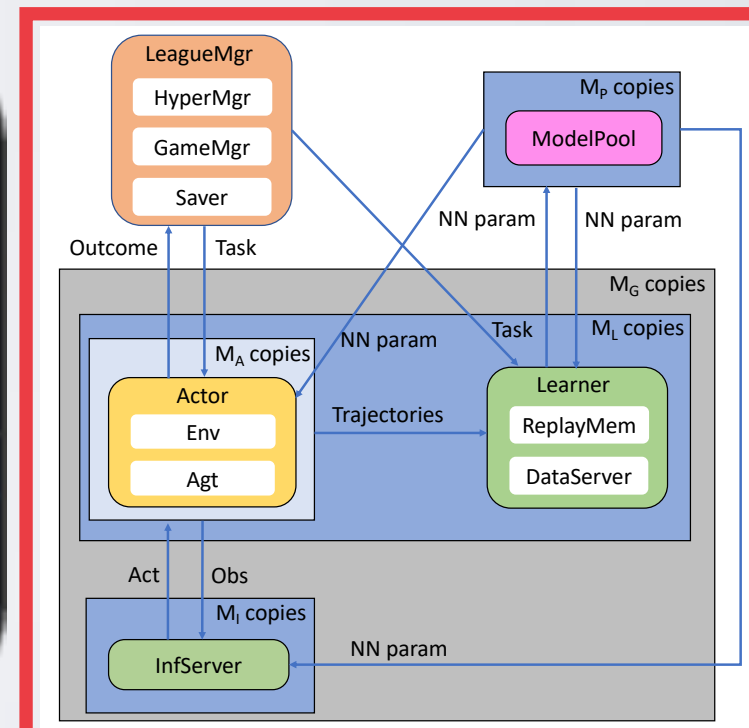


StarCraft micro-management  
BiCNet, deep MARL methods  
**1-2 GPUs, 1-2 days**

Dota 2 full game (OpenAI Five)  
Population-based + Rapid training system  
**128,000 CPUs, 100 GPUs, 180 years of plays per day**



StarCraft full game (AlphaStar)  
Populating-based Training  
Training for single agent costs **14 days, 16 TPU/Agent, 200 years of real-time play.**



王者荣耀 (绝悟)  
Populating-based Competitive Self-play + Policy distillation  
**35,000 CPUs, 320 GPUs, begin to converge after 336 hours**



# Training Population of RL Agents Require Powerful AI Systems

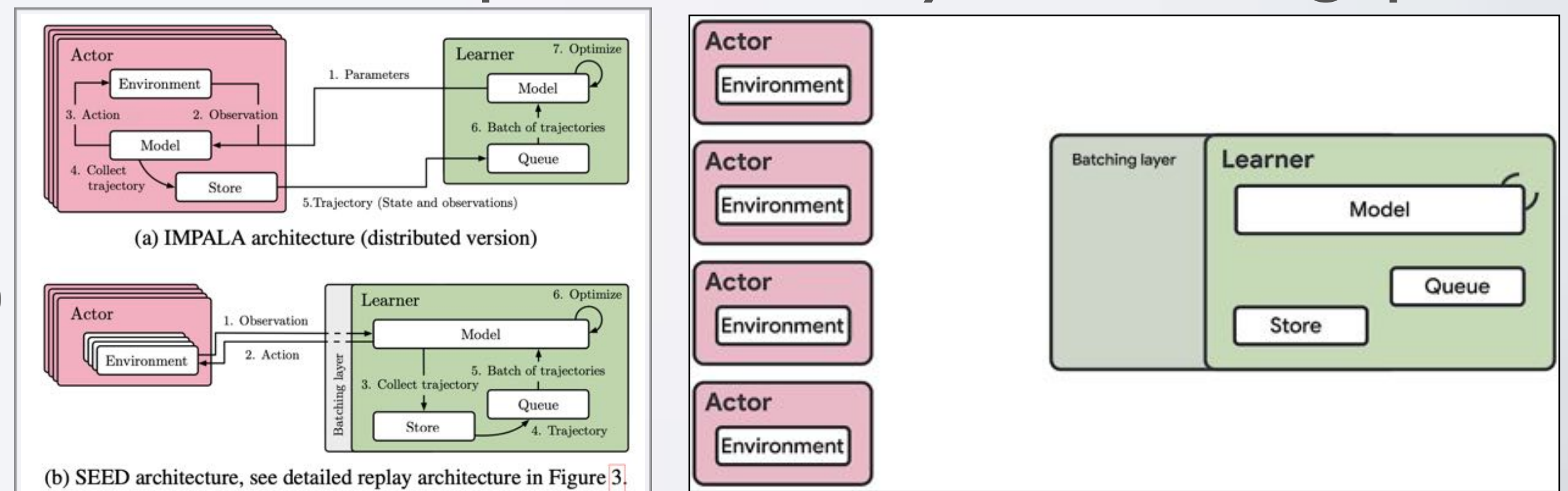
- Distributed RL systems has made substantial progress nowadays.
  - ◆ A rule-of-thumb is that: **decoupling learning and rollout enables great speedups.**



<https://cloud.tencent.com/developer/article/1119569>

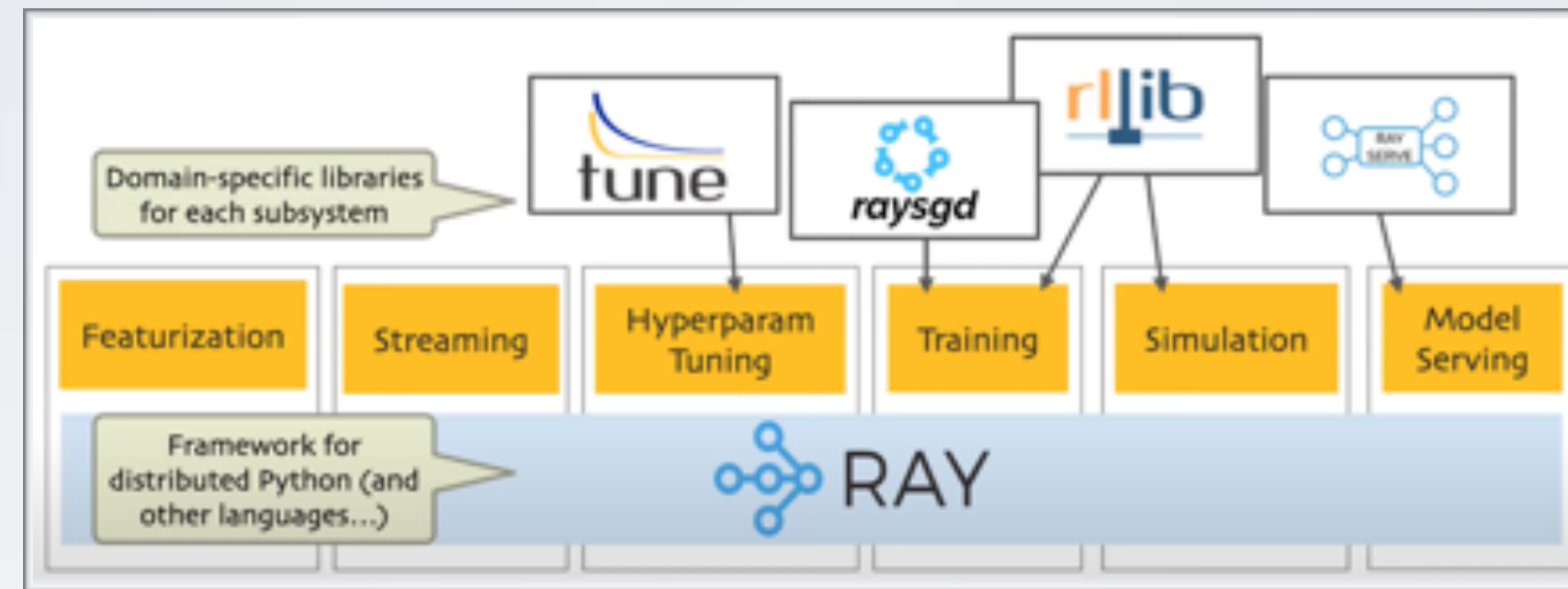
- SEED RL implements a highly scalable IMPALA that uses batched inference and optimisation on a single accelerators to maximise compute efficiency and throughput.

- ◆ Both training and inference are on the GPU
- ◆ Actors only run the environments
- ◆ Split env step (actor) and inference step (model)
- ◆ Employ RPCs between actor and learner

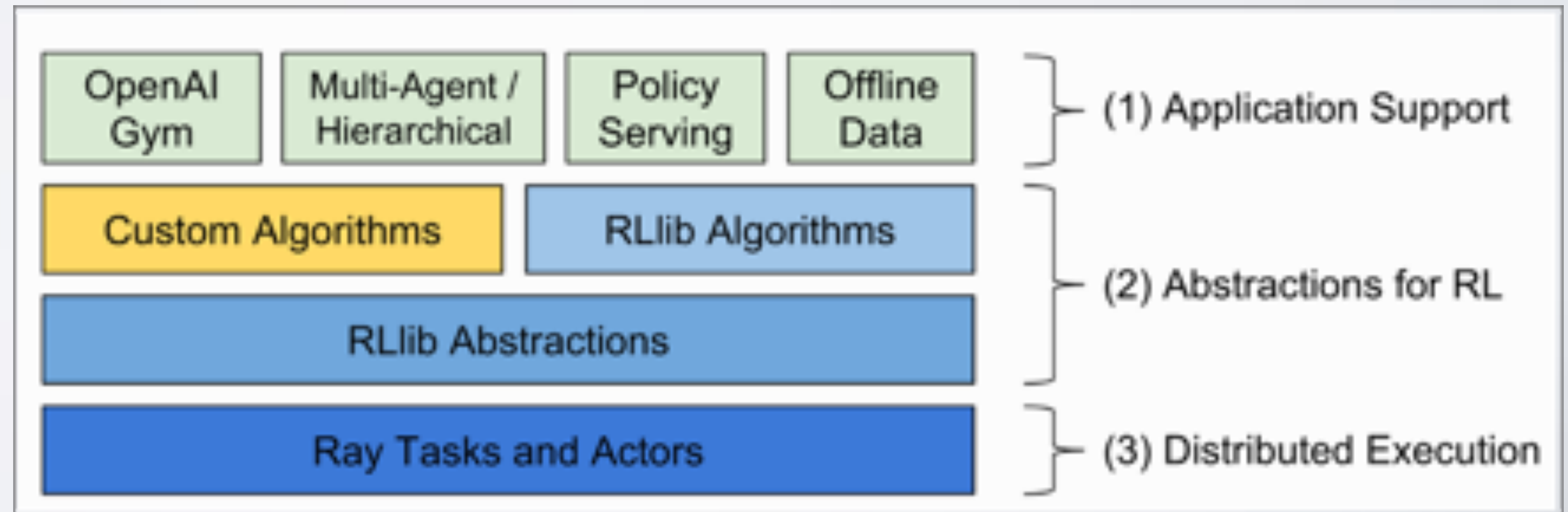
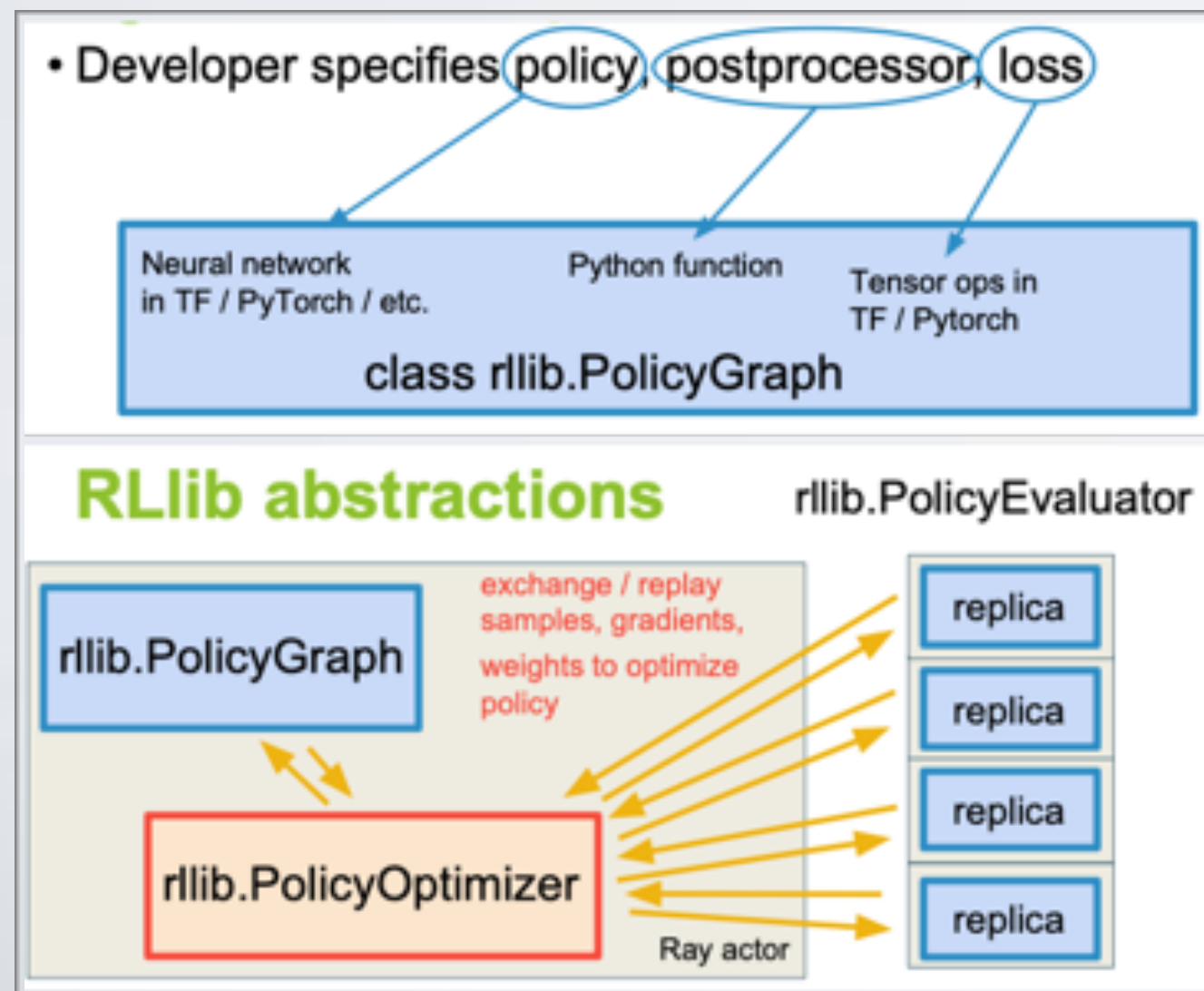


# Training Population of RL Agents Require Powerful AI Systems

- Distributed RL systems has made substantial progress nowadays.
  - ◆ RAY is an ecosystem that help build distribution applications.



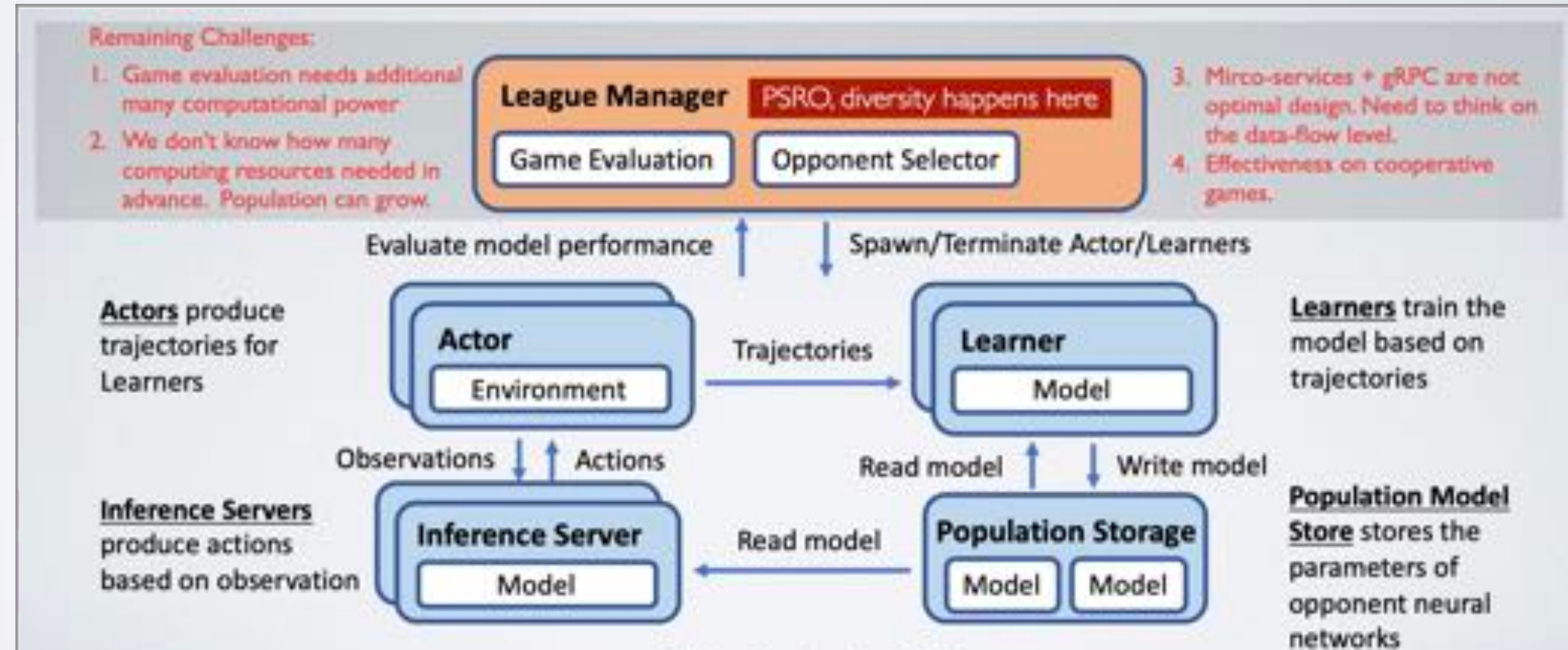
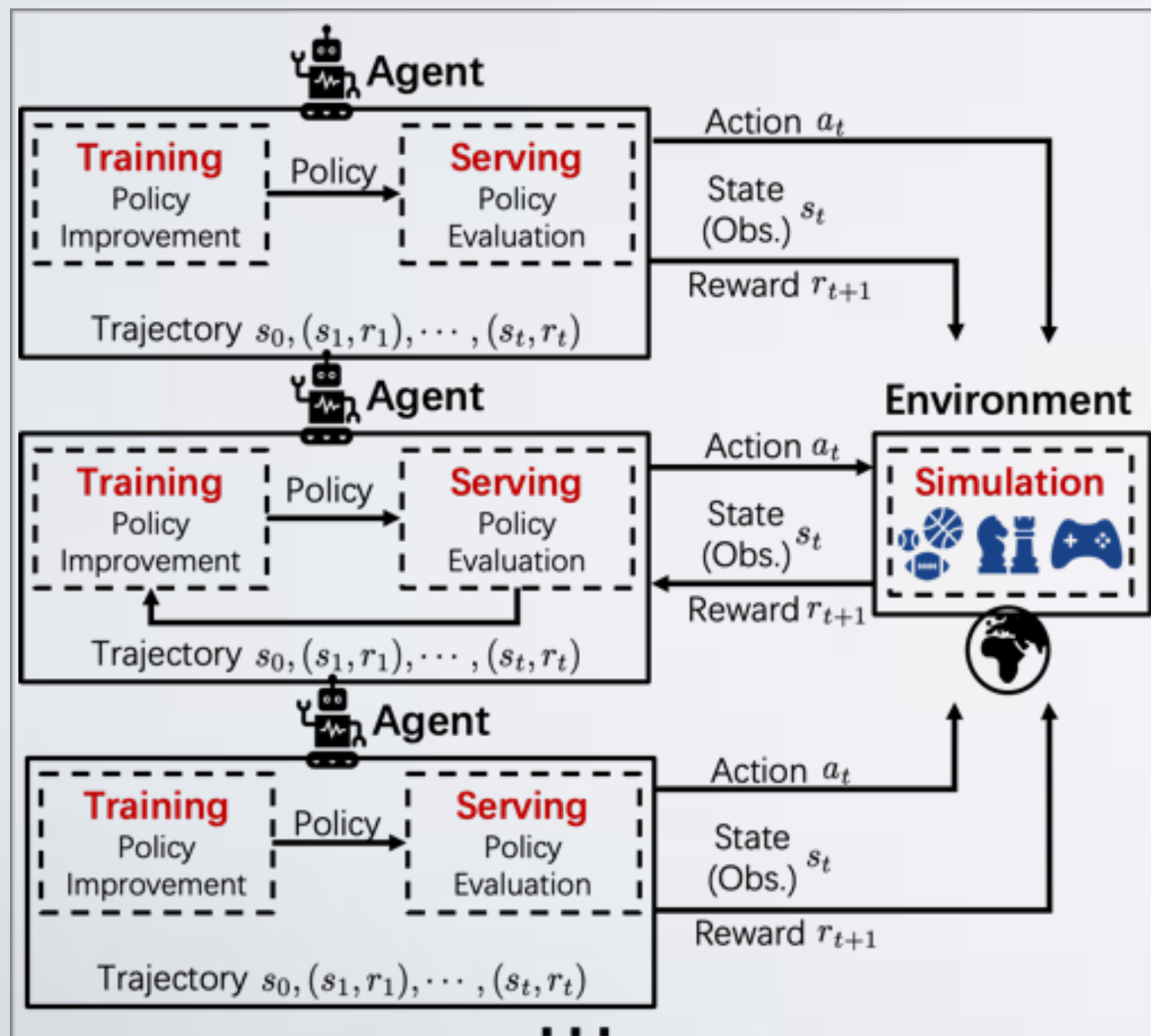
- ◆ The implementation of RLib uses RAY for RL applications.



# PB-MARL Poses New Requirements for AI Systems

- PB-MARL requires more thinkings on the efficient implementations
  - ◆ Support for distributed PB-MARL is limited.

Framework	Single-Agent	Multi-Agent	Population Management
RLlib	✓	✓	×
SeedRL	✓	×	×
Sample-Factory	✓	✓	×
MALib	✓	✓	✓



# PB-MARL Poses New Requirements for AI Systems

- PB-MARL requires more thinkings on the efficient implementations
  - computational demands for RL
    - **Policy optimization**
    - **Policy inference**
    - **Environment simulation & rollout**

These 3 subtasks can be easily implemented, e.g., Actor-Learner Architecture

- **Extra demands for MARL**
  - Rollout with joint policies
  - Exponentially increased rollouts for training and evaluation
  - Complicated management of policy pool and population
  - More complex task- & data-flow
  - Meta-game -> extra policy interactions
  - ...

There is not a good solution to these requirements.

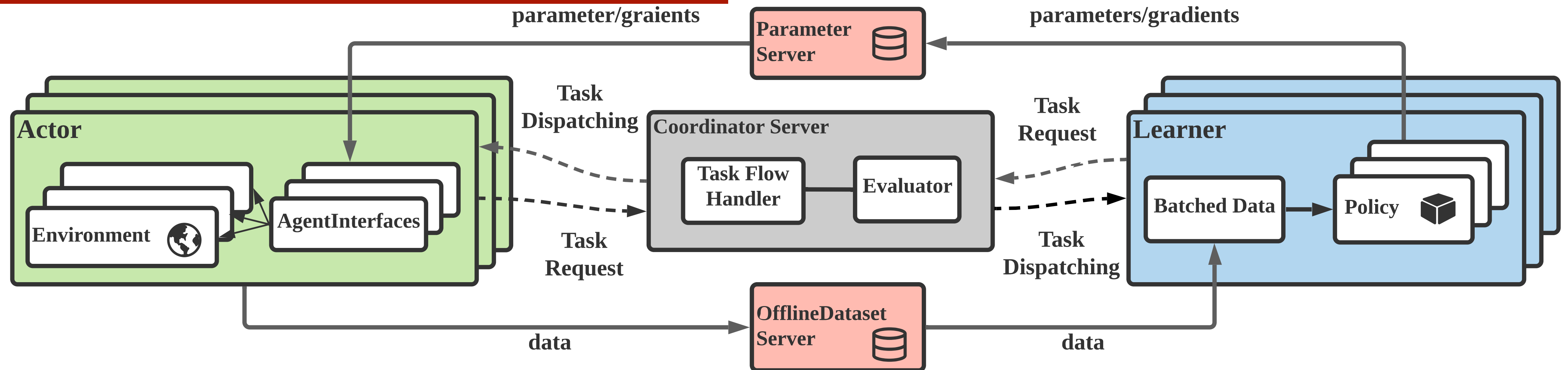
## For Population-based MARL

- **Large-scale rollout & training framework**
  - Support heterogenous tasks
  - High throughput distributed frameworks
  - Maximize utilization of different hardware
- **Support population-based training**
  - self-play / league mechanism
  - meta-game analysis that based on game/graph theory and representation learning.
  - Others: imitation learning/transfer learning/model based/heuristic

# MALib: Designed for PB-MARL

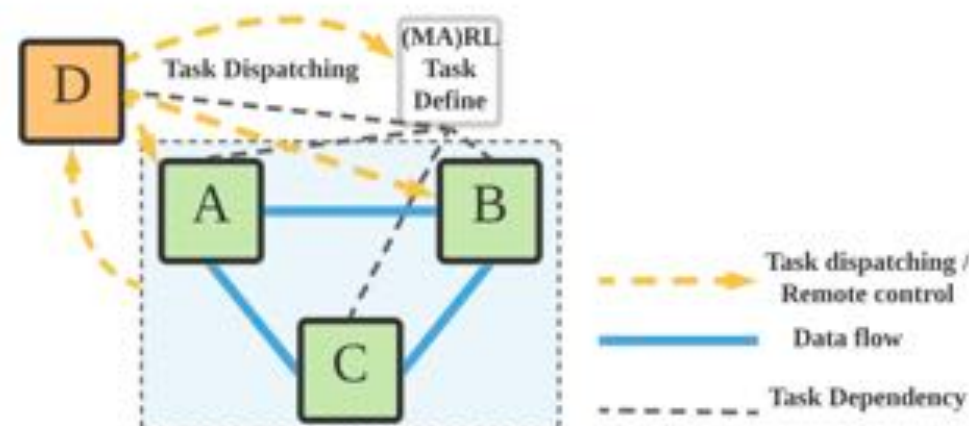
- MALib: <https://malib.io>

## Actor-Evaluator-Learner Architecture: Decouple the Task-/Data-flow



### MALib: Centralized Task Dispatching Model (CTD)

- Modular design, easy to implement and reuse.
- Centralized control: generate tasks and allocate compute resource dynamically.
- Semi-passively executing submodule



In MALib, we focus on training paradigms and then build algorithms upon them, which improve the reusing rate.

- Independent Learning: DQN, PPO
- Centralized Learning: MADDPG, QMIX
- Async Learning IMPALA
- Sync Learning: ...

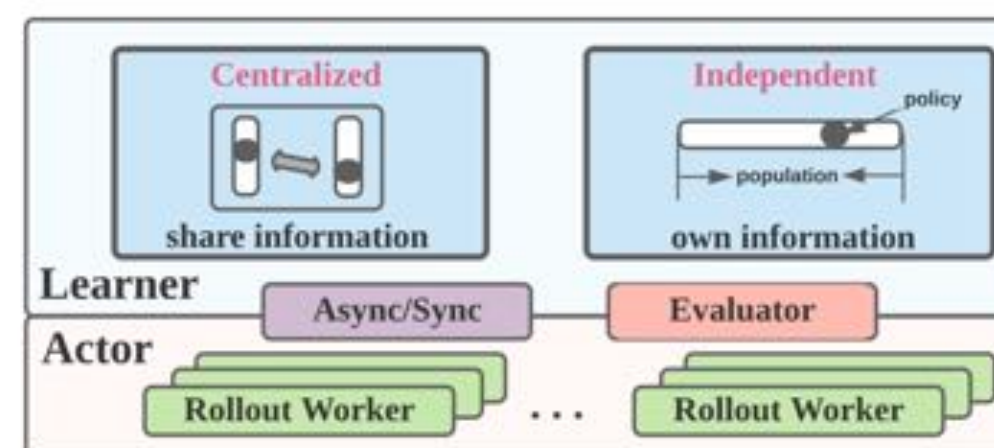


Table 4: Implemented algorithms in MALib.

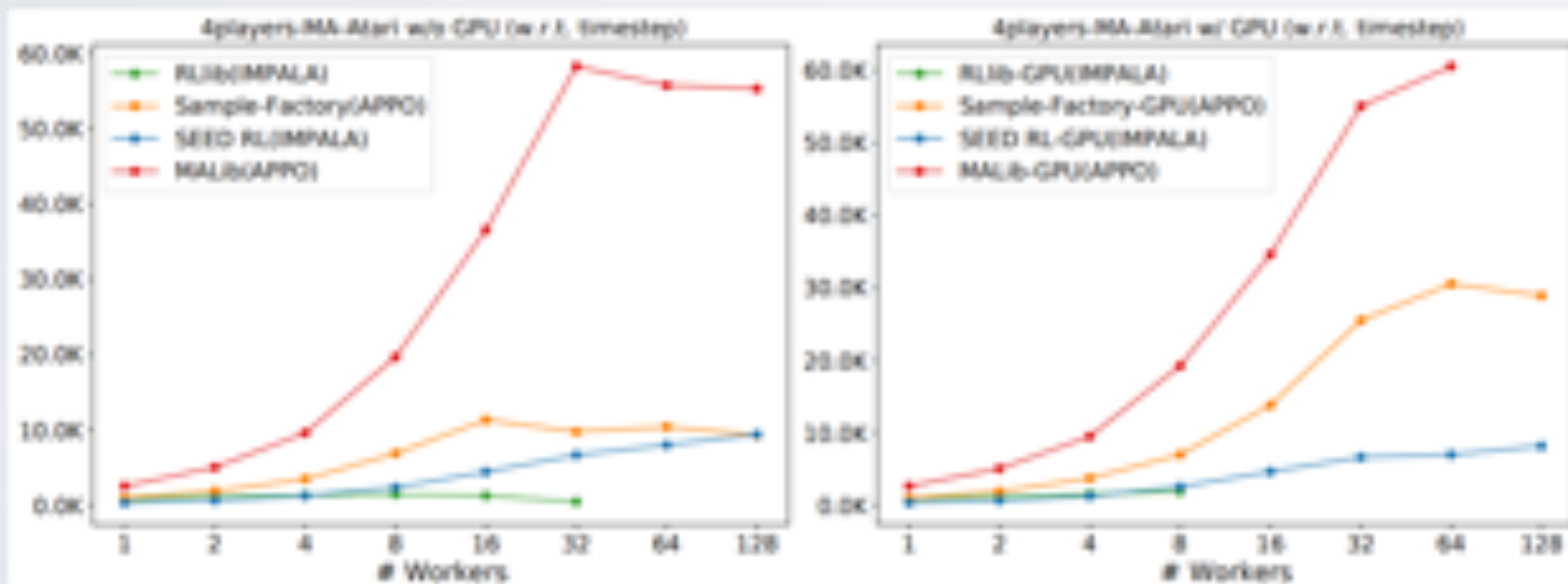
Algorithm	Training Interface	Execution Mode	PB-MARL Support
DQN [42]	Independent	Async/Sync	PSRO/FSP/SP
Gorilla [25]	Independent	Async	PSRO/FSP/SP
A2C [45]	Independent	Sync	PSRO/FSP/SP
A3C [26]	Independent	Async	PSRO/FSP/SP
SAC [49]	Independent	Async/Sync	PSRO/FSP/SP
DDPG [50]	Independent	Async/Sync	PSRO/FSP/SP
PPO [43]	Independent	Sync	PSRO/FSP/SP
APPO	Independent	Async	PSRO/FSP/SP
MADDPG [16]	Centralized	Async/Sync	PBT
QMIX [17]	Centralized	Async/Sync	PBT
MAAC [46]	Centralized	Async/Sync	PBT

# MALib has high throughput

- Single machine: 32 CPUs, 2× RTX 3090 ;
- Multiple machines: 2× 128 CPUs、 256 GB
- Environments: MA-Atari, StarCraft II (SMAC) ;
- Compared Frameworks: RLlib, SEED RL, Sample Factory

## single machine

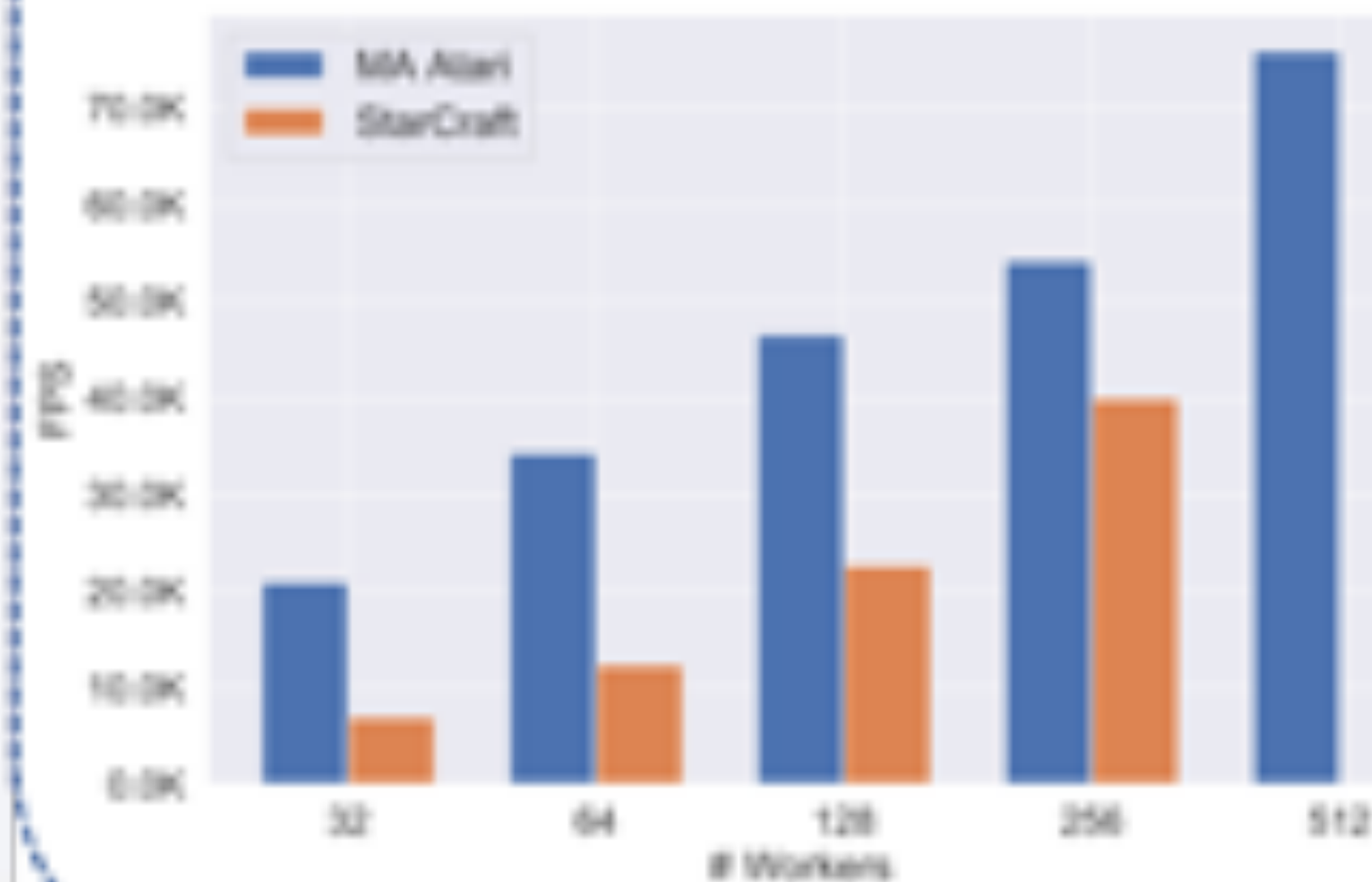
1. Outperforms **all** compared frameworks.



(1) CPU-only

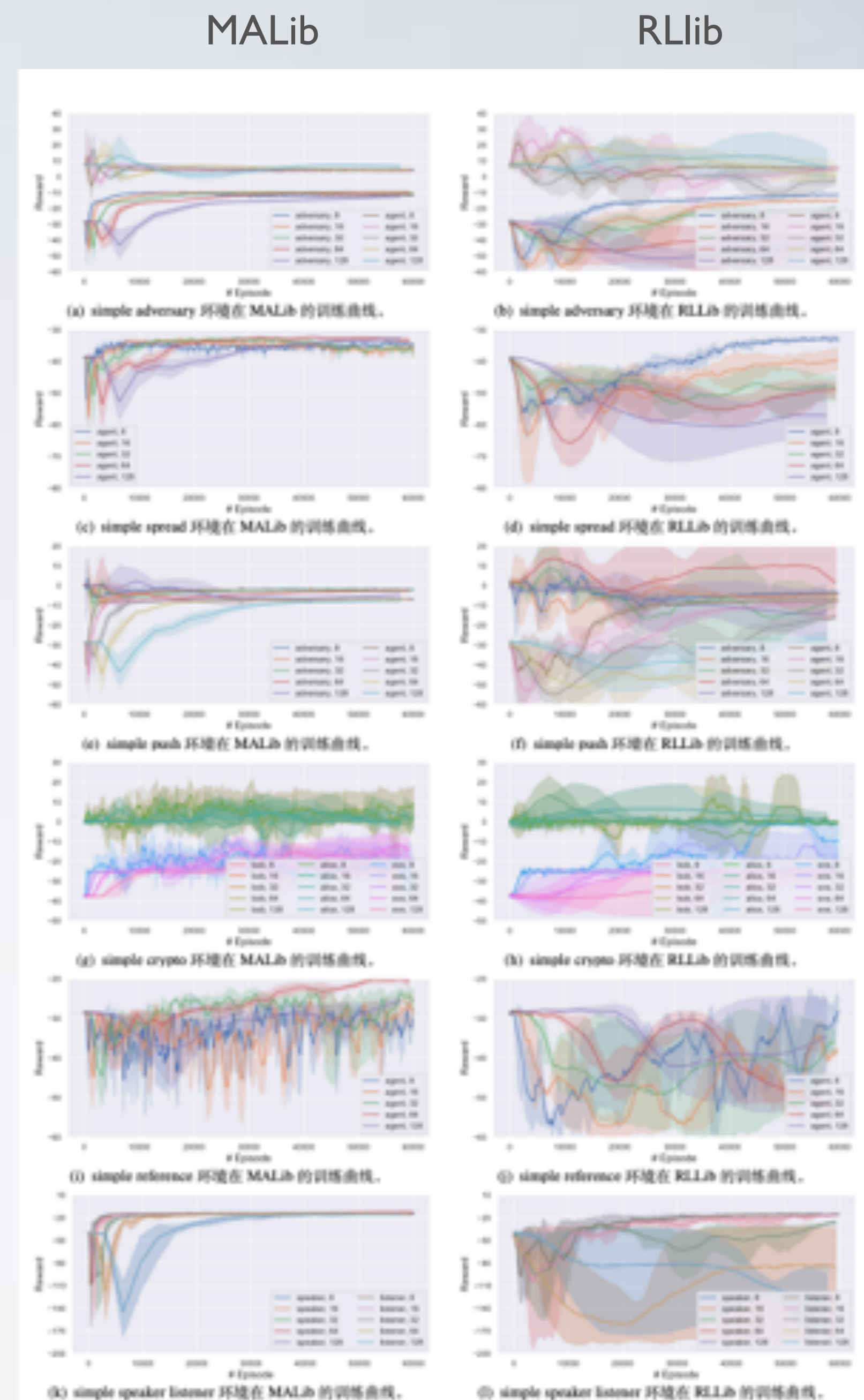
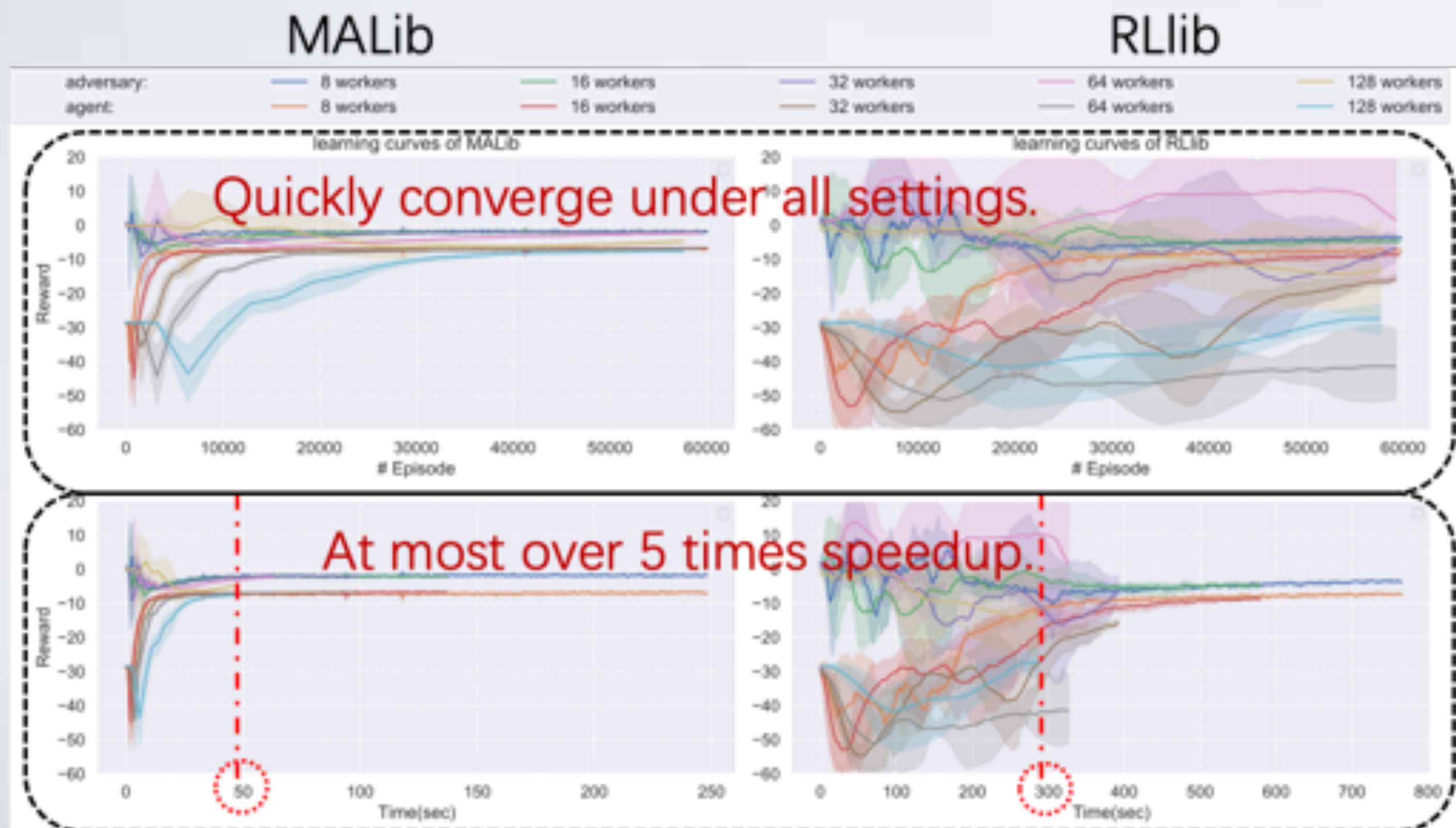
(2) GPU accelerated

Multiple machines  
Throughput continues to increase. Scalability ✓



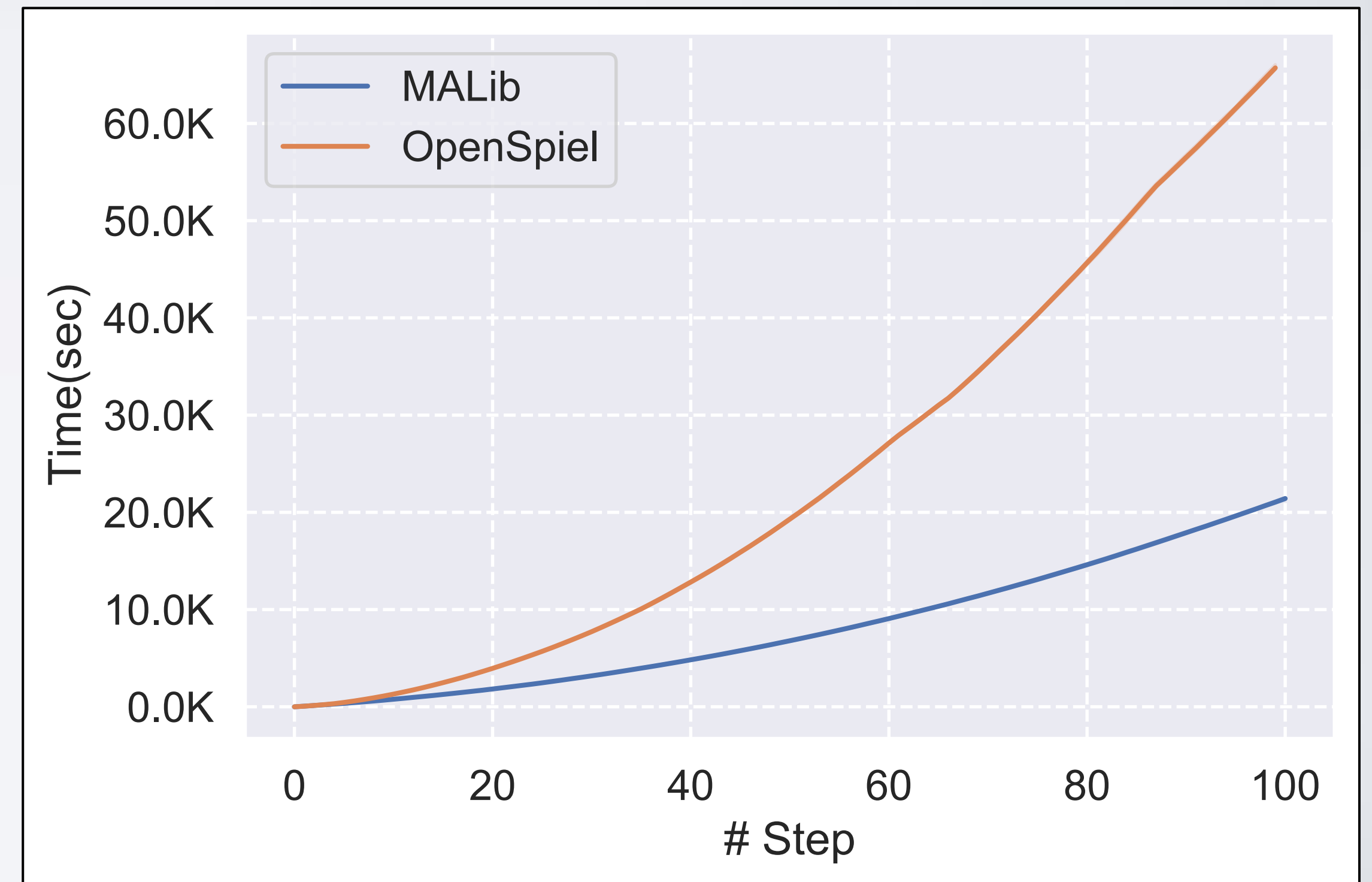
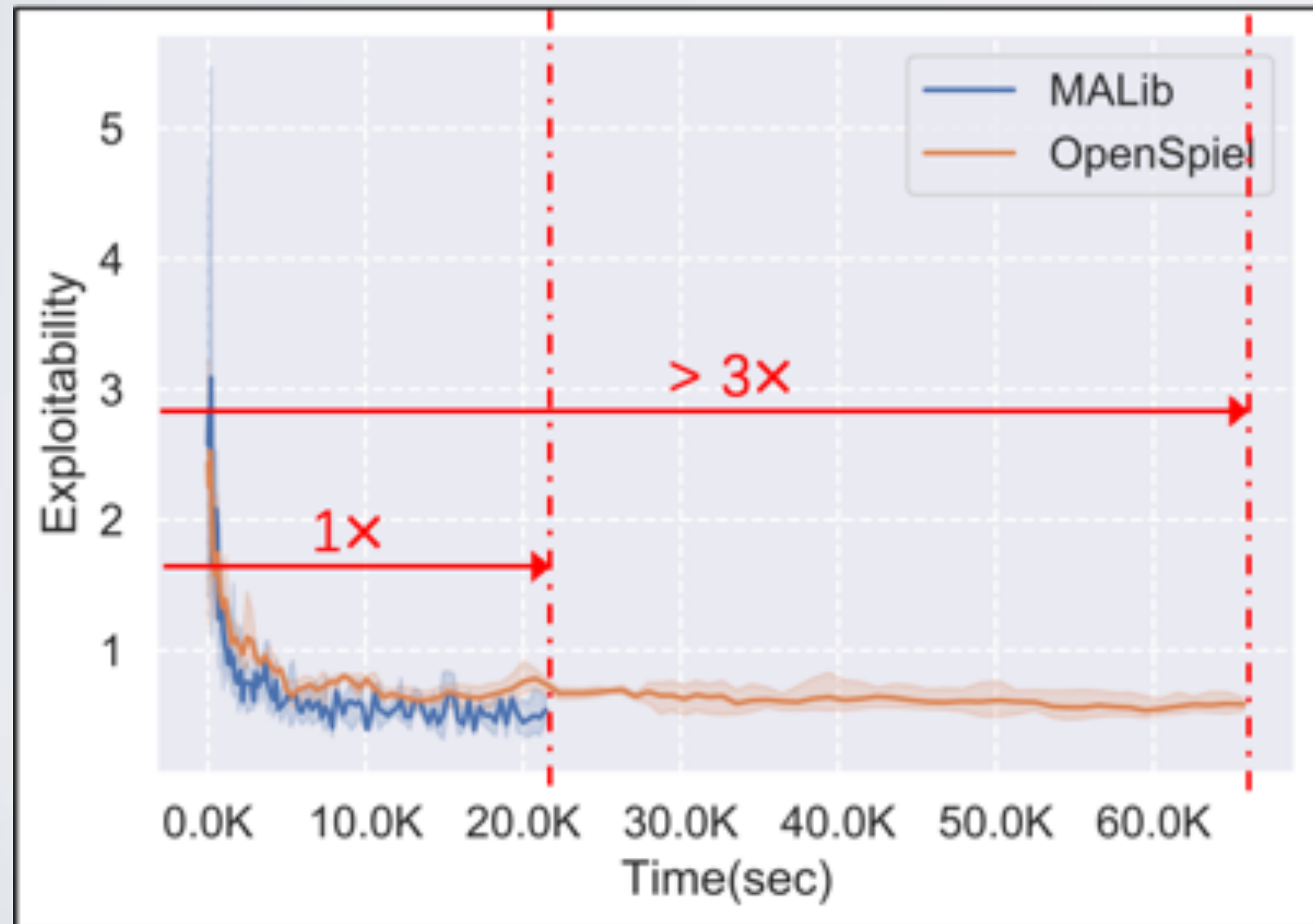
# MALib is approximately **5x faster** than RLib

- Environments: six scenarios in MPE;
- Compared framework: RLib;
- Algorithm : MADDPG;
- Settings: different #rollout workers;
- Metrics: convergence, execution time.



# MALib is approximately **3x faster** than OpenSpiel

- Environment: Leduc Poker ;
- Compared framework: OpenSpiel ;
- Algorithm: PSRO ;
- Metrics: Exploitability, execution time.





# Conclusions

## ● **Formulation & Challenges of Training A Population of RL Agents**

- ◆ Compute Nash is PPAD

## ● **Training A Population of RL Agents on Fully-Cooperative Games**

- ◆ IGM can fail in cooperative problems
- ◆ MAPG has large variance, one can think of using min-variance optimal baseline
- ◆ Advantage decomposition lemma, MA-TRPO, MA-CPO methods are the big hopes

## ● **Training A Population of RL Agents on Zero-Sum Games**

- ◆ Meta-game level thinkings are important due to the non-transitivity issue
- ◆ PSRO methods and its variations can deal with non-transitivity
- ◆ Online Double Oracle take the merits from both population-based methods and no-regret methods

## ● **Some System Level Thinkings**

- ◆ Training population of MARL agents poses new challenges for existing AI system
- ◆ Try MAlib!