CoG 2022 Tutorial

SOLVING TWO-PLAYER ZERO-SUM GAMES THROUGH REINFORCEMENT LEARNING

Dr. Yaodong Yang (Peking University)
Le Cong Dinh (University of Southampton)

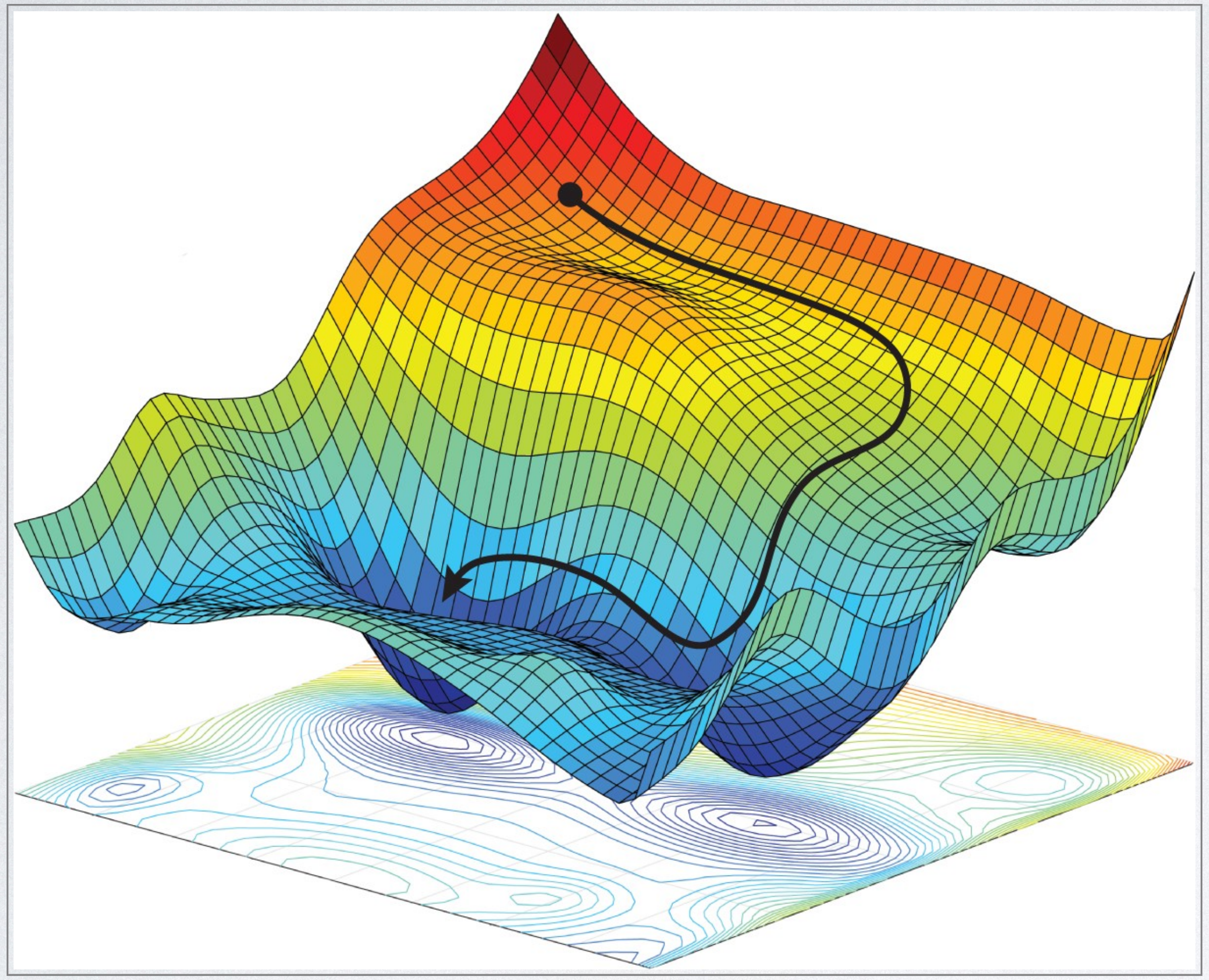
08/2022

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

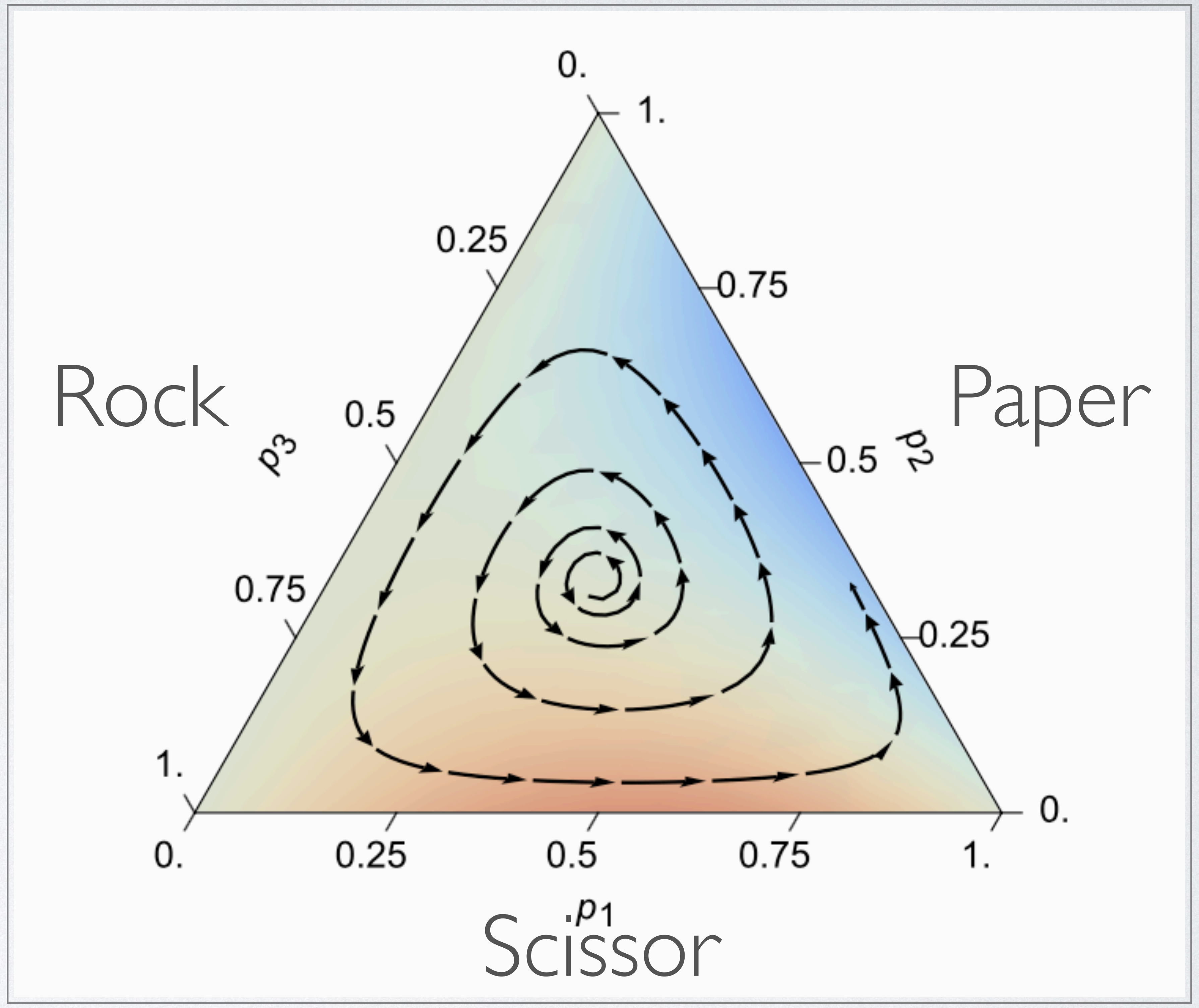
Normal machine learning problems:



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

Loss landscape keeps fixed

Multi-agent Learning problems:

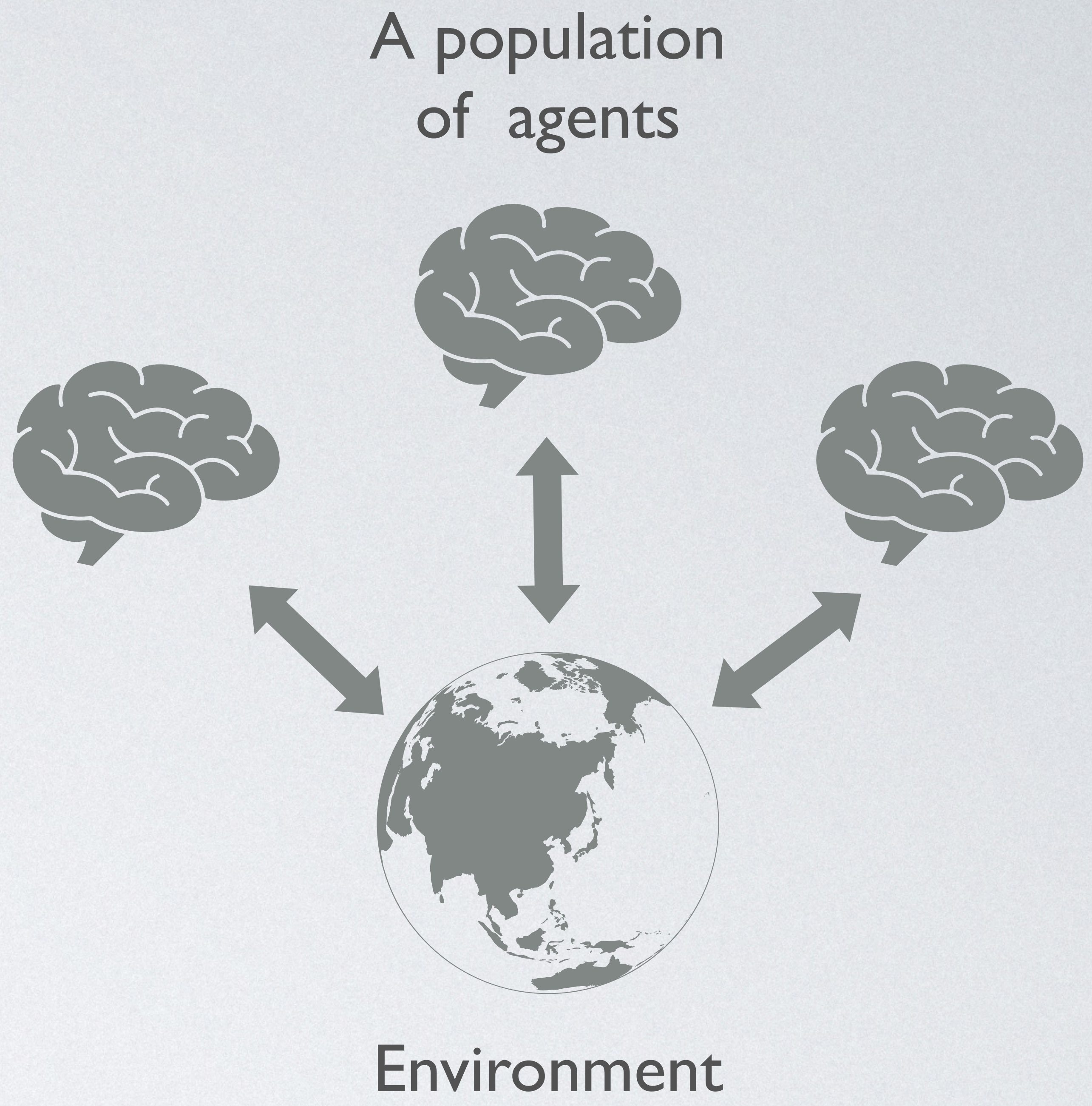


agents learn to reach some dynamic equilibrium

Loss landscape changes with opponents' actions

Multi-Agent Reinforcement Learning

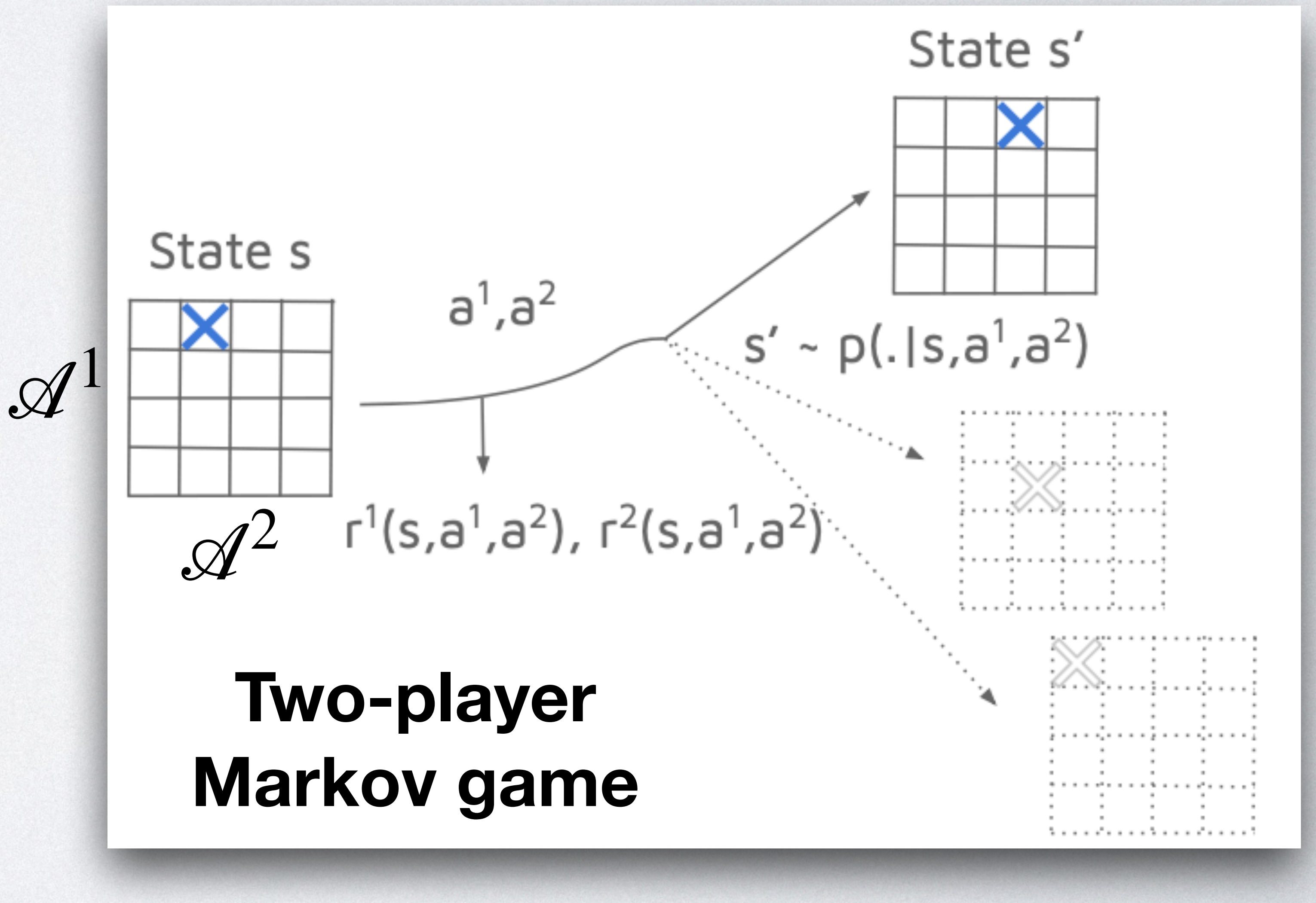
- Modelled by a Stochastic Game $(\mathcal{S}, \mathcal{A}^{\{1,\ldots,n\}}, \mathcal{R}^{\{1,\ldots,n\}}, \mathcal{T}, \mathcal{P}_0, \gamma)$
  - $\mathcal{S}$ denotes the state space,
  - $\boldsymbol{\mathcal{A}}$ is the joint-action space $\mathcal{A}^1 \times \ldots \times \mathcal{A}^n$,
  - $\mathcal{R}^i = \mathcal{R}^i(s, a^i, a^{-i})$ is the reward function for the i-th agent,
  - $\mathcal{T} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \to [0,1]$ is the transition function based on the joint action,
  - $\mathcal{P}_0$ is the distribution of the initial state, γ is a discount factor.
  - **Special case:** $n = 1 \to$ single-agent MDP, $|\mathcal{S}| = 1 \to$ 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:

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

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



Two-player Markov game

Multi-Agent Reinforcement Learning

- Value-based method:
  - The sense of optimality changes, now it depends on other agents !

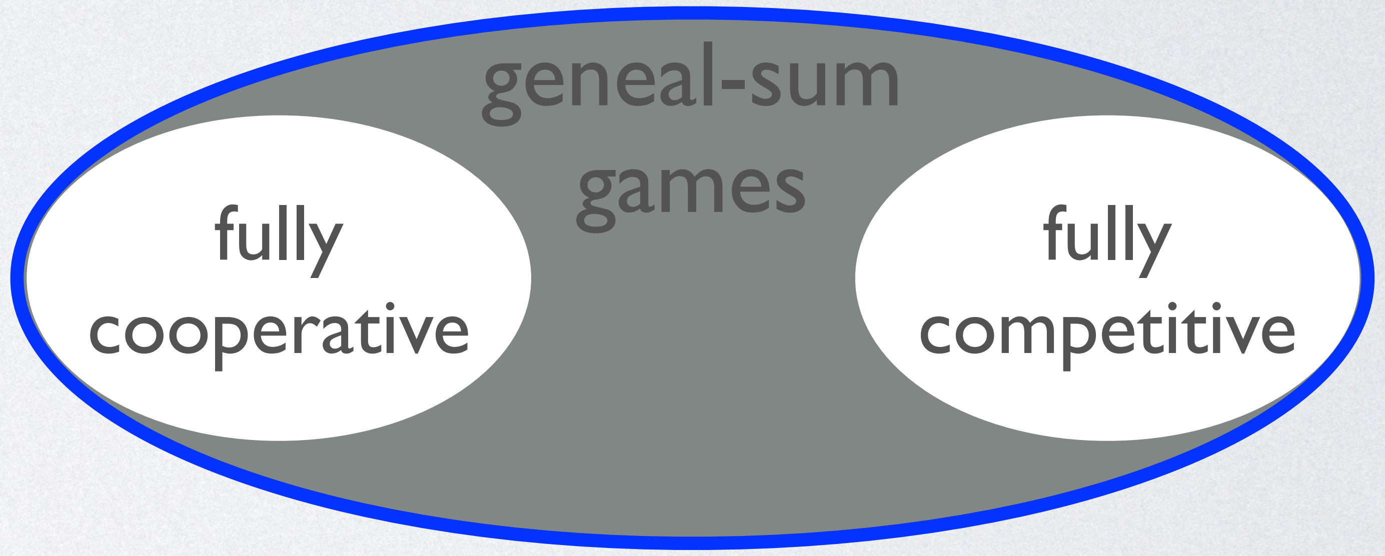
$$Q_{i,t+1}\left(s_k, \boldsymbol{\pi}_t\right) = Q_{i,t}\left(s_t, \boldsymbol{\pi}_t\right) + \alpha\Big[R_{i,t+1} + \gamma \cdot \mathbf{eval}_i\big\{Q_{\cdot,t}(s_{t+1}, \cdot\,)\big\} - Q_{i,t}\left(s_t, \boldsymbol{\pi}_t\right)\Big]$$

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

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

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

$$\mathbf{solve}_i\big\{Q_{\cdot,t}(s_t, \cdot\,)\big\} = \arg\max_{a_i}\Big(\max_{a^{-i}} Q_{i,t}(s_t, a_i, a_{-i})\Big)$$



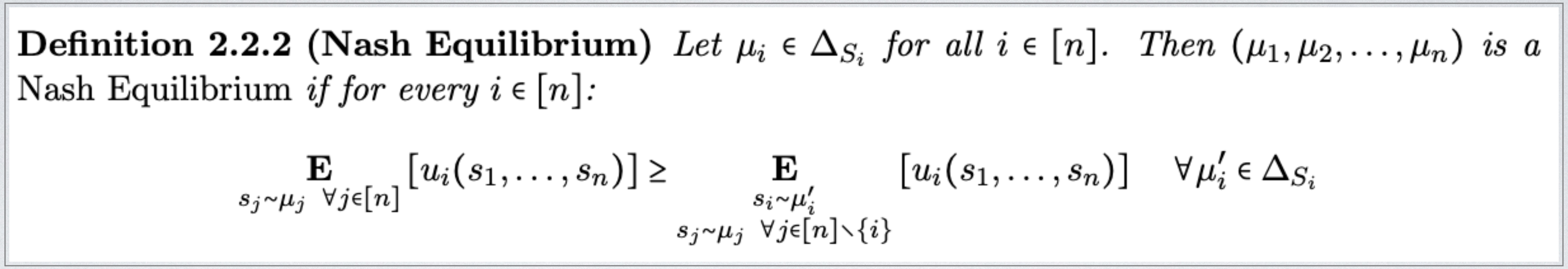
    - **Fully-competitive game**: sum of agents' reward is zero

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

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

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

Nash Equilibrium

- 

Definition 2.2.2 (Nash Equilibrium) *Let $\mu_i \in \Delta_{S_i}$ for all $i \in [n]$. Then $(\mu_1, \mu_2, \dots, \mu_n)$ is a* Nash Equilibrium *if for every $i \in [n]$:*

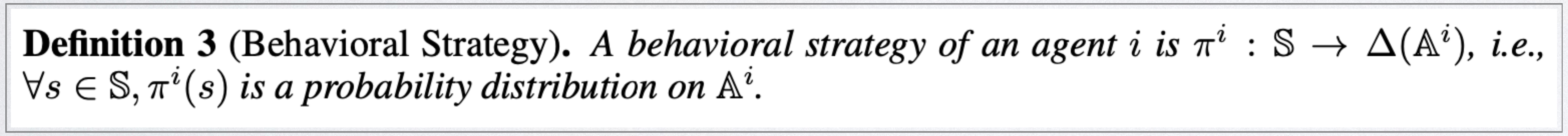
$$\mathop{\mathbf{E}}_{s_j \sim \mu_j \ \forall j \in [n]} [u_i(s_1, \dots, s_n)] \geq \mathop{\mathbf{E}}_{\substack{s_i \sim \mu_i' \\ s_j \sim \mu_j \ \forall j \in [n] \setminus \{i\}}} [u_i(s_1, \dots, s_n)] \quad \forall \mu_i' \in \Delta_{S_i}$$

- Mixed strategy Nash equilibrium always exists in finite player finite action games.

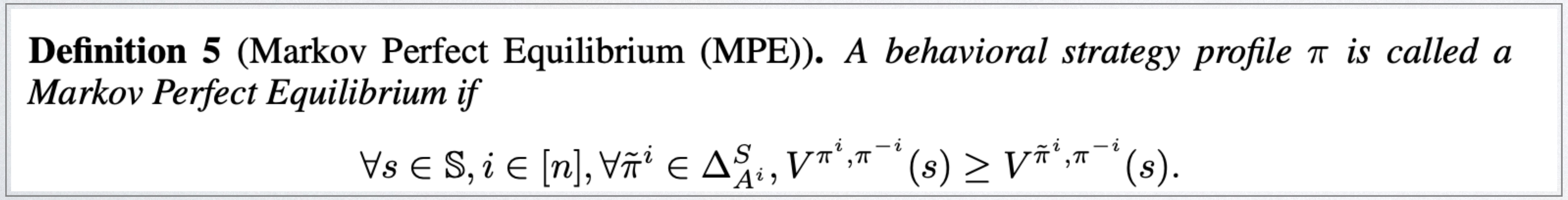
- For continuous utility games, the strategy set needs to be compact

- 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.

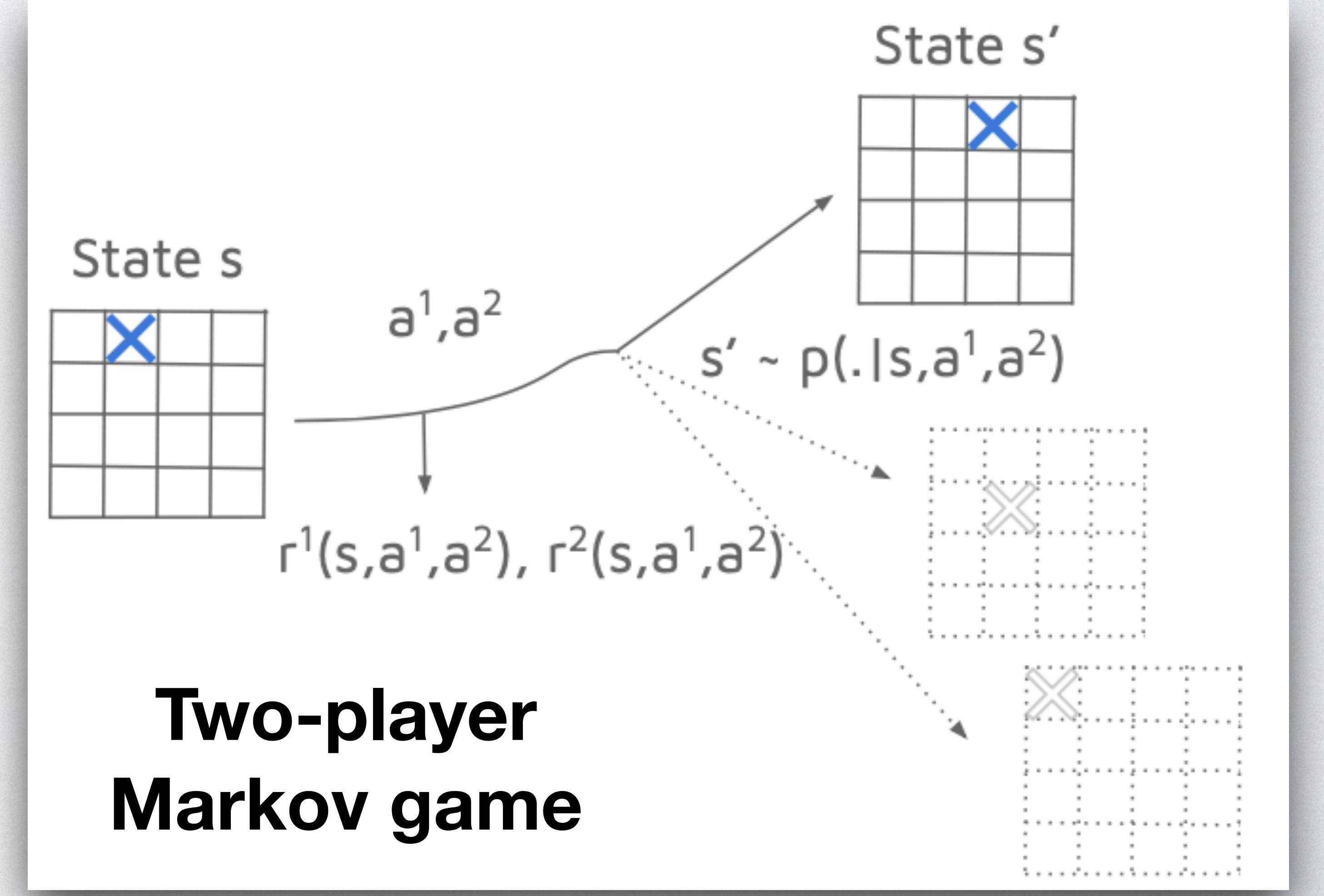


Definition 3 (Behavioral Strategy). *A behavioral strategy of an agent i is $\pi^i : \mathbb{S} \to \Delta(\mathbb{A}^i)$, i.e., $\forall s \in \mathbb{S}, \pi^i(s)$ is a probability distribution on $\mathbb{A}^i$.*



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

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



Two-player Markov game

Nash Equilibrium to MARL

- Value-based method:

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

$$Q_{i,t+1}\left(s_k,\boldsymbol{\pi}_t\right) = Q_{i,t}\left(s_t,\boldsymbol{\pi}_t\right) + \alpha\Big[R_{i,t+1} + \gamma\cdot\textbf{eval}_i\left\{Q_{\cdot,t}\left(s_{t+1},\cdot\right)\right\} - Q_{i,t}\left(s_t,\boldsymbol{\pi}_t\right)\Big]$$

  - 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

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

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

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

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

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

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.

$$R_i\left(a_i, a_{-i}\right) \geq R_i\left(a_i', a_{-i}\right) - \epsilon$$

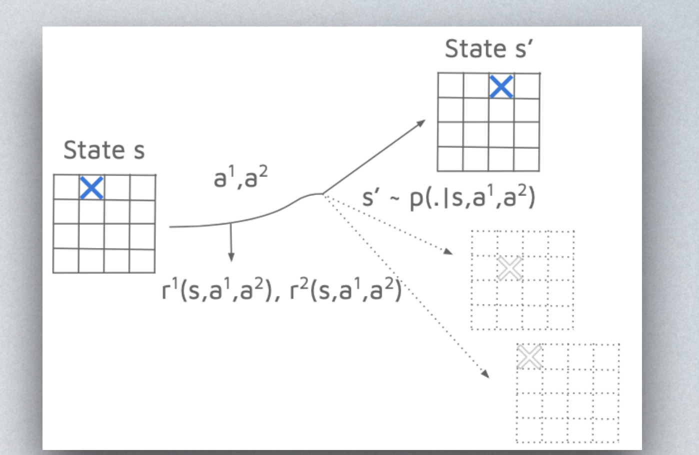
$$\epsilon = .75 \rightarrow .50 \rightarrow .38 \rightarrow .37 \rightarrow .3393$$ [Tsaknakis 2008]

recently 1/3 !

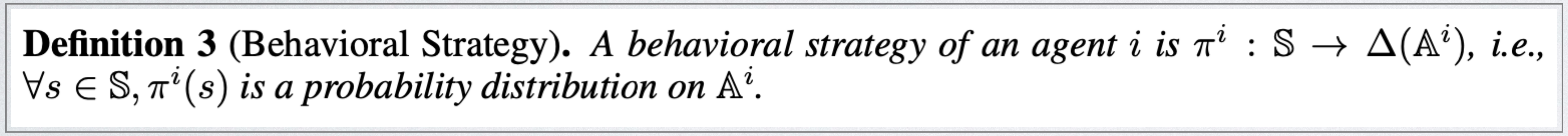
  - Equilibrium selection is problematic, how to coordinate agents to agree on Nash during training is unknown.
  - Nash equilibrium assumes perfect rationality, but can be unrealistic in the real world.

- 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.

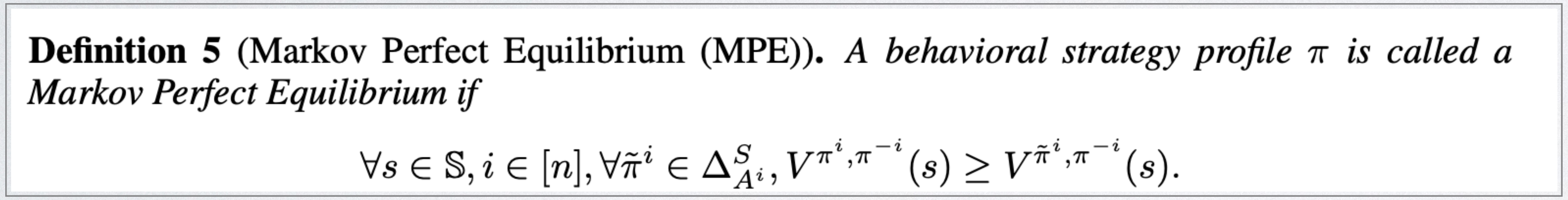
Computing Nash Equilibrium in Stochastic Games



- 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.**

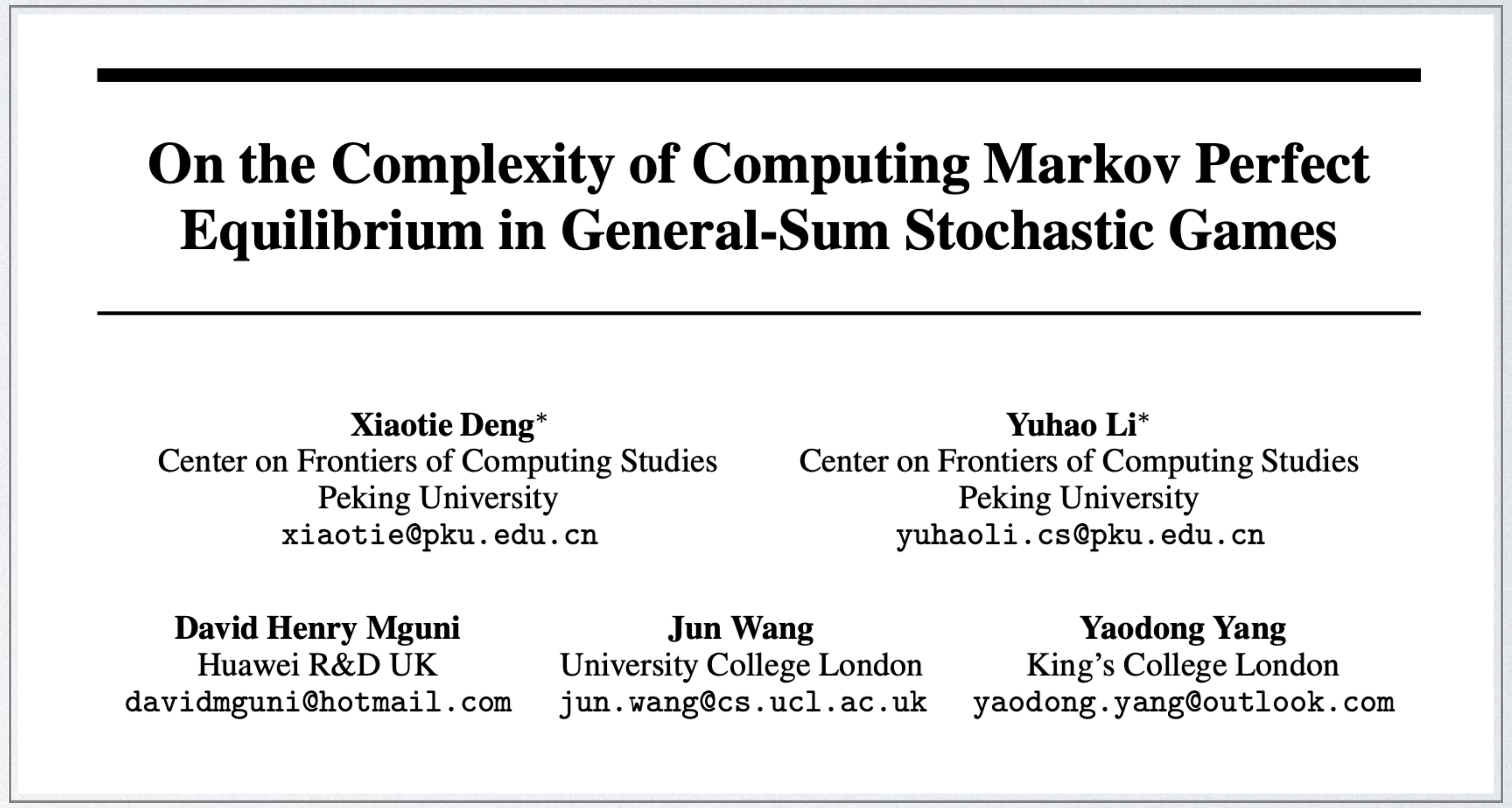


Definition 3 (Behavioral Strategy). *A behavioral strategy of an agent i is $\pi^i : \mathbb{S} \rightarrow \Delta(\mathbb{A}^i)$, i.e., $\forall s \in \mathbb{S}, \pi^i(s)$ is a probability distribution on $\mathbb{A}^i$.*



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

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



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
yuhaoli.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≠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

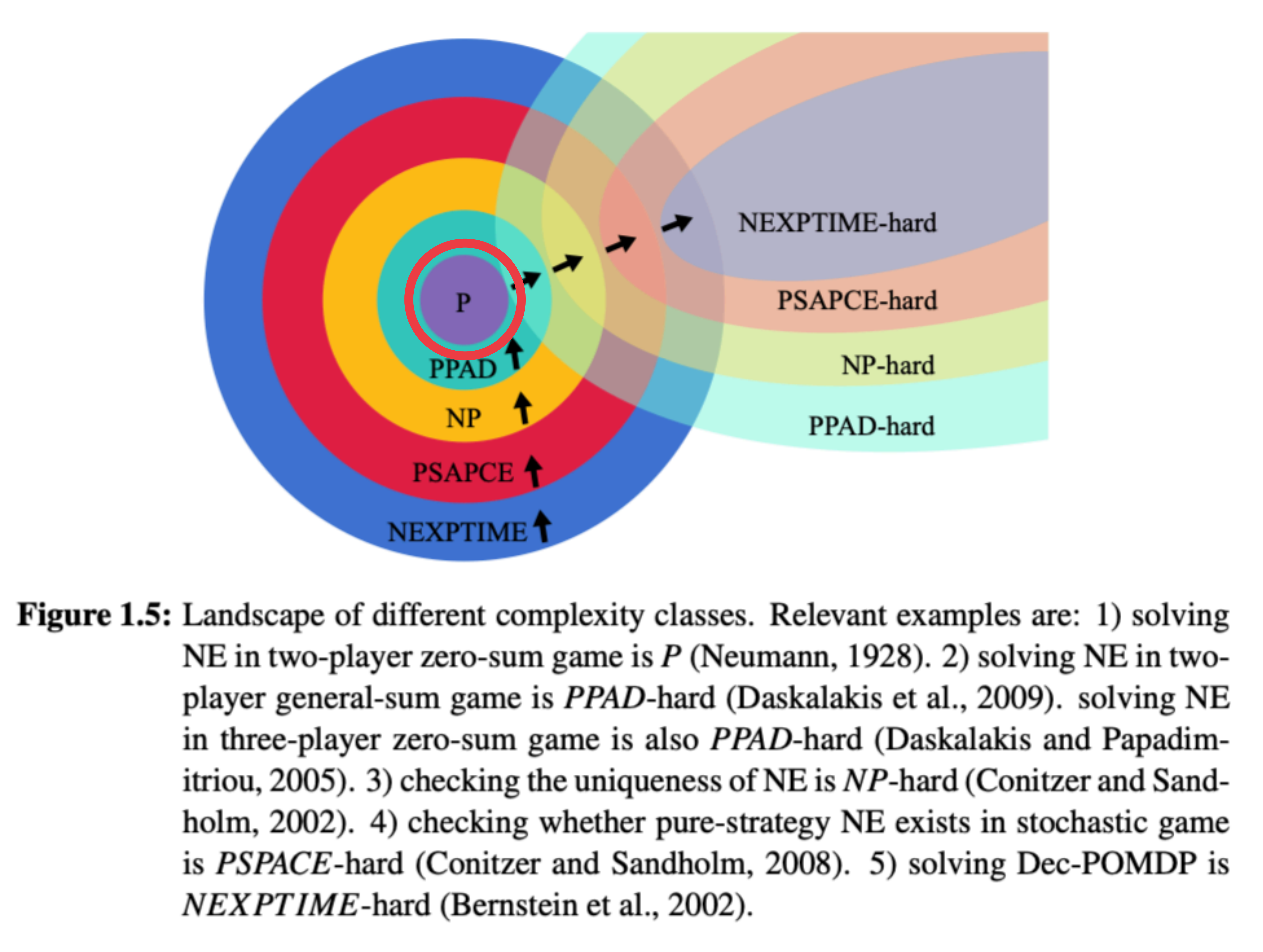
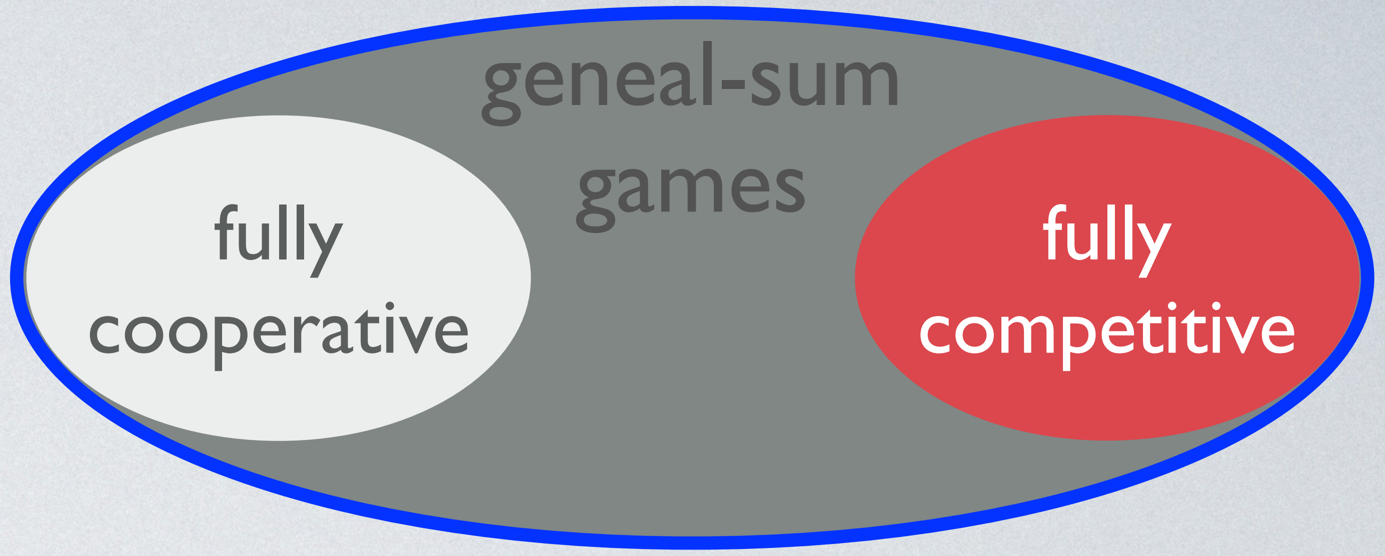


Figure 1.5: Landscape of different complexity classes. Relevant examples are: 1) solving NE in two-player zero-sum game is *P* (Neumann, 1928). 2) solving NE in two-player general-sum game is *PPAD*-hard (Daskalakis et al., 2009). solving NE in three-player zero-sum game is also *PPAD*-hard (Daskalakis 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).

https://arxiv.org/abs/2011.00583

MARL in Zero-Sum Games



Great advantages have been made in 2019!

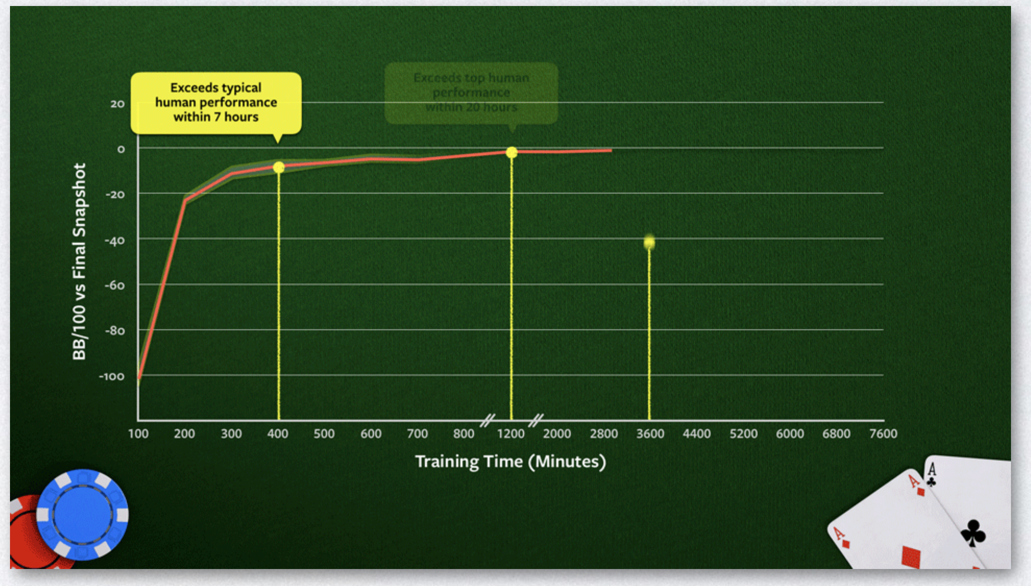
Jan 2016 | Dec 2017 | July 2018 | Jan 2019 | Apr 2019 | July 2019 | Sep 2019

AlphaGO Series

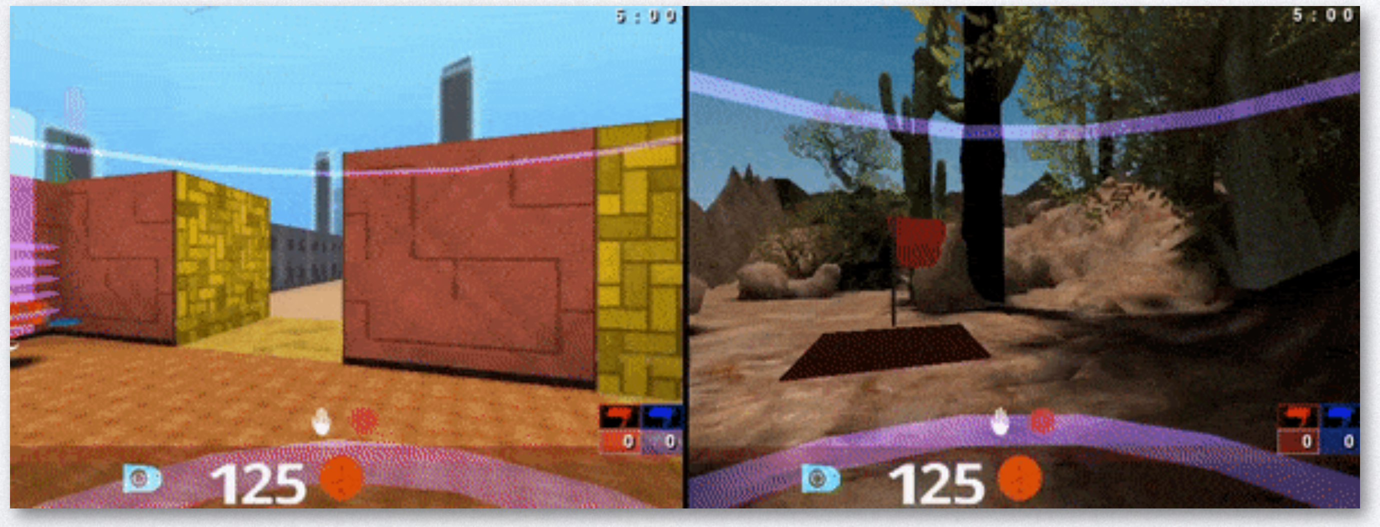
technique of single-agent decision-making is mature

AlphaStar (DeepMind)

Pluribus Poker (FAIR)

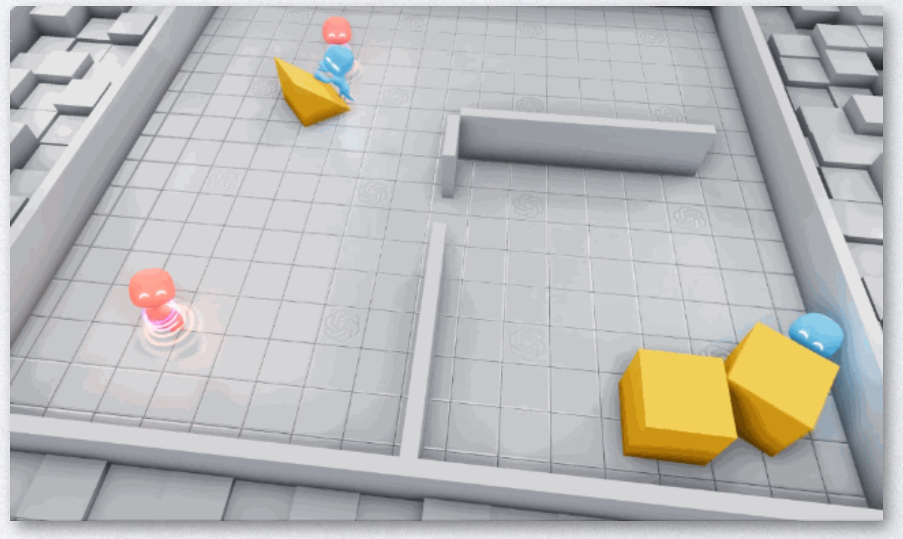


Capture-the-flag (DeepMind)



Dota2 (OpenAI)

Hide and Seek (OpenAI)



techniques of multi-agent decision-making is getting mature !

A General Solver to Two-Player Zero-Sum Games

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

Black-box multi-agent game engine

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

input

Our algorithm:

output

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

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

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

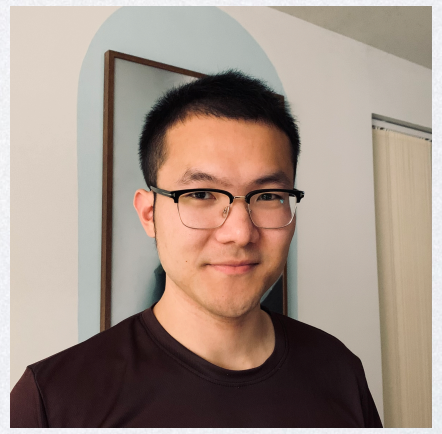
Two Mainstreams of Multi-Agent Learning algorithms

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.

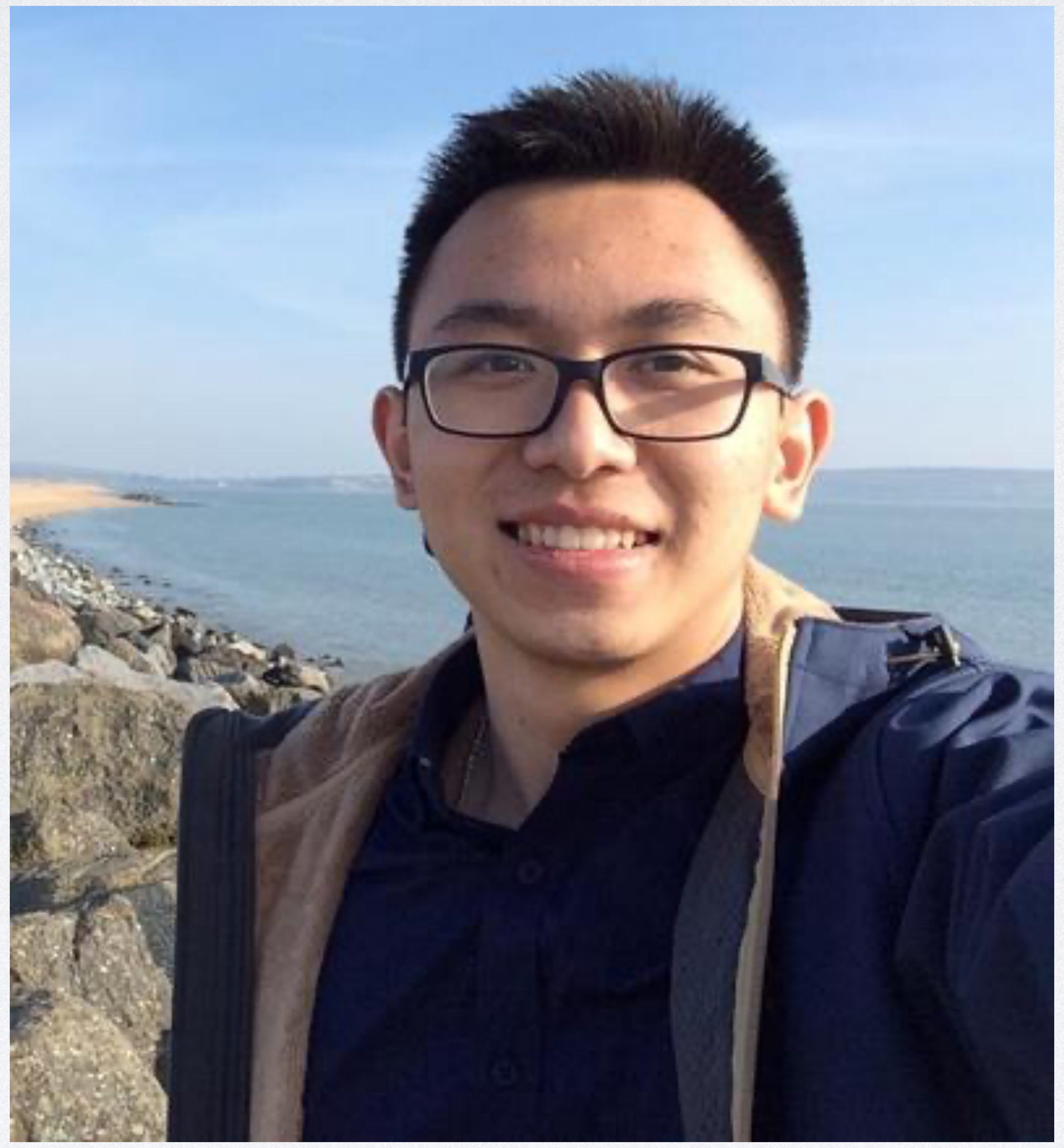
Population-based methods:

- 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 genera-sum games.
- Average policy have convergence guarantee but generally no last-iteration convergence



Le Cong Dinh

University of Southampton



Summary of Online Double Oracle Results

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

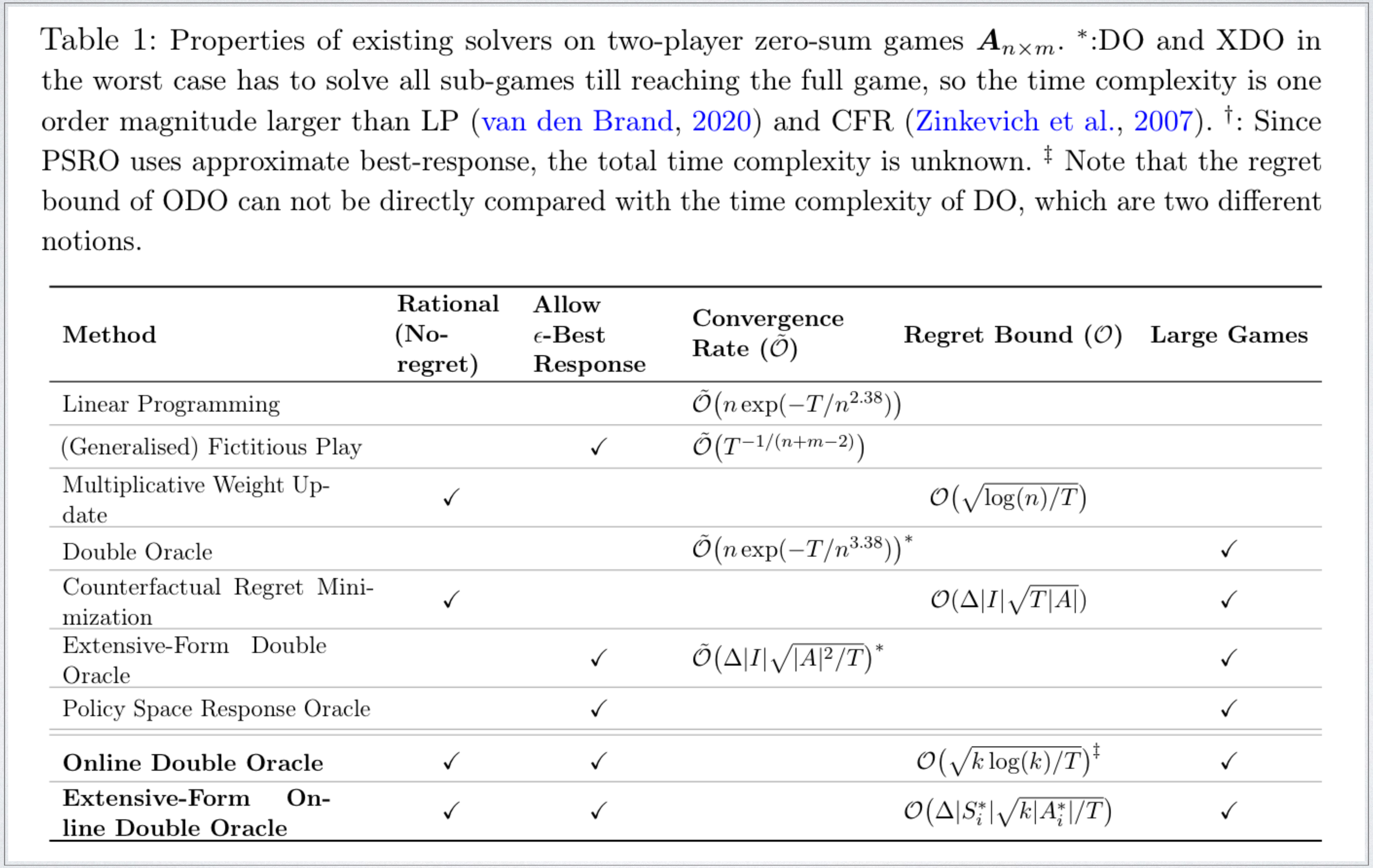


Table 1: Properties of existing solvers on two-player zero-sum games $\boldsymbol{A}_{n\times m}$. *:DO and XDO in the worst case has to solve all sub-games till reaching the full game, so the time complexity is one order magnitude larger than LP (van den Brand, 2020) and CFR (Zinkevich et al., 2007). †: 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 ϵ-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	✓			$\mathcal{O}(\Delta\lvert I\rvert\sqrt{T\lvert A\rvert})$	✓
Extensive-Form Double Oracle		✓	$\tilde{\mathcal{O}}(\Delta\lvert I\rvert\sqrt{\lvert A\rvert^2/T})^*$		✓
Policy Space Response Oracle		✓			✓
Online Double Oracle	✓	✓		$\mathcal{O}(\sqrt{k\log(k)/T})^\ddagger$	✓
Extensive-Form Online Double Oracle	✓	✓		$\mathcal{O}(\Delta\lvert S_i^*\rvert\sqrt{k\lvert A_i^*\rvert/T})$	✓

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

$$\max_{v\in\mathbb{R}} v$$

s.t. $\mathbf{p}^\top\mathbf{A} \geq v\cdot\mathbf{1}$

$\mathbf{p}\geq\mathbf{0}$ and $\mathbf{p}^\top\mathbf{1}=1$

/

Dual problem

column player's view

$$\min_{v\in\mathbb{R}} v$$

s.t. $\mathbf{q}^\top\mathbf{A}^\top \leq v\cdot\mathbf{1}$

$\mathbf{q}\geq\mathbf{0}$ and $\mathbf{q}^\top\mathbf{1}=1$

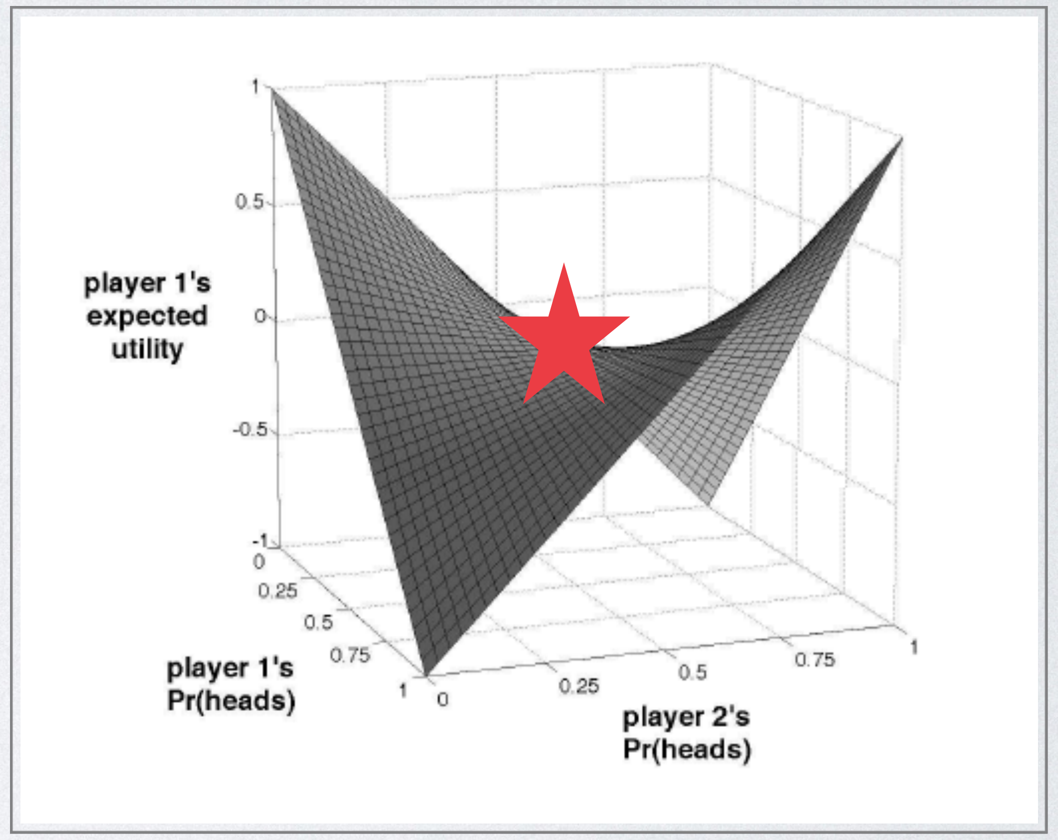
⟷

Minimax theorem

zero-duality gap for convex problems

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

- 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 (p, q) is the Nash equilibrium:
  - proof: suppose the player plays $\boldsymbol{x},\boldsymbol{y}$ instead of $\boldsymbol{p},\boldsymbol{q}$
  - $x^TAq = \sum_{i=1}^{N} x_i(Aq)_i \leq \max_{i\in[N]}(Aq)_i = v_c = v^*$, $p^TAy = \sum_{j=1}^{M}\left(p^TA\right)_j y_j \geq \min_{j\in[M]}\left(p^TA\right)_j = v_r = v^*$, thus no incentives to deviate.

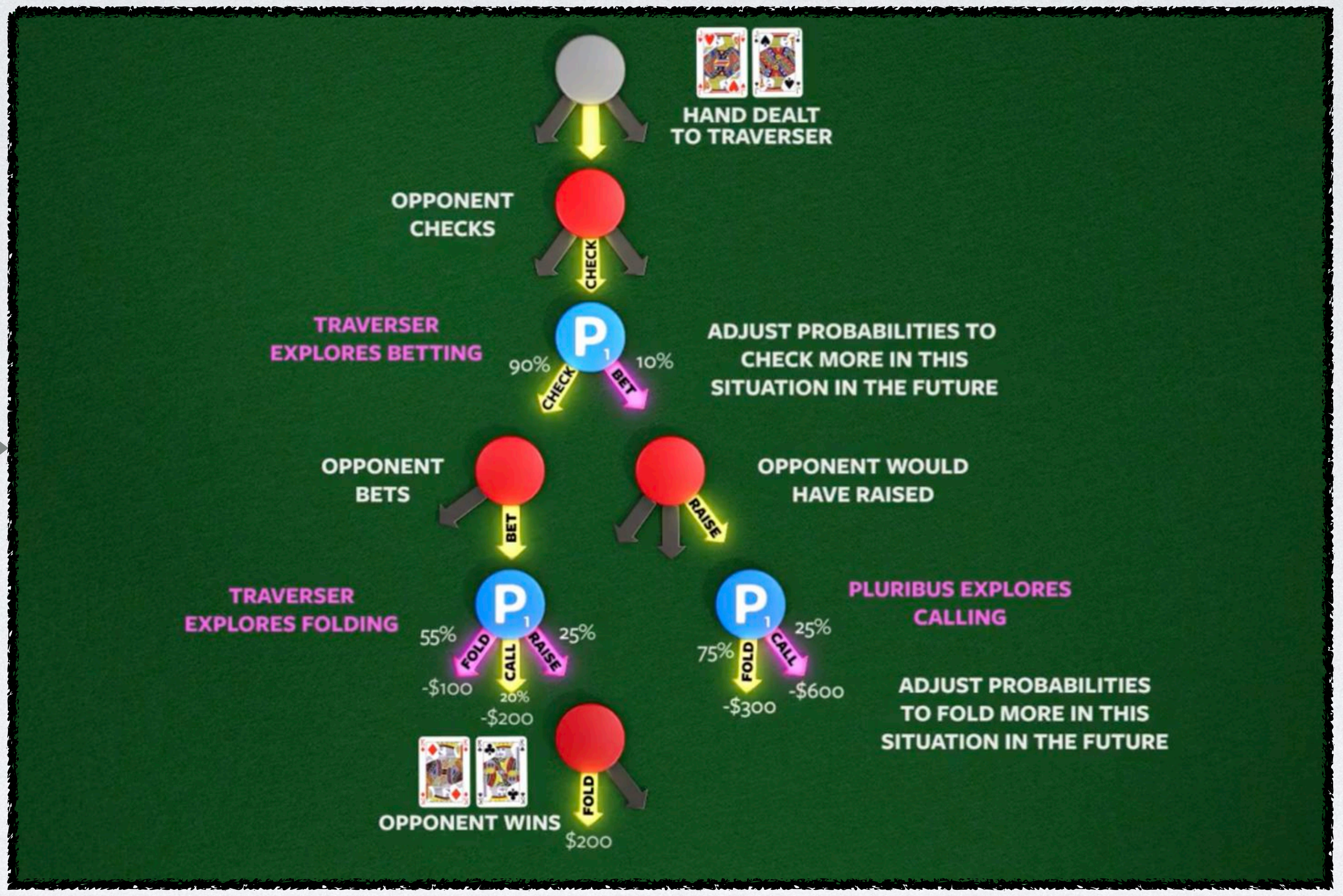
- Sion's minimax theorem generalises to quasi-convex/concave functions $\min_{x\in X}\sup_{y\in Y} f(x,y) = \sup_{y\in Y}\min_{x\in X} f(x,y)$

When and Why we need Population-based Methods ?

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

Black-box multi-agent game engine

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


HAND DEALT TO TRAVERSER
OPPONENT CHECKS
CHECK
TRAVERSER EXPLORES BETTING
90%
CHECK
BET
10%
ADJUST PROBABILITIES TO CHECK MORE IN THIS SITUATION IN THE FUTURE
OPPONENT BETS
BET
OPPONENT WOULD HAVE RAISED
RAISE
TRAVERSER EXPLORES FOLDING
55%
FOLD
CALL
20%
RAISE
25%
-$100
-$200
PLURIBUS EXPLORES CALLING
75%
FOLD
CALL
25%
-$300
-$600
ADJUST PROBABILITIES TO FOLD MORE IN THIS SITUATION IN THE FUTURE
OPPONENT WINS
FOLD
$200

Regret based methods: Poker 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

Population based methods: StarCraft type

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 (due to non-transitivity).
- 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.

meta-game analysis

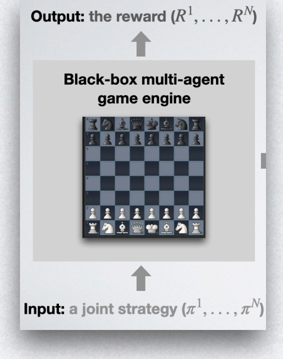
Player = Agent

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

$$\phi : \underset{\text{RL model}}{V} \times \underset{\text{RL model}}{W} \to \mathbb{R}$$


Output: the reward $(R^1, \ldots, R^N)$
Black-box multi-agent game engine
Input: a joint strategy $(\pi^1, \ldots, \pi^N)$

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

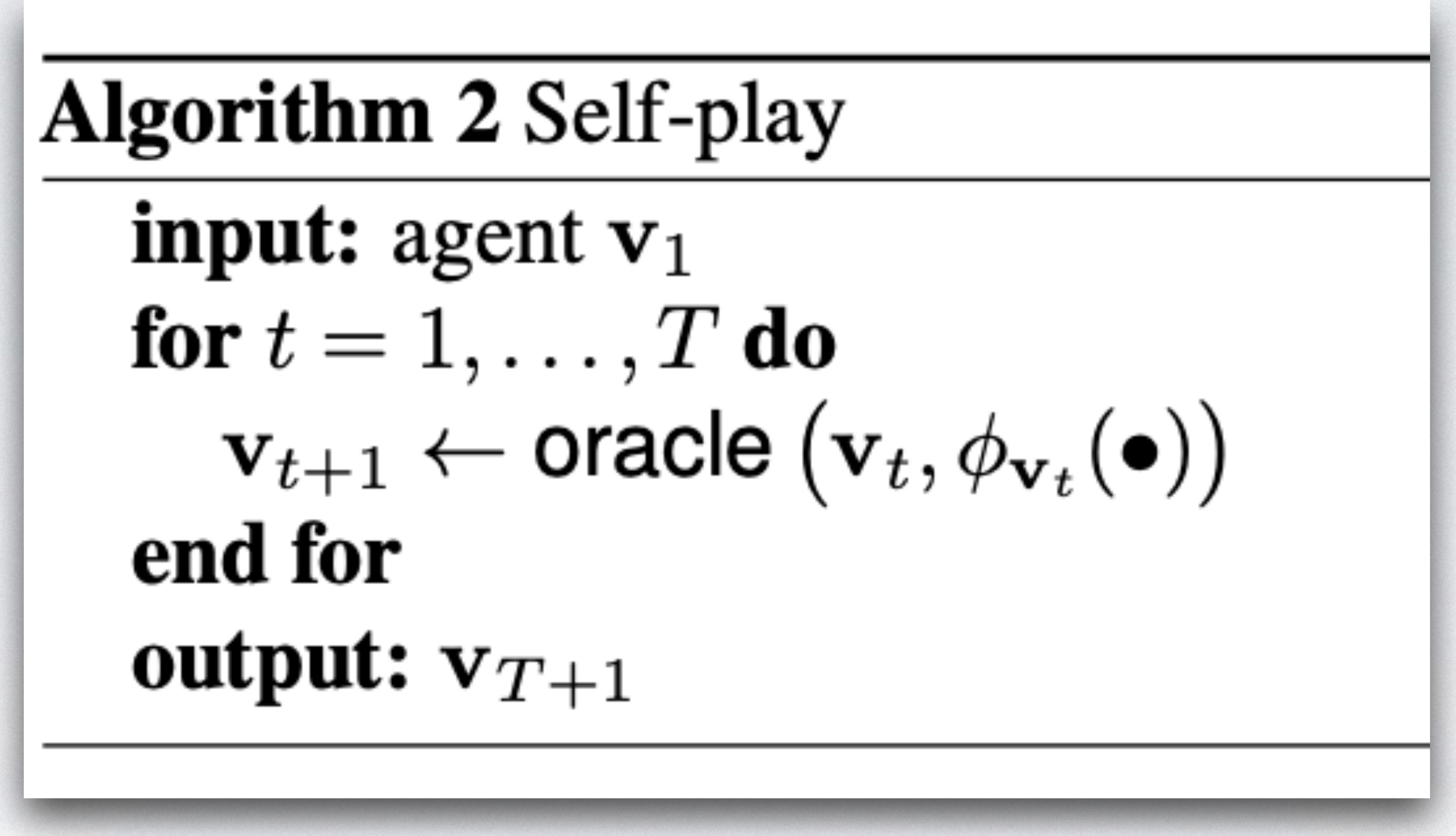
$$v' := \textbf{Br}(w) = \textbf{Oracle}\Big(v, \phi_w(\,\cdot\,)\Big) \quad \textbf{s.t.} \quad \phi_{\mathbf{w}}(\mathbf{v}') > \phi_{\mathbf{w}}(\mathbf{v}) + \epsilon$$

  - **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 $\mathbf{v}_1$
  for $t = 1, \ldots, T$ do
    $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \phi_{\mathbf{v}_t}(\bullet)\right)$
  end for
  output: $\mathbf{v}_{T+1}$
```

self-plays

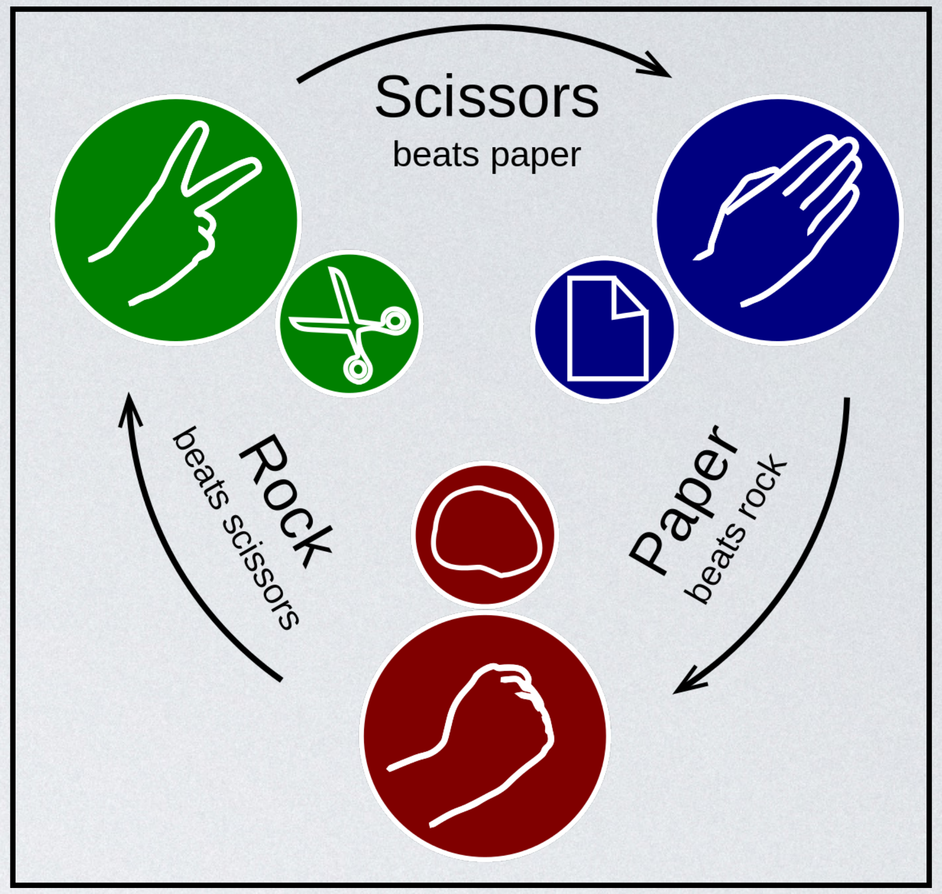
$$\begin{tikzcd}
\left(\pi^1, \pi^2\right) \arrow[r] & \left(\pi^1, \pi^{2,*} = \mathbf{Br}(\pi^1)\right) \arrow[r] & \left(\pi^{1,*} = \mathbf{Br}(\pi^{2,*}), \pi^{2,*}\right) \arrow[ll, bend left]
\end{tikzcd}$$

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

Naive Self-play Will Not Work

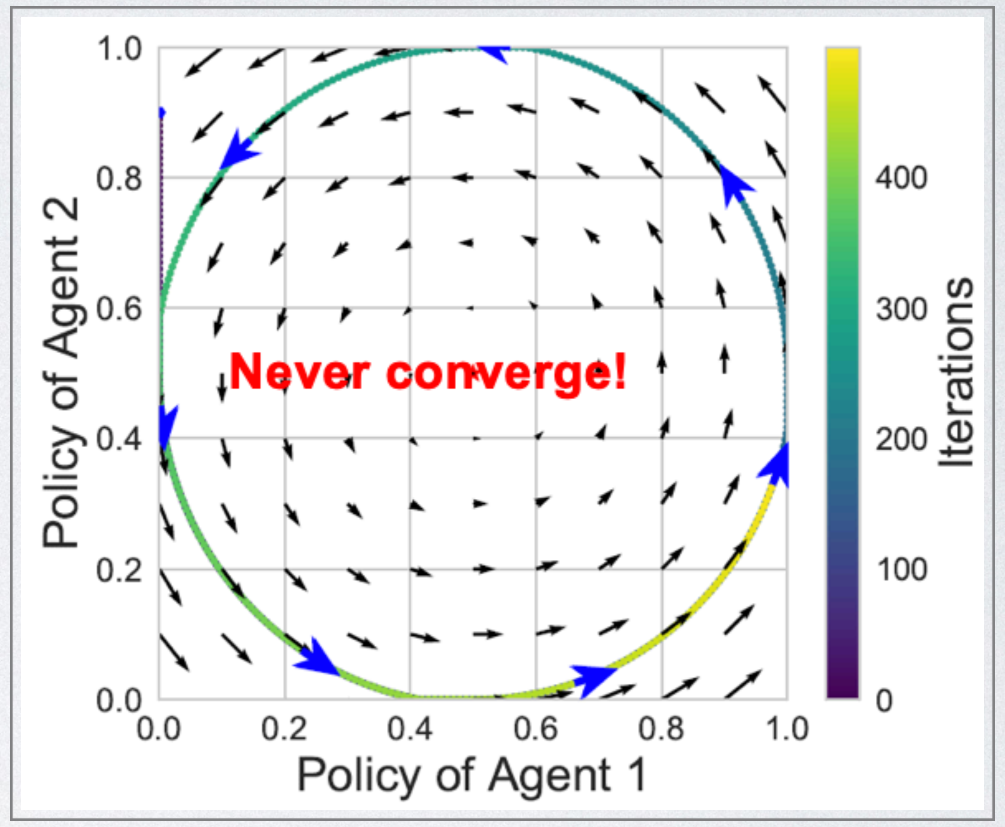
- It is because of Non-Transitivity

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



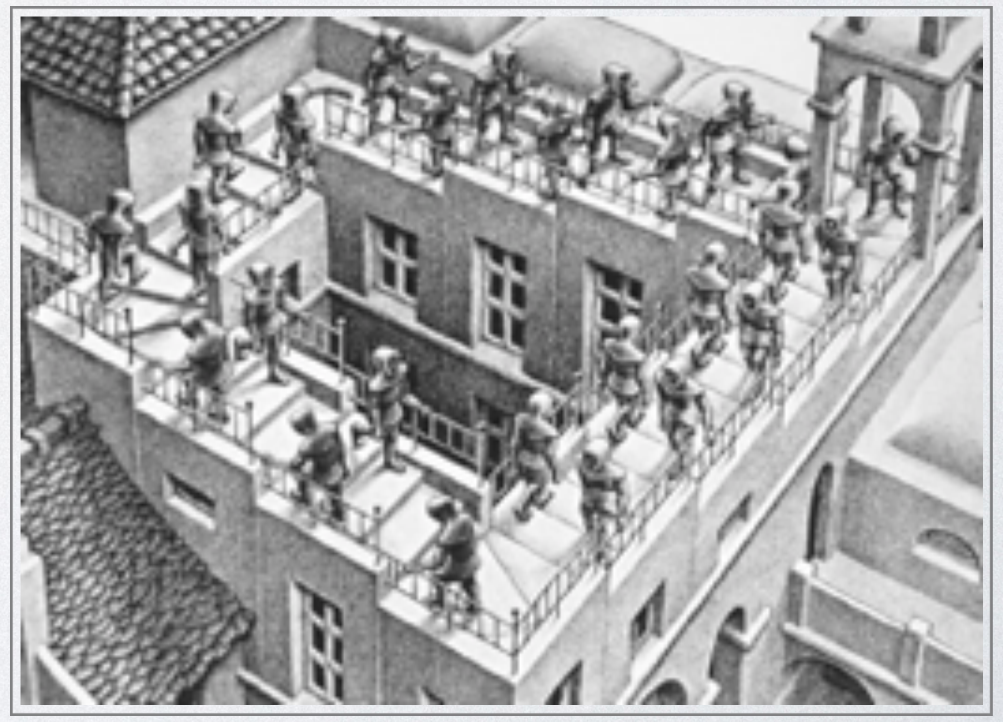
- Rock-Paper-Scissor game:

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



- 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]

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

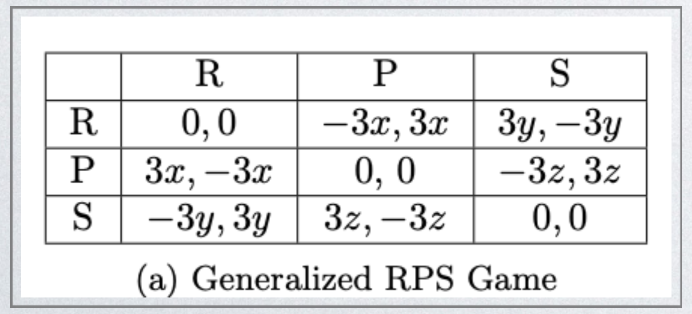
- Let $v, w \in W$ be a compact set and $\phi(v, w)$ prescribe the flow from v to w, then this is a natural result after applying *combinatorial hodge theory* [Jiang 2011].
- We can write any games ϕ as summation of two **orthogonal** components

$$\text{grad}(f)(\mathbf{v}, \mathbf{w}) := f(\mathbf{v}) - f(\mathbf{w}) \qquad \text{div}(\phi)(\mathbf{v}) := \int_W \phi(\mathbf{v}, \mathbf{w}) \cdot d\mathbf{w} \qquad \text{curl}(\phi)(\mathbf{u}, \mathbf{v}, \mathbf{w}) := \phi(\mathbf{u}, \mathbf{v}) + \phi(\mathbf{v}, \mathbf{w}) - \phi(\mathbf{u}, \mathbf{w})$$

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

Transitive game **Non-transitive game**

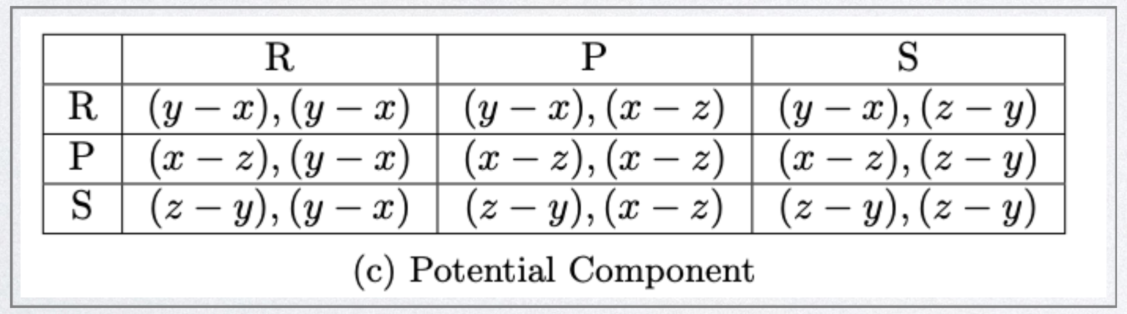
- Example on Rock-Paper-Scissor



	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

=

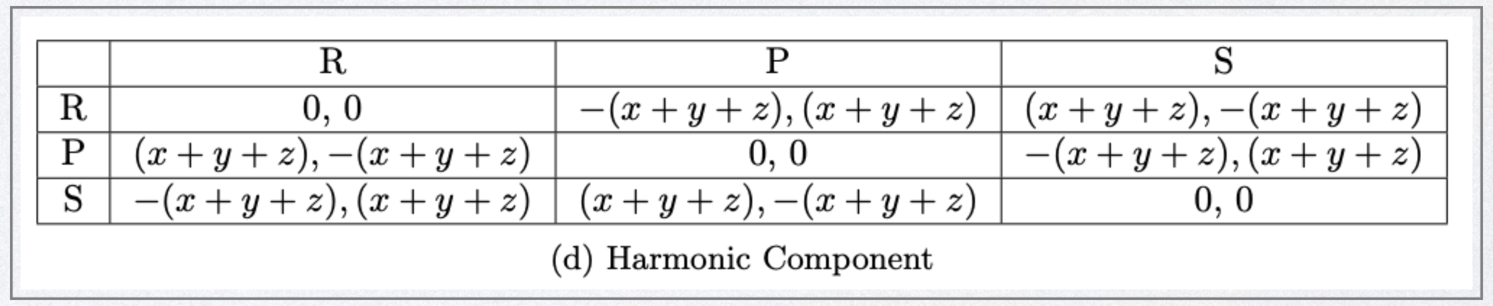


	R	P	S
R	$(y-x), (y-x)$	$(y-x), (x-z)$	$(y-x), (z-y)$
P	$(x-z), (y-x)$	$(x-z), (x-z)$	$(x-z), (z-y)$
S	$(z-y), (y-x)$	$(z-y), (x-z)$	$(z-y), (z-y)$

(c) Potential Component

Transitive game

+

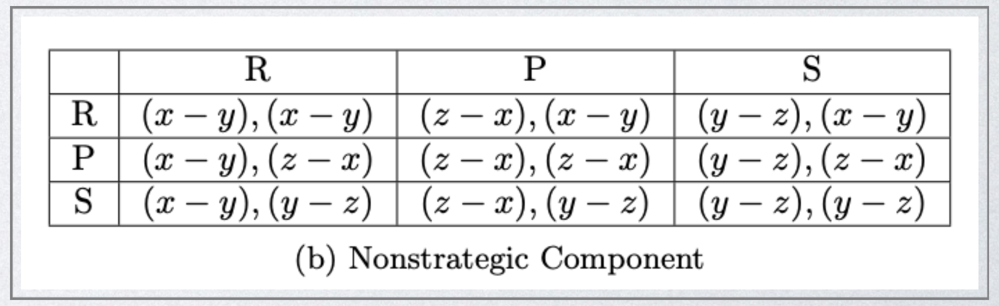


	R	P	S
R	$0, 0$	$-(x+y+z), (x+y+z)$	$(x+y+z), -(x+y+z)$
P	$(x+y+z), -(x+y+z)$	$0, 0$	$-(x+y+z), (x+y+z)$
S	$-(x+y+z), (x+y+z)$	$(x+y+z), -(x+y+z)$	$0, 0$

(d) Harmonic Component

Non-transitive game

+



	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

What is Transitivity ?

- Every FFG can be decomposed into two parts

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

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

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

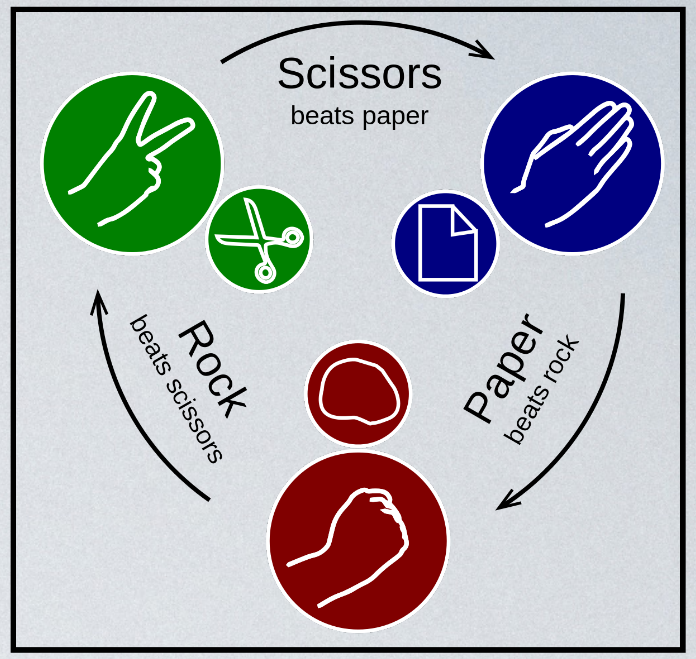
- Example: Elo rating (段位) offers rating scores $f(\cdot)$ that assume transitivity.

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

- 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 $f(\boldsymbol{v}_t) \gg f(\boldsymbol{w})$,

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

What is Non-Transitivity ?



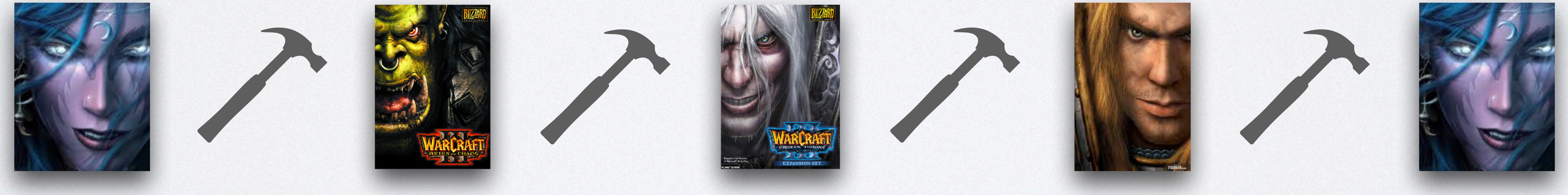
- Every FFG can be decomposed into two parts

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

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

$$\boldsymbol{v}_t \text{ beats } \boldsymbol{v}_{t-1}, \quad \boldsymbol{v}_{t+1} \text{ beats } \boldsymbol{v}_t \nrightarrow \boldsymbol{v}_{t+1} \text{ beats } \boldsymbol{v}_{t-1}$$

- 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 could 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

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

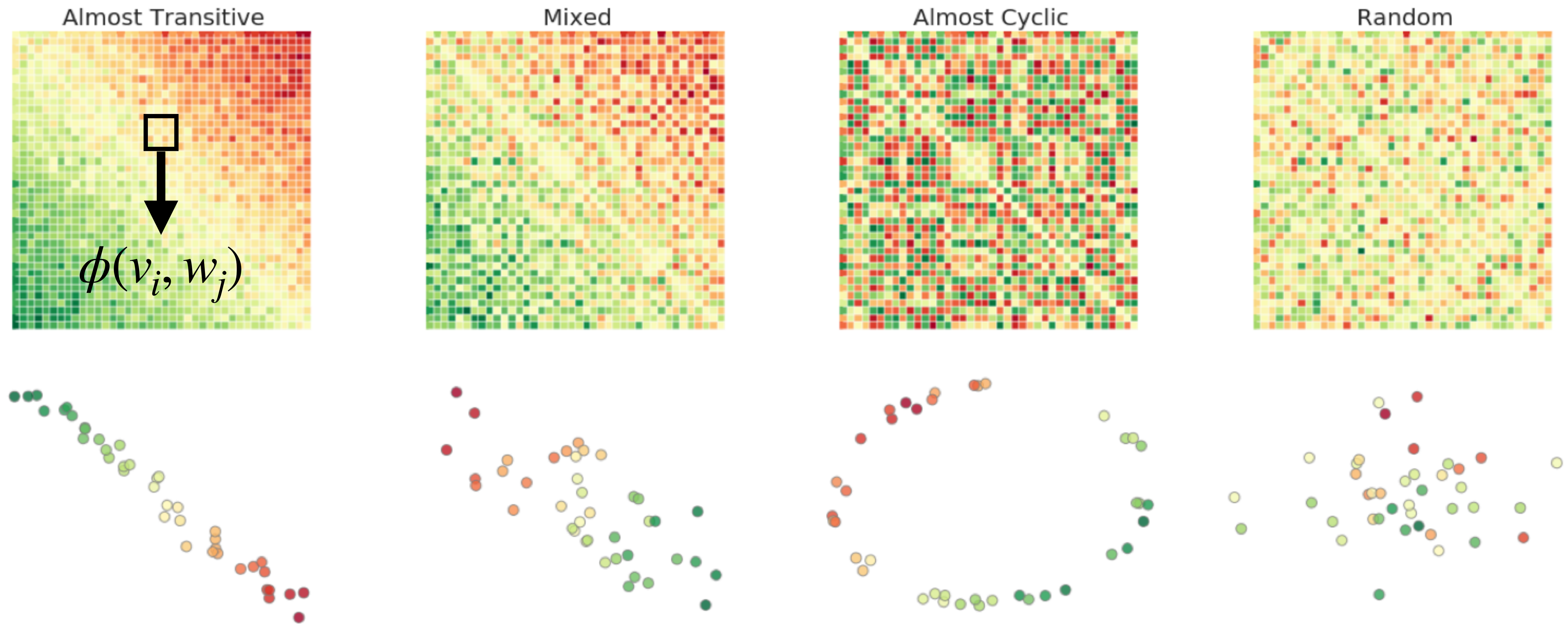


Figure 1. Low-dim gamescapes of various basic game structures. **Top row:** Evaluation matrices of populations of 40 agents each; colors vary from red to green as ϕ ranges over $[-1, 1]$. **Bottom row:** 2-dim embedding obtained by using first 2 dimensions of Schur decomposition of the payoff matrix; Color corresponds to average payoff of an agent against entire population; EGS of the transitive game is a line; EGS of the cyclic game is two-dim near-circular polytope given by convex hull of points. For extended version see Figure 6 in the Appendix.

[Balduzzi 2019]

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).

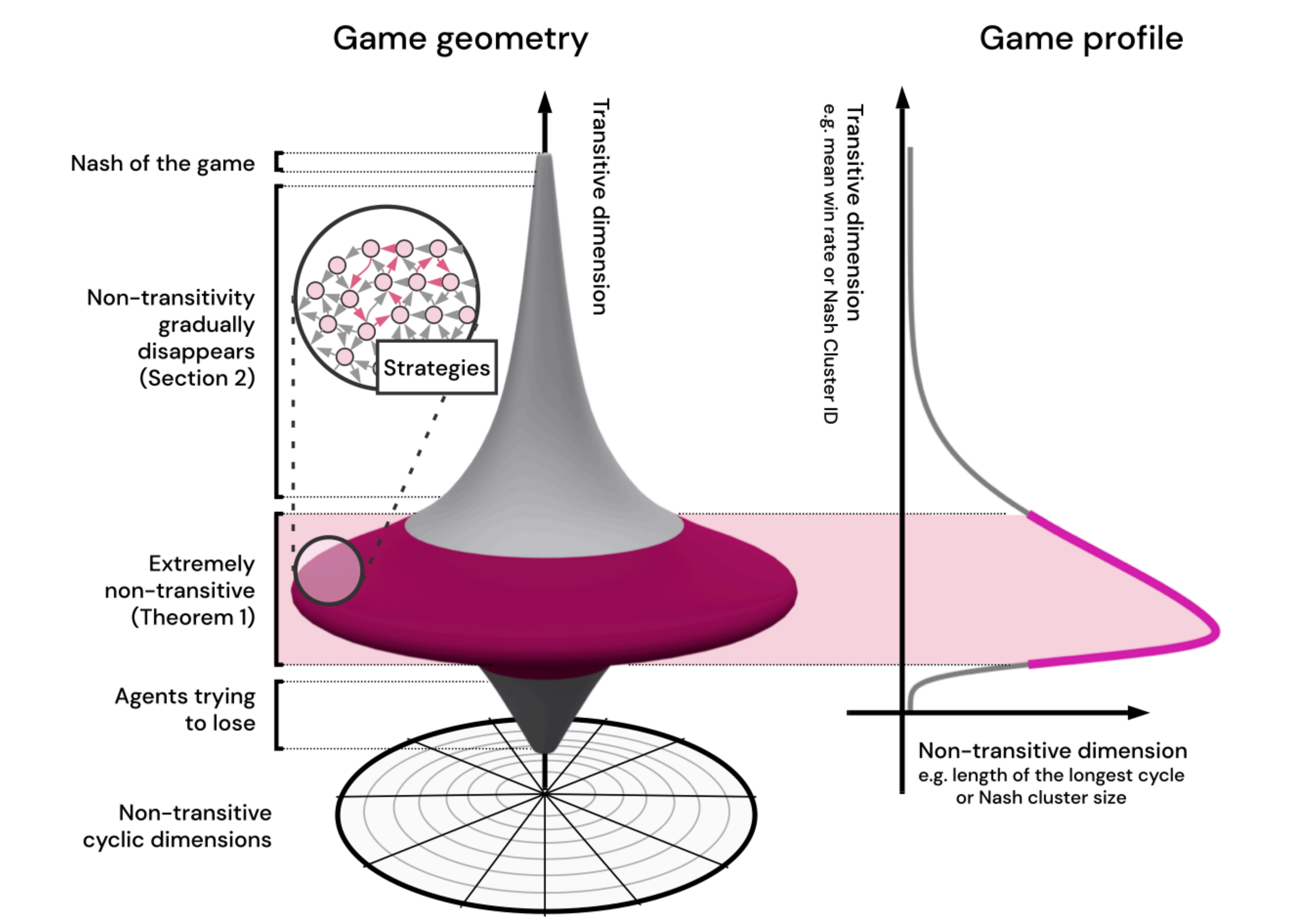
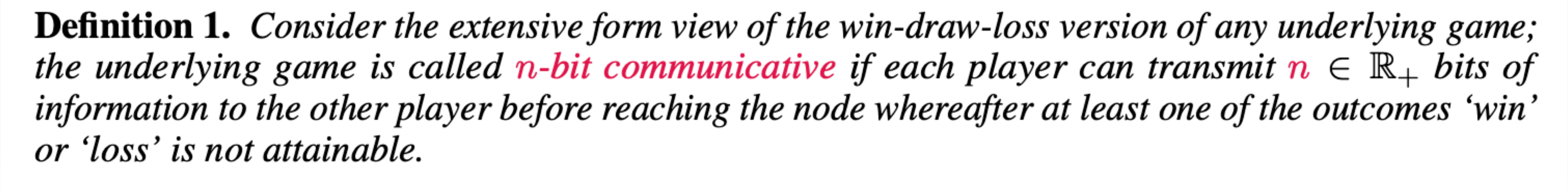

Game geometry
Game profile
Nash of the game
Non-transitivity gradually disappears (Section 2)
Strategies
Transitive dimension
Transitive dimension
e.g. mean win rate or Nash Cluster ID
Extremely non-transitive (Theorem 1)
Agents trying to lose
Non-transitive cyclic dimensions
Non-transitive dimension
e.g. length of the longest cycle or Nash cluster size

Figure 1: High-level visualisation of the geometry of Games of Skill. It shows a strong transitive dimension, that is accompanied by the highly cyclic dimensions, which gradually diminishes as skill grows towards the Nash Equilibrium (upward), and diminishes as skill evolves towards the worst possible strategies (downward). The simplest example of non-transitive behaviour is a cycle of length 3 that one finds e.g. in the Rock Paper Scissors game.

[Czarnecki 2020]

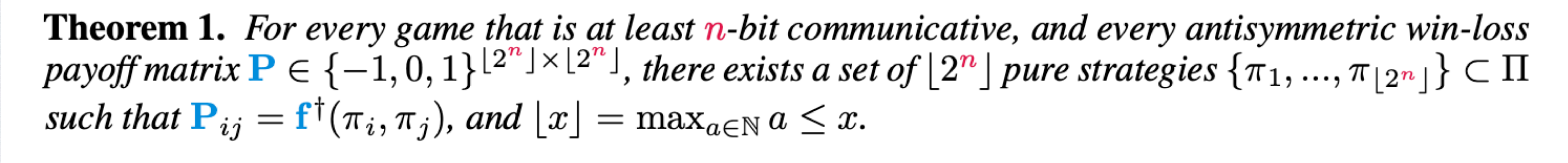
Measuring the Non-Transitivity

- 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.*

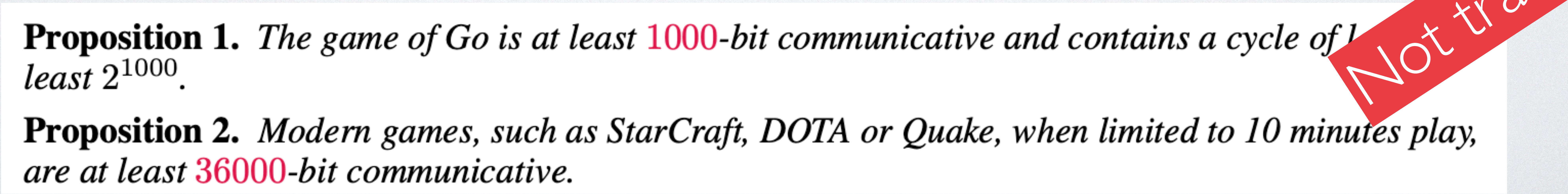
bit: how many action one can take before the outcome of the game is predetermined



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 $\{\pi_1, ..., \pi_{\lfloor 2^n \rfloor}\} \subset \Pi$ 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:



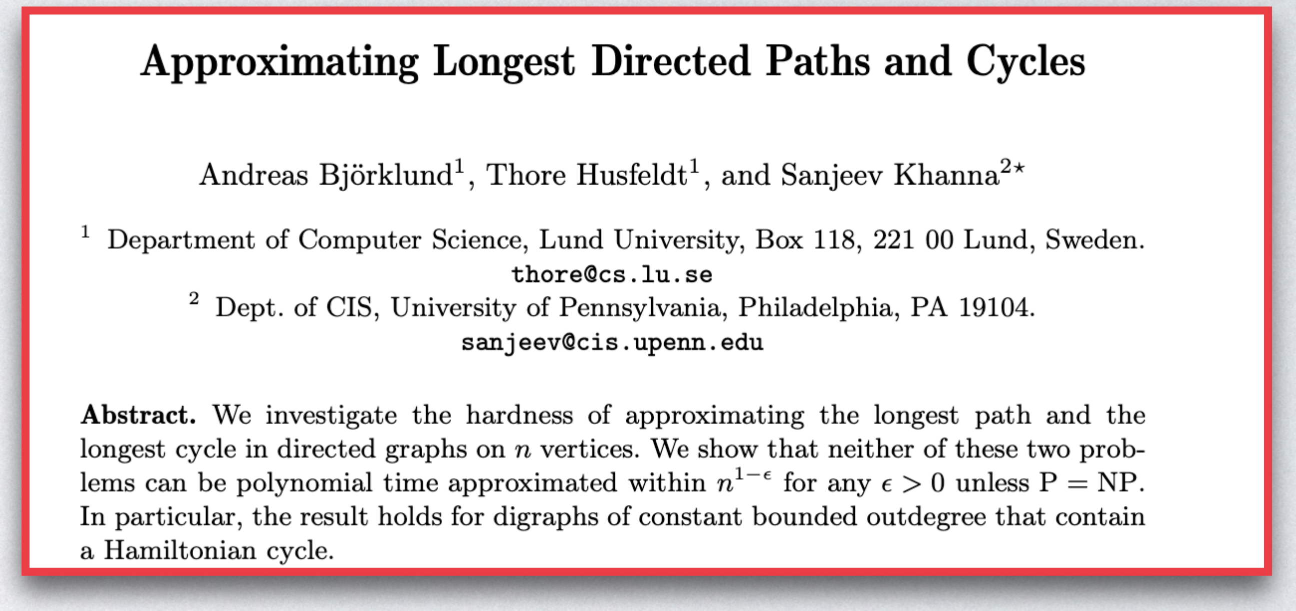
Proposition 1. *The game of Go is at least 1000-bit communicative and contains a cycle of l least 2^{1000}.*

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

Not tractable!

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[1], Thore Husfeldt[1], and Sanjeev Khanna[2*]

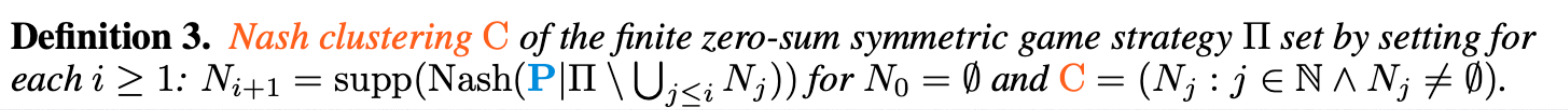
[1] Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden.
thore@cs.lu.se
[2] Dept. of CIS, University of Pennsylvania, Philadelphia, PA 19104.
sanjeev@cis.upenn.edu

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

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

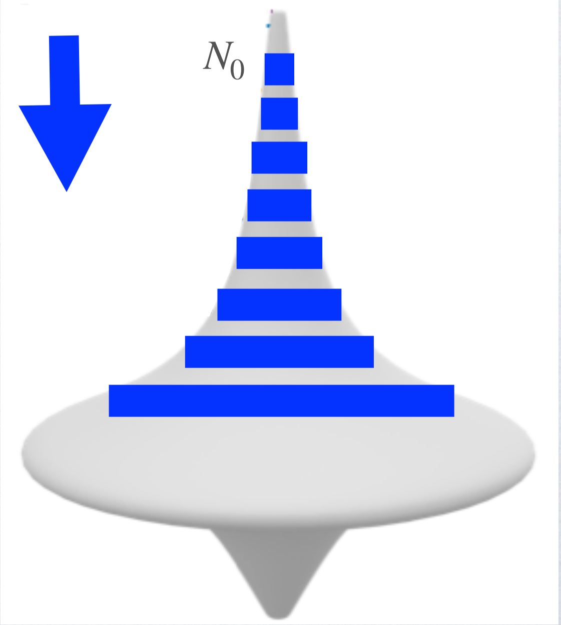
  - Method II, at each transitivity level, we can measure the *Nash Clustering*



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

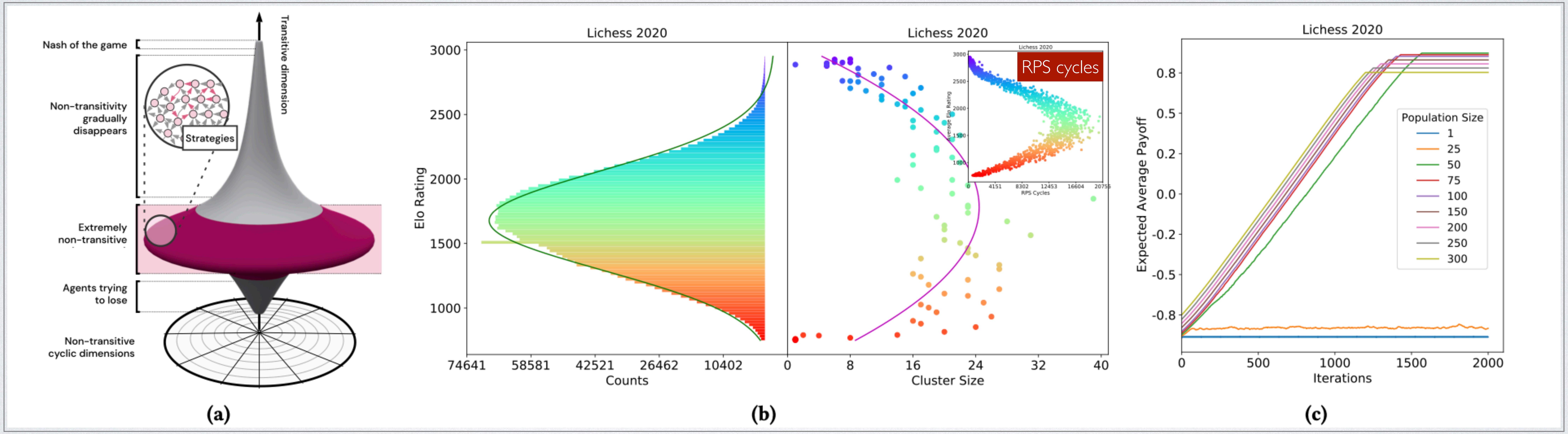
$$N_{i+1} = \mathrm{supp}\big(\mathrm{Nash}(\mathbf{P} \mid \Pi \backslash \cup_{j \leq i} N_j)\big)$$

strategies that at the
higher level of transitivity



Measuring the Non-Transitivity in Chess

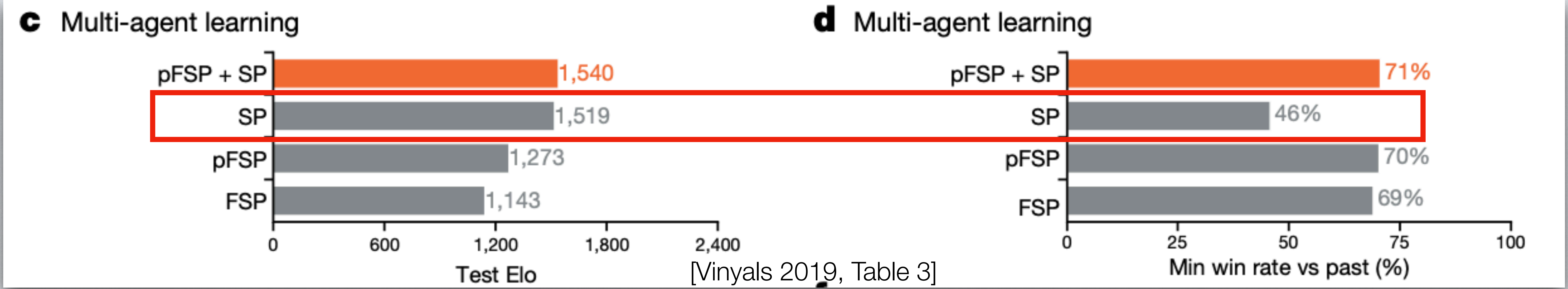
- Real-world data set from human players on Chess
  - We study one billion human player records from Lichess platform
  - Human Chess players presents the spinning-top pattern, which verifies the hypothesis



https://arxiv.org/pdf/2110.11737.pdf

Non-Transitivity Harms Training !

Example on training AlphaStar:



[Vinyals 2019, Table 3]

Example on training Soccer AI:

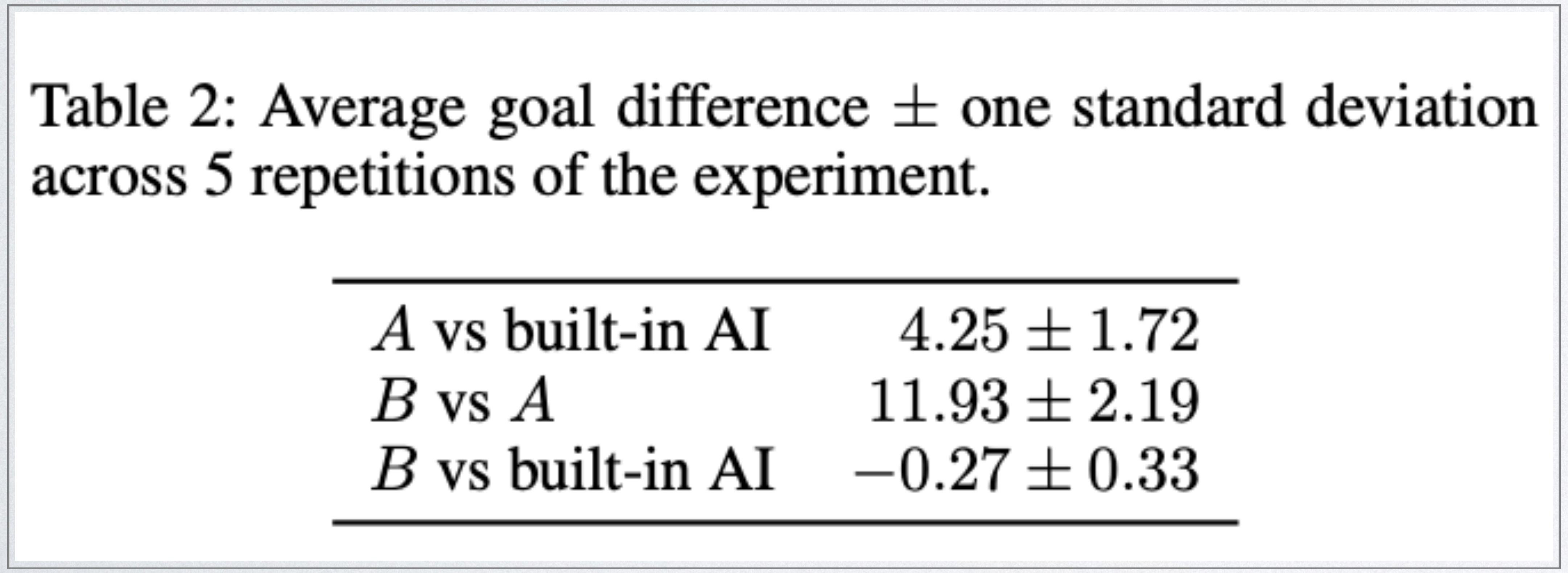


Table 2: Average goal difference ± one standard deviation across 5 repetitions of the experiment.

A vs built-in AI	4.25 ± 1.72
B vs A	11.93 ± 2.19
B vs built-in AI	-0.27 ± 0.33

[Karol 2020, table 2]

Example on training AlphaGO:

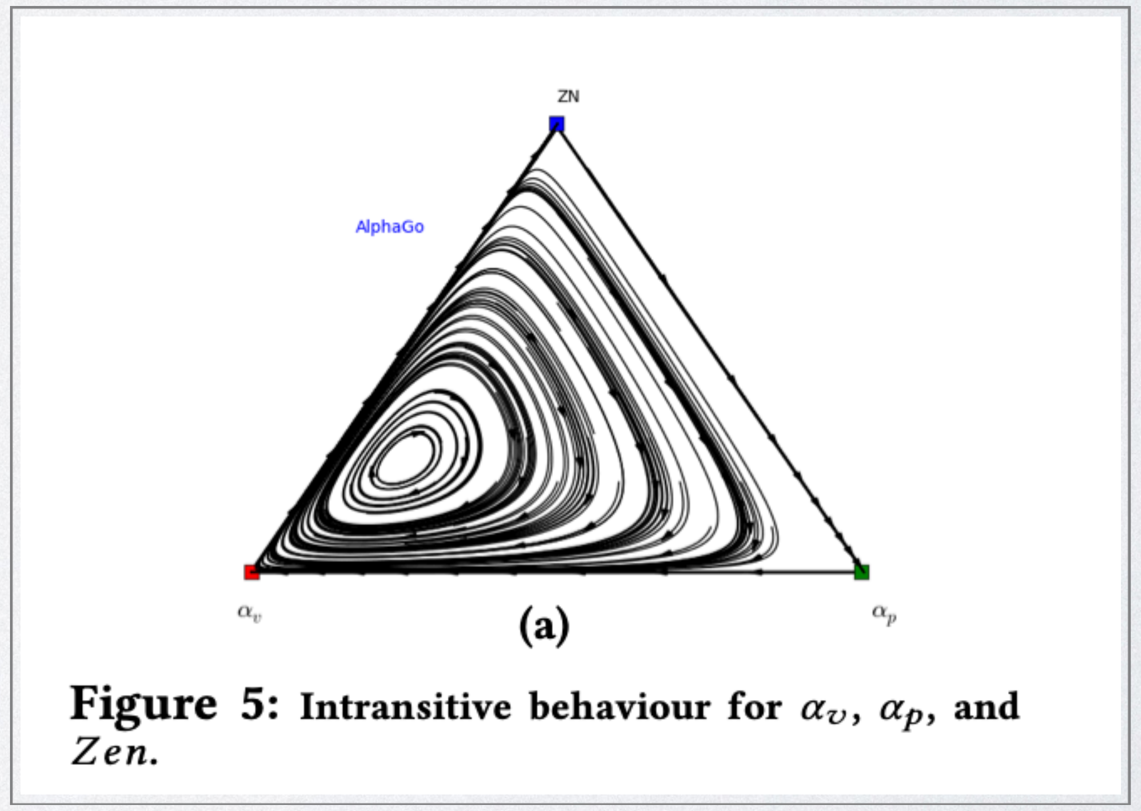
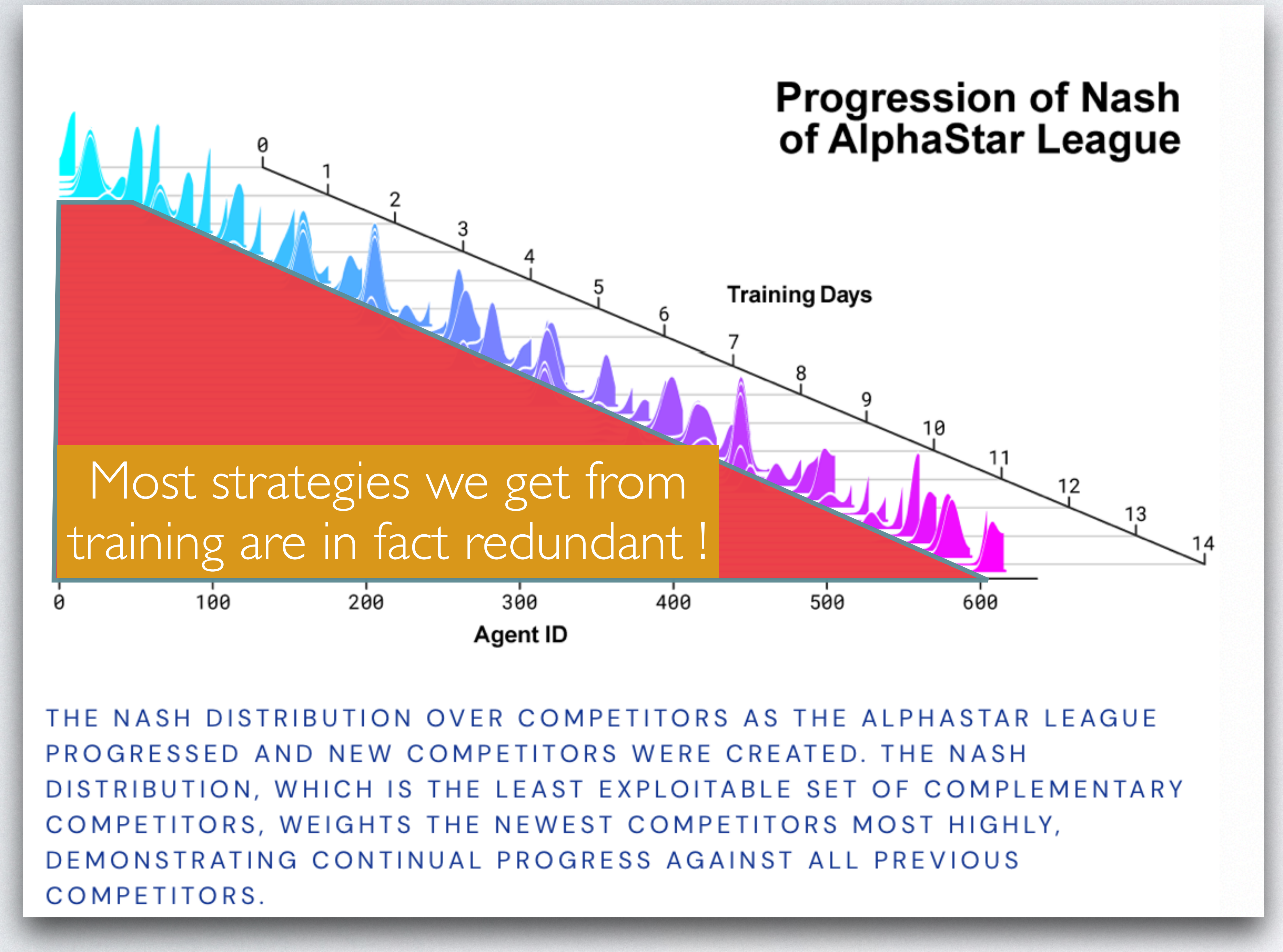


Figure 5: Intransitive behaviour for α_v, α_p, and *Zen*.

[Silver 2016, table 9]

Dealing With Non-Transitivity Helps Save Training Time



Most strategies we get from training are in fact redundant !

THE NASH DISTRIBUTION OVER COMPETITORS AS THE ALPHASTAR LEAGUE PROGRESSED AND NEW COMPETITORS WERE CREATED. THE NASH DISTRIBUTION, WHICH IS THE LEAST EXPLOITABLE SET OF COMPLEMENTARY COMPETITORS, WEIGHTS THE NEWEST COMPETITORS MOST HIGHLY, DEMONSTRATING CONTINUAL PROGRESS AGAINST ALL PREVIOUS COMPETITORS.

[AlphaStar Blog]

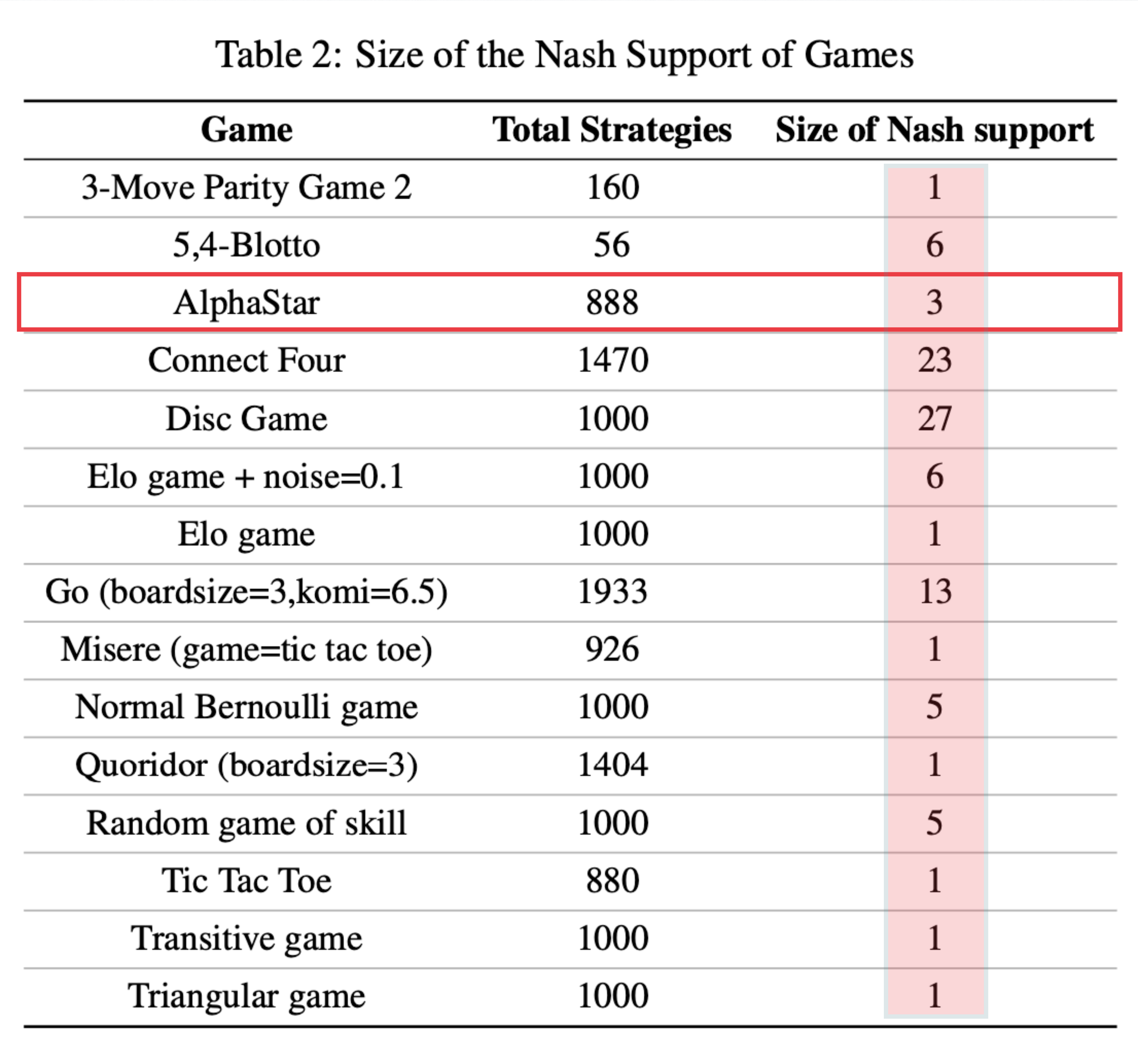


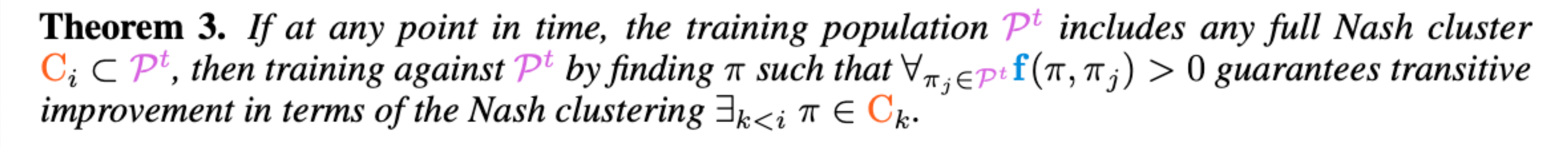
Table 2: Size of the Nash Support of Games

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]

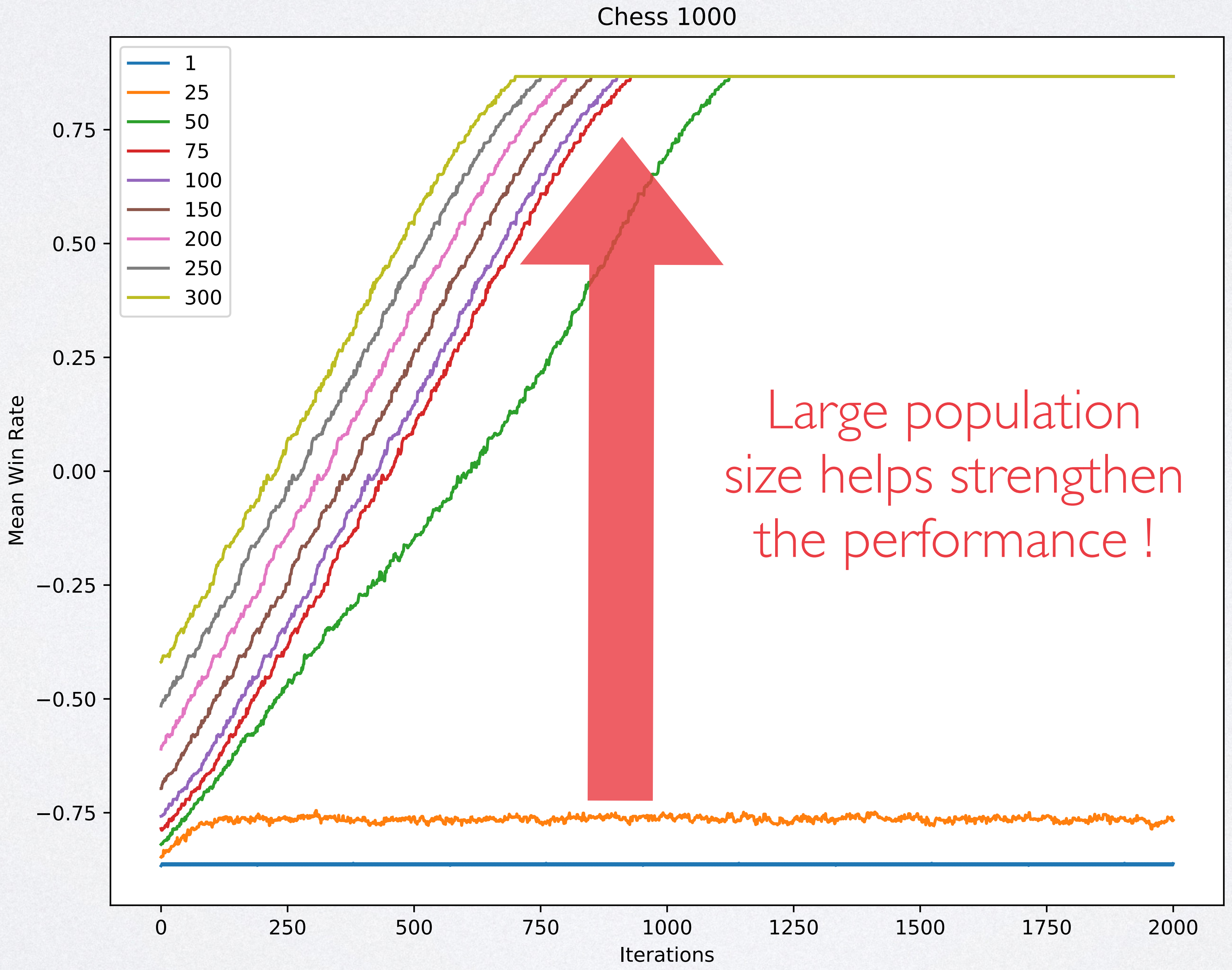
Understanding Non-Transitivity Helps Develop Algorithms !

- 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.



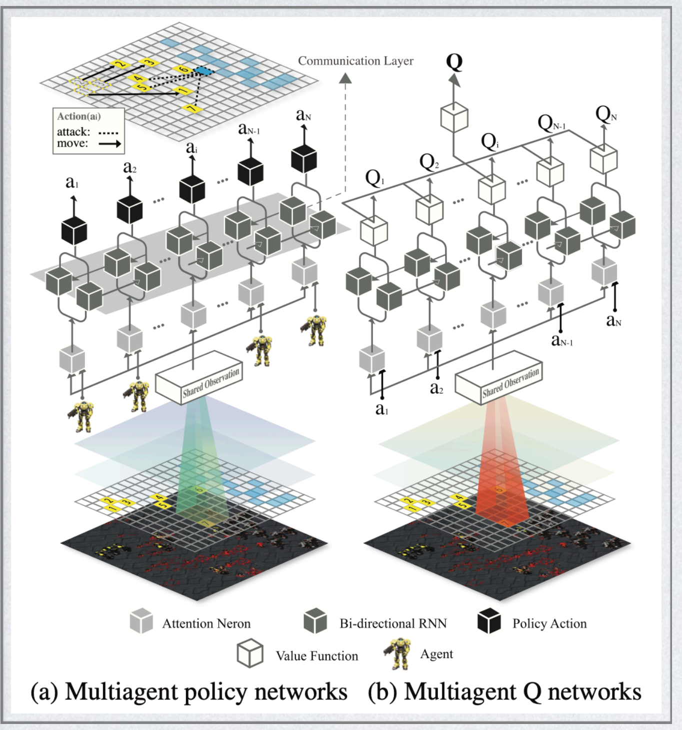
Theorem 3. *If at any point in time, the training population $\mathcal{P}^t$ includes any full Nash cluster $\mathbf{C}_i \subset \mathcal{P}^t$, then training against $\mathcal{P}^t$ by finding π such that $\forall_{\pi_j \in \mathcal{P}^t} \mathbf{f}(\pi, \pi_j) > 0$ guarantees transitive improvement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

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



Large population size helps strengthen the performance !

Understanding Non-Transitivity Helps Develop Efficient Algorithms !



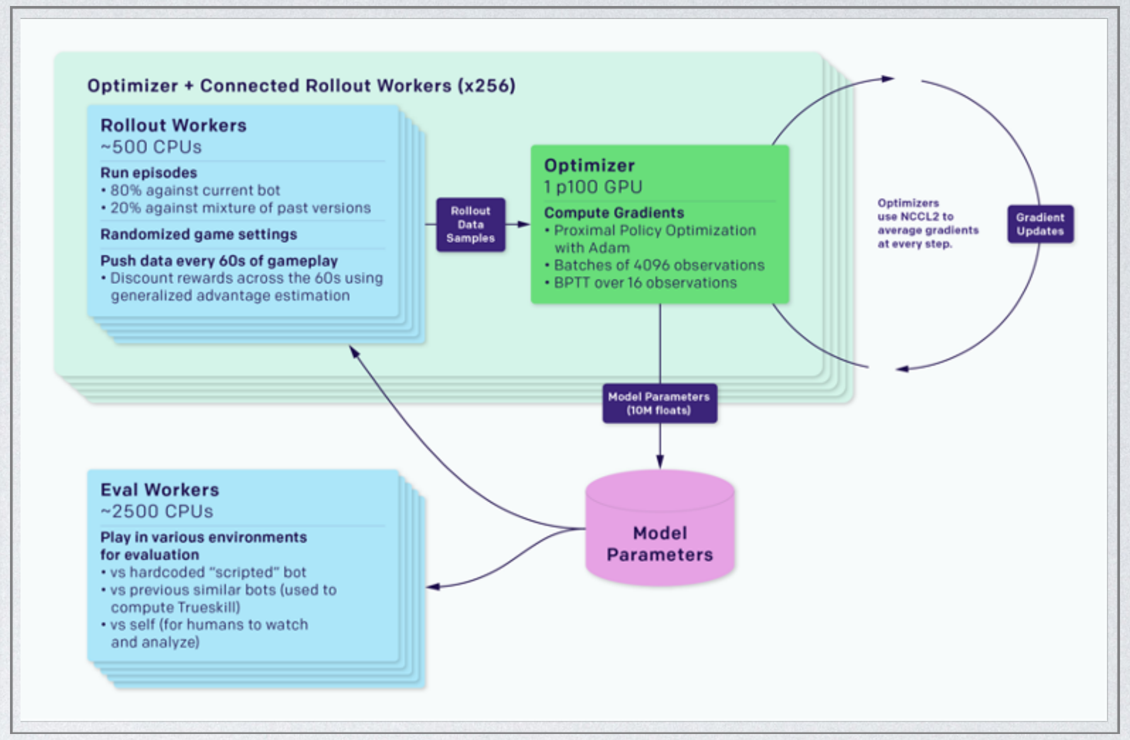
(a) Multiagent policy networks (b) Multiagent Q networks

StarCraft micro-management

BiCNet, deep MARL methods

1-2 GPUs, **1-2** days

2017.1



Optimizer + Connected Rollout Workers (x256)

Rollout Workers
~500 CPUs

Run episodes
- 80% against current bot
- 20% against mixture of past versions

Randomized game settings

Push data every 60s of gameplay
- Discount rewards across the 60s using generalized advantage estimation

Rollout Data Samples

Optimizer
1 p100 GPU

Compute Gradients
- Proximal Policy Optimization with Adam
- Batches of 4096 observations
- BPTT over 16 observations

Optimizers use NCCL2 to average gradients at every step.

Gradient Updates

Model Parameters (10M floats)

Model Parameters

Eval Workers
~2500 CPUs

Play in various environments for evaluation
- vs hardcoded "scripted" bot
- vs previous similar bots (used to compute Trueskill)
- vs self (for humans to watch and analyze)

Dota 2 full game (OpenAI Five)

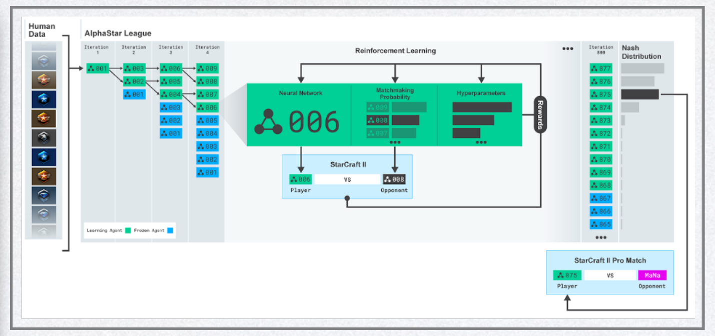
Population-based + Rapid training system

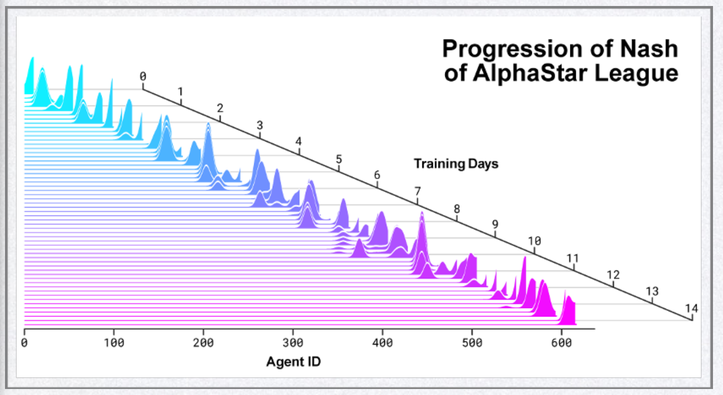
128,000 CPUs, **100** GPUs, **180** years of plays per day

2019.4

2019.1

2020.11

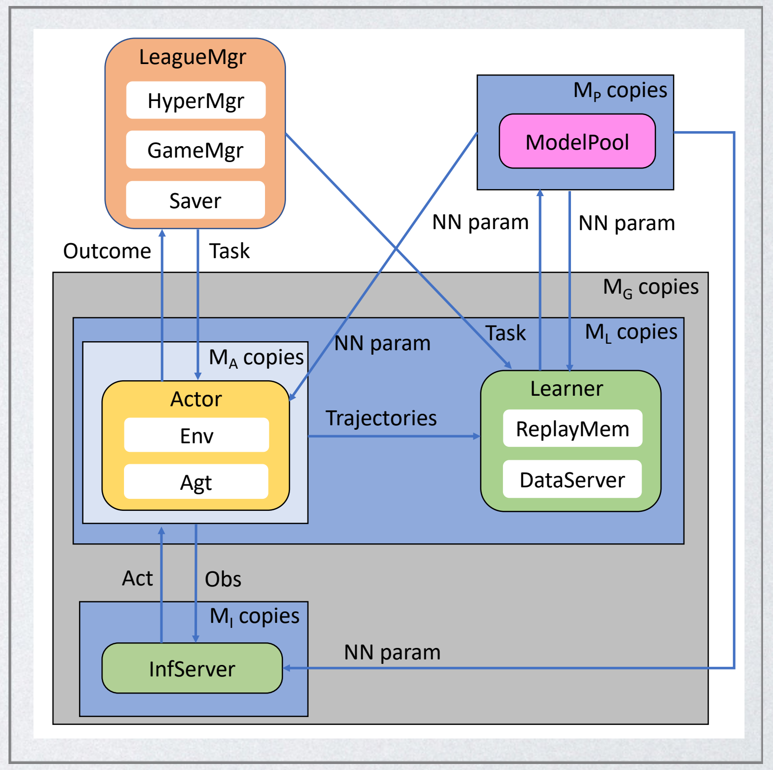




StarCraft full game (AlphaStar)

Populating-based Training

Training for single agent costs **14** days, **16** TPUs/Agent, **200** years of real-time play.

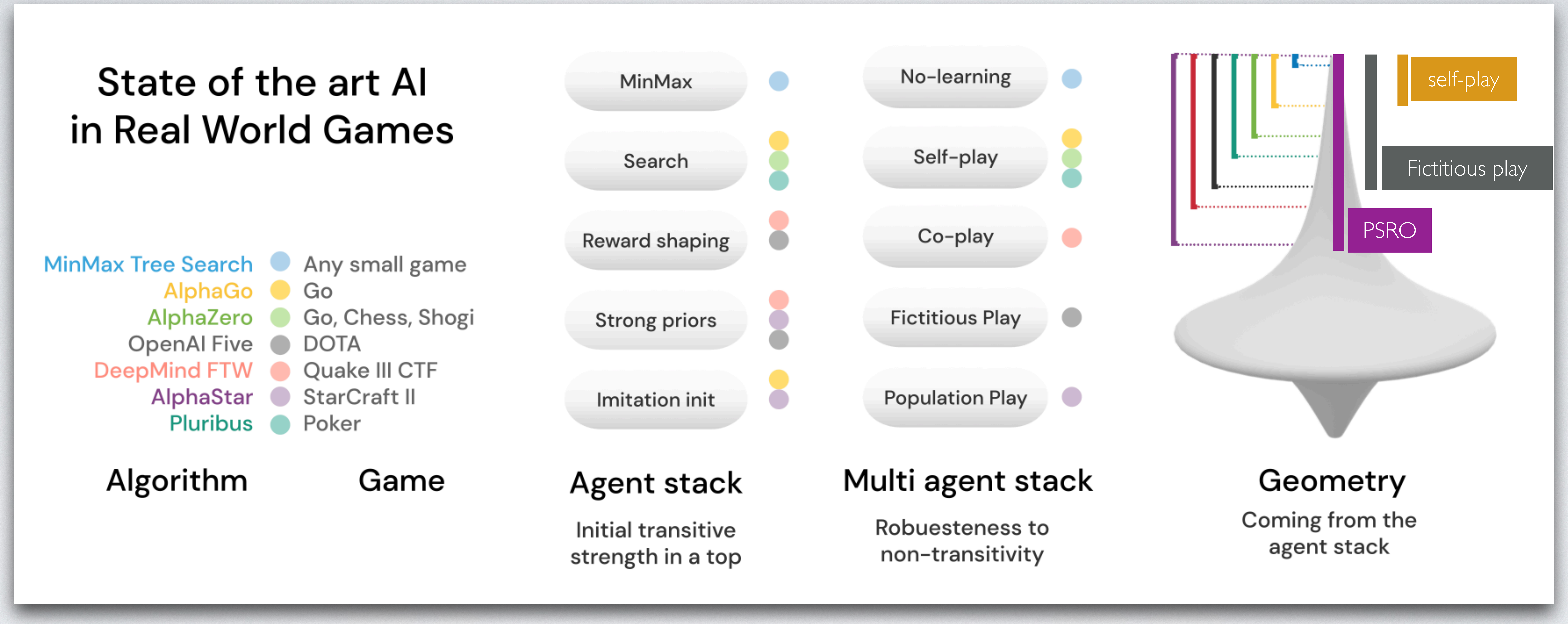


王者荣耀 (绝悟)

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 !



State of the art AI in Real World Games

Algorithm	Game
MinMax Tree Search	Any small game
AlphaGo	Go
AlphaZero	Go, Chess, Shogi
OpenAI Five	DOTA
DeepMind FTW	Quake III CTF
AlphaStar	StarCraft II
Pluribus	Poker

Agent stack

- MinMax
- Search
- Reward shaping
- Strong priors
- Imitation init

Initial transitive strength in a top

Multi agent stack

- No-learning
- Self-play
- Co-play
- Fictitious Play
- Population Play

Robuesteness to non-transitivity

Geometry

Coming from the agent stack

[Czarnecki 2020]

Solutions: Fictitious Play [Brown 1951]

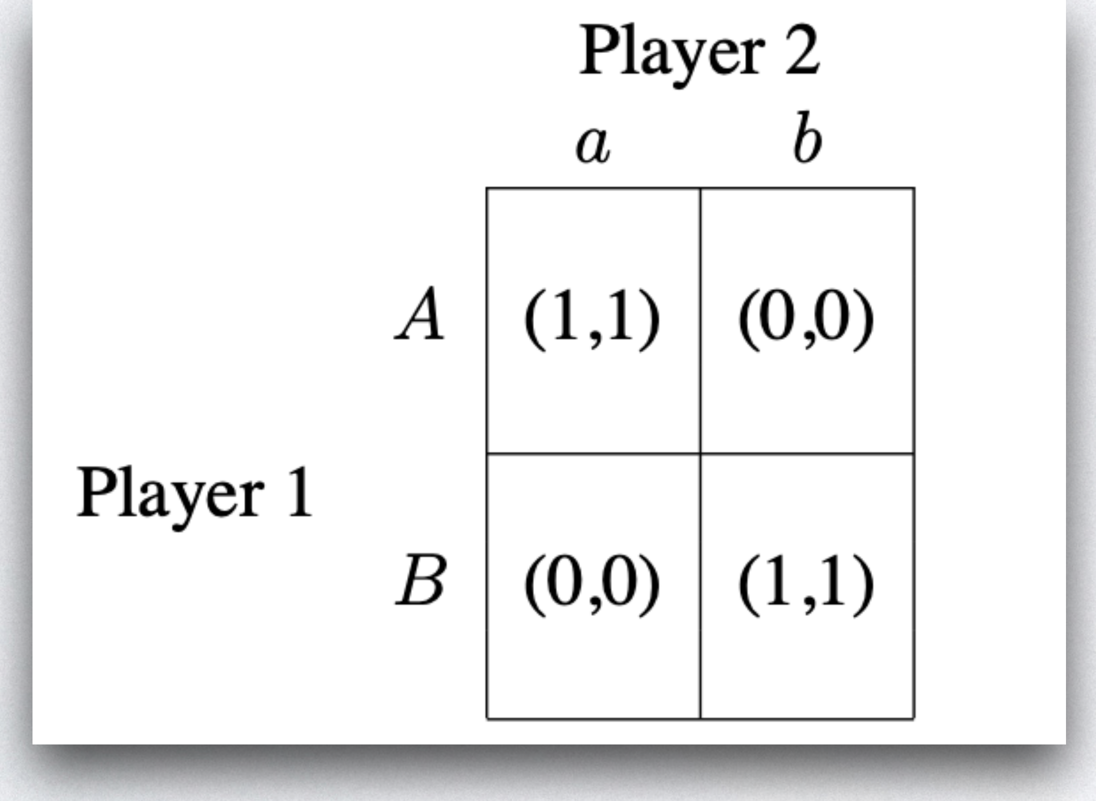
- Maintain a belief over the historical actions that the opponent has played, and the learning agent then takes the best response to this empirical average distribution.

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

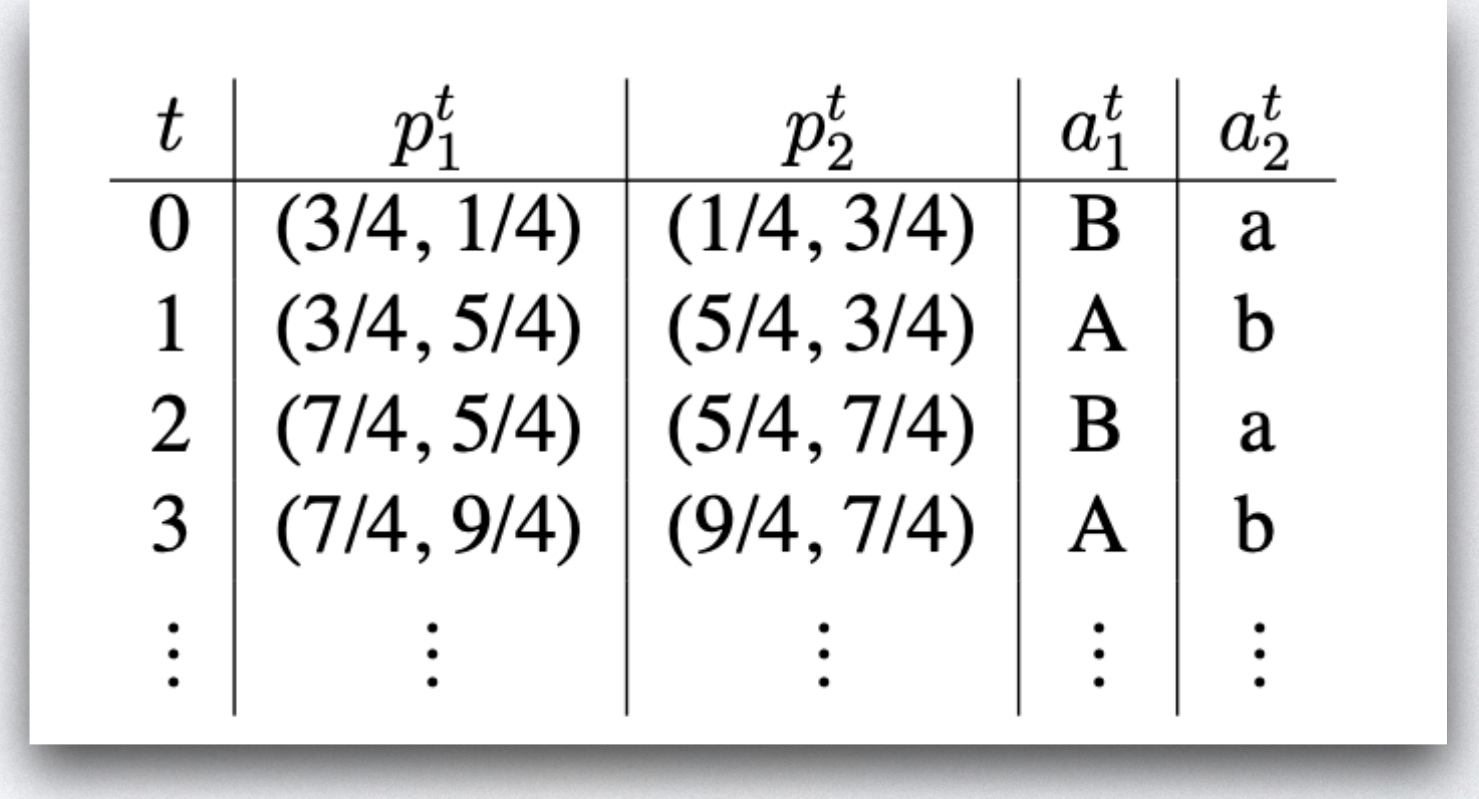
$$p_i^{t+1} = \left(1-\frac{1}{t}\right)p_i^t + \frac{1}{t}a_i^{t,*}, \text{ for all } i$$

- It guarantees to converge, in terms of the Nash value, in two-player zero-sum games, potential games and 2×2 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
⋮	⋮	⋮	⋮	⋮

∞ **(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.

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

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

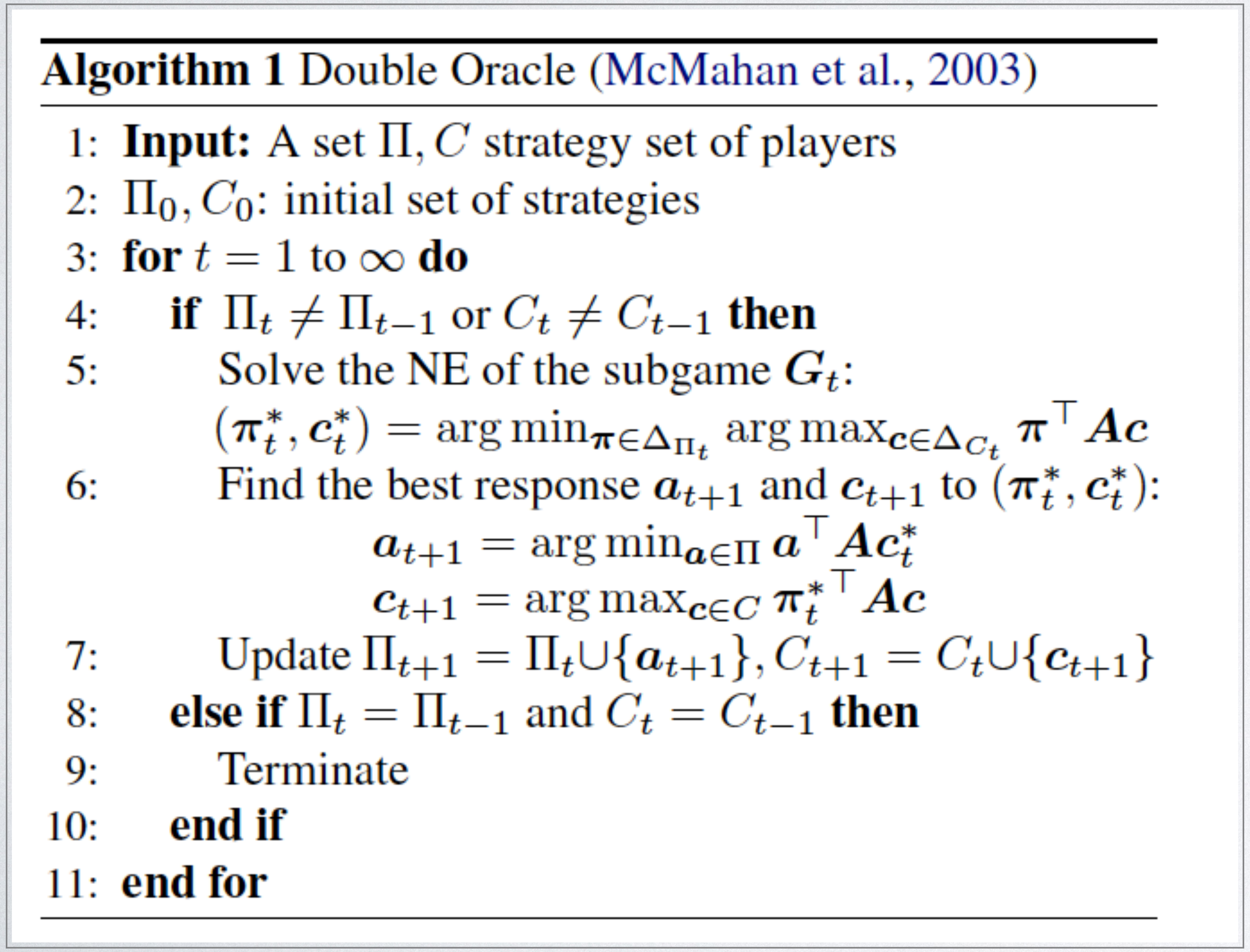
$$t \to \infty, \alpha_t \to 0, \epsilon^t \to 0, \sum_{t=1} \alpha^t = \infty, \{M^t\} \text{ meets } \lim_{t\to\infty} \sup_k \left\{ \left\| \sum_{i=t}^{k-1} \alpha^{i+1} M^{i+1} \right\| \text{ s.t. } \sum_{i=t}^{k-1} \alpha^{i+1} \leq T \right\} = 0$$

- 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]

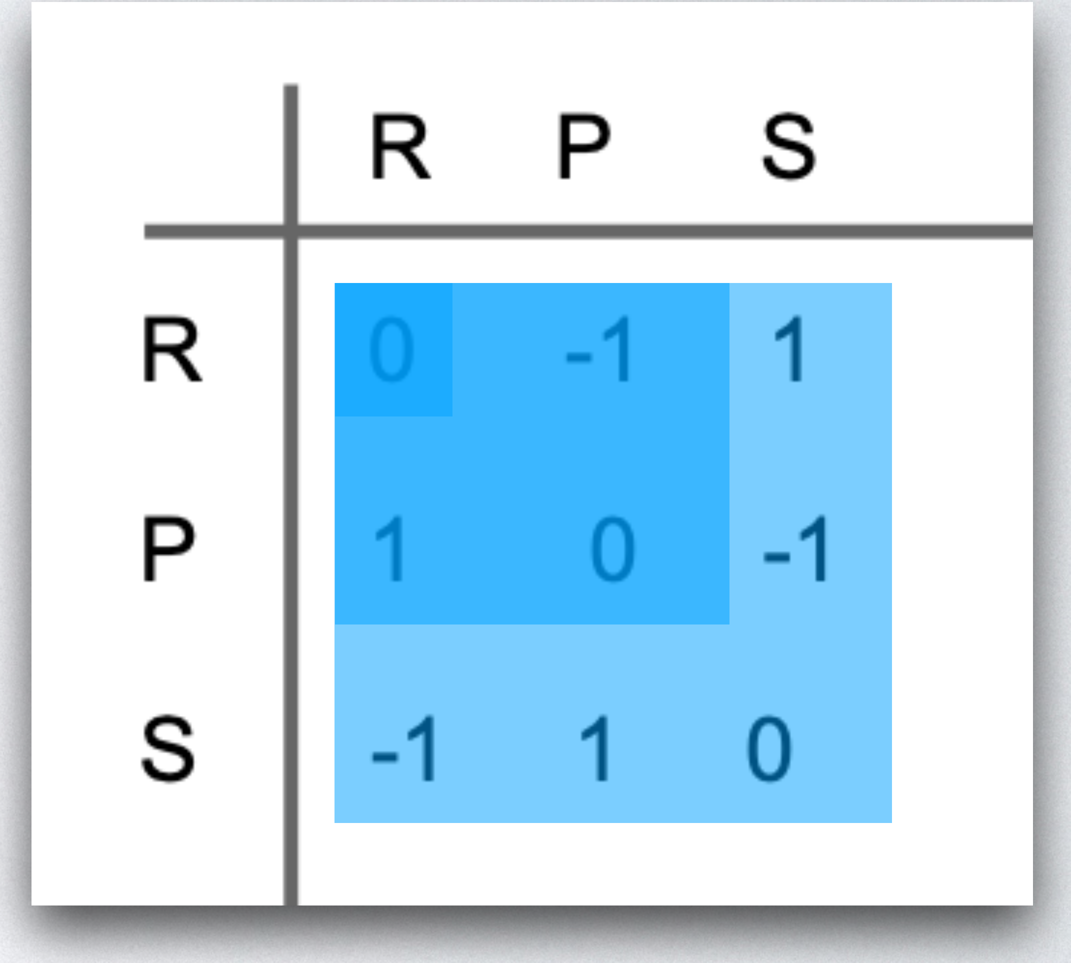
- Double Oracle best responds to the opponent's Nash equilibrium at each iteration.
- To solve the game before seeing all pure strategies (not all of them are in Nash), much faster than LP, but In the worst-case scenario, it recovers to solve the original game.



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

```
Input: A set $\Pi, C$ strategy set of players
$\Pi_0, C_0$: initial set of strategies
for $t = 1$ to $\infty$ do
    if $\Pi_t \neq \Pi_{t-1}$ or $C_t \neq C_{t-1}$ then
        Solve the NE of the subgame $G_t$:
        $(\pi_t^*, c_t^*) = \arg\min_{\pi \in \Delta_{\Pi_t}} \arg\max_{c \in \Delta_{C_t}} \pi^\top A c$
        Find the best response $a_{t+1}$ and $c_{t+1}$ to $(\pi_t^*, c_t^*)$:
                $a_{t+1} = \arg\min_{a \in \Pi} a^\top A c_t^*$
                $c_{t+1} = \arg\max_{c \in C} \pi_t^{*\top} A c$
        Update $\Pi_{t+1} = \Pi_t \cup \{a_{t+1}\}, C_{t+1} = C_t \cup \{c_{t+1}\}$
    else if $\Pi_t = \Pi_{t-1}$ and $C_t = C_{t-1}$ then
        Terminate
    end if
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 can be added
  - $\forall p, V(p, q_j) \geq v \Rightarrow \forall p, \max_q V(p, q) \geq v$

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

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

p_j must be the minimax optimal,

q_j vice versa

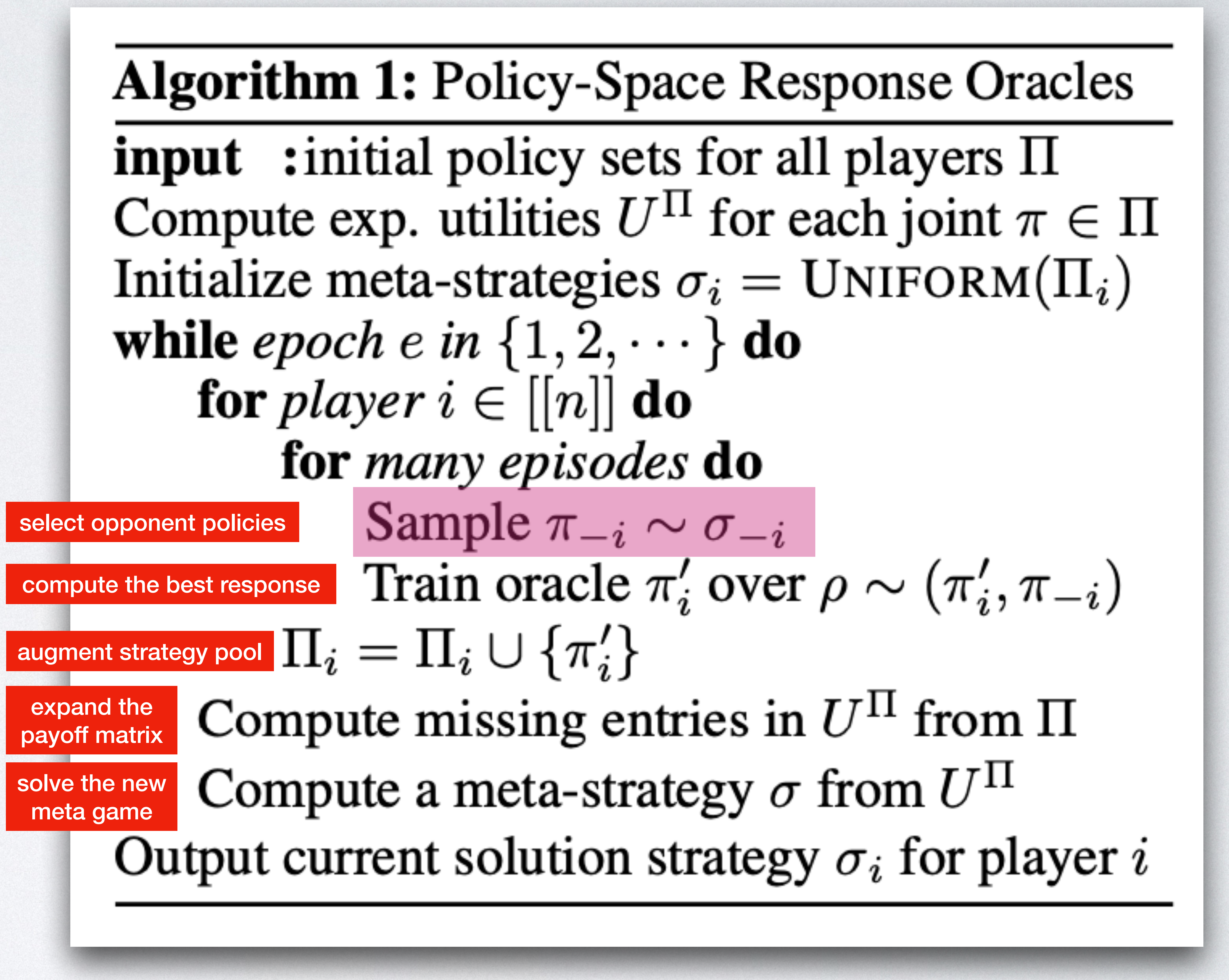
Policy Space Response Oracle = Double Oracle with RL Agent

- A generalisation of double oracle methods on meta-games, with the best responser is implemented through deep RL algorithms.

- A meta-game is (Π, U, n) where $\Pi = (\Pi_1, \ldots, \Pi_n)$ 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.

- σ_{-i} is distribution over $(\Pi_1^0, \ldots, \Pi_1^T)$, a.k.a meta-solver

- PSRO generalises all previous methods by varying σ_{-i}.
  - independent learning: $\sigma_{-i} = (0,\ldots,0,0,1)$
  - self-play: $\sigma_{-i} = (0,\ldots,0,1,0)$
  - fictitious play: $\sigma_{-i} = (1/T, 1/T, \ldots, 1/T, 0)$
  - PSRO: $\sigma_{-i} = \textbf{Nash}\left(\Pi^{T-1}, U\right)$ or $\textbf{RD}\left(\Pi^{T-1}, U\right)$



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 $\{1, 2, \cdots\}$ do
    for player $i \in [[n]]$ do
        for many episodes do
            Sample $\pi_{-i} \sim \sigma_{-i}$
            Train oracle $\pi_i'$ over $\rho \sim (\pi_i', \pi_{-i})$
        $\Pi_i = \Pi_i \cup \{\pi_i'\}$
    Compute missing entries in $U^\Pi$ from $\Pi$
    Compute a meta-strategy $\sigma$ from $U^\Pi$
Output current solution strategy $\sigma_i$ for player $i$
```

select opponent policies

compute the best response

augment strategy pool

expand the payoff matrix

solve the new meta game

Contents

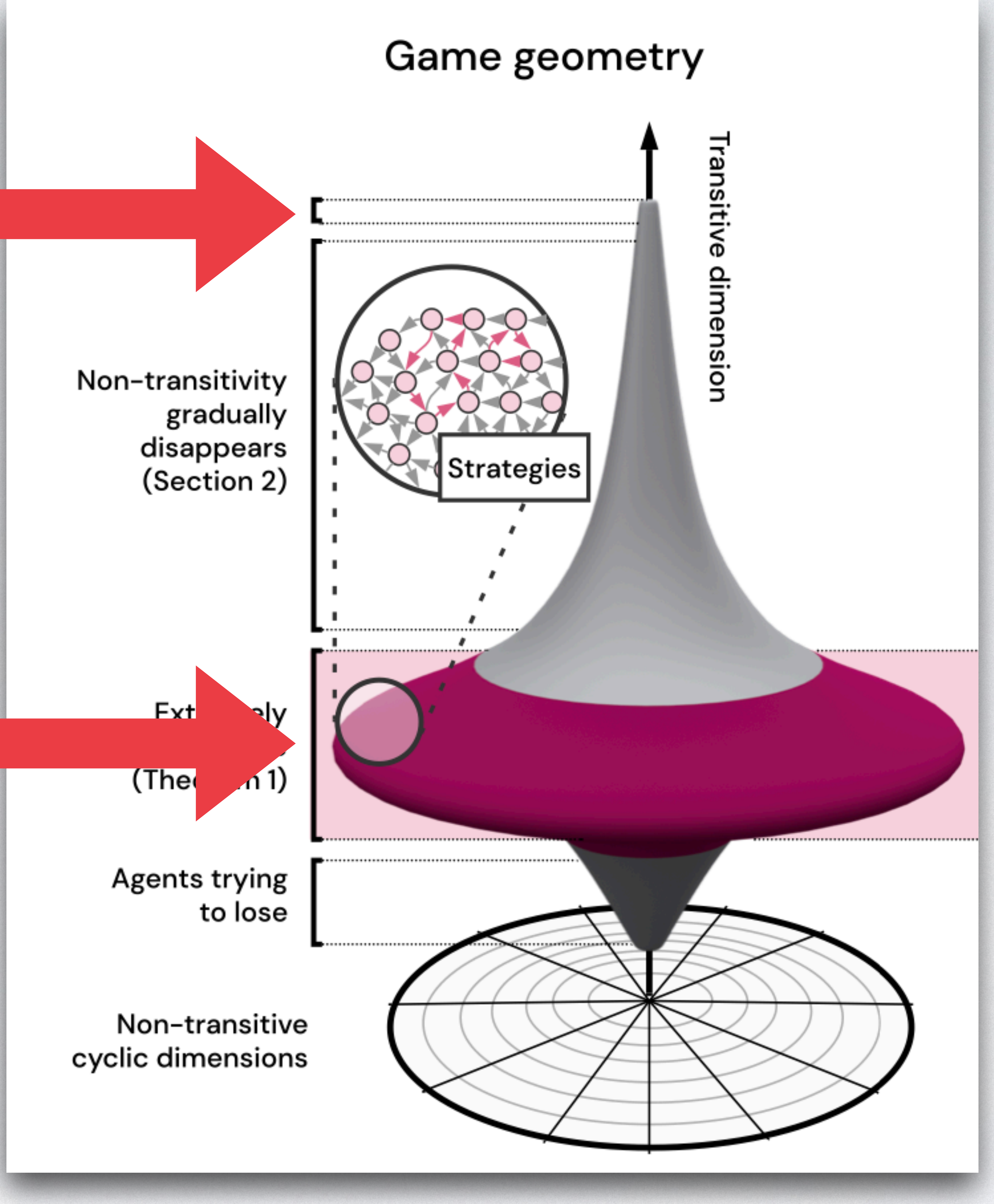
- **Rectified Nash**
- **Diverse PSRO**
- **PSRO with Behavioural Diversity**
- **Joint PSRO**
- **Pipeline PSRO**
- **Mixed Oracles / Opponents**
- **Neural Auto-curricula**

Meta-Game Structure [Czarnecki et al. 2020]

Interesting games display a particular spinning-top structure

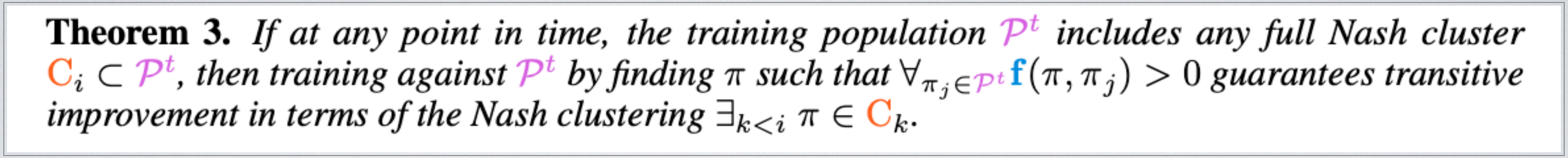
Diversity disappears and skill becomes the dominant factor, I.e. the game becomes fully transitive

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

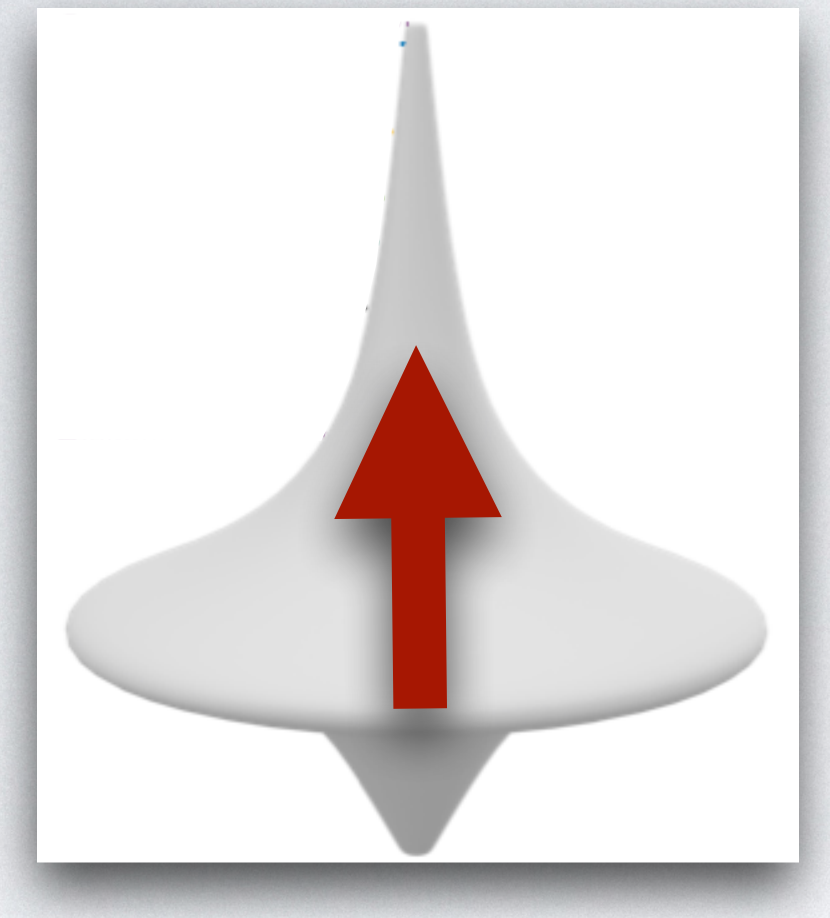

Game geometry
Transitive dimension
Non-transitivity gradually disappears (Section 2)
Strategies
Ext ely
(The n 1)
Agents trying to lose
Non-transitive cyclic dimensions

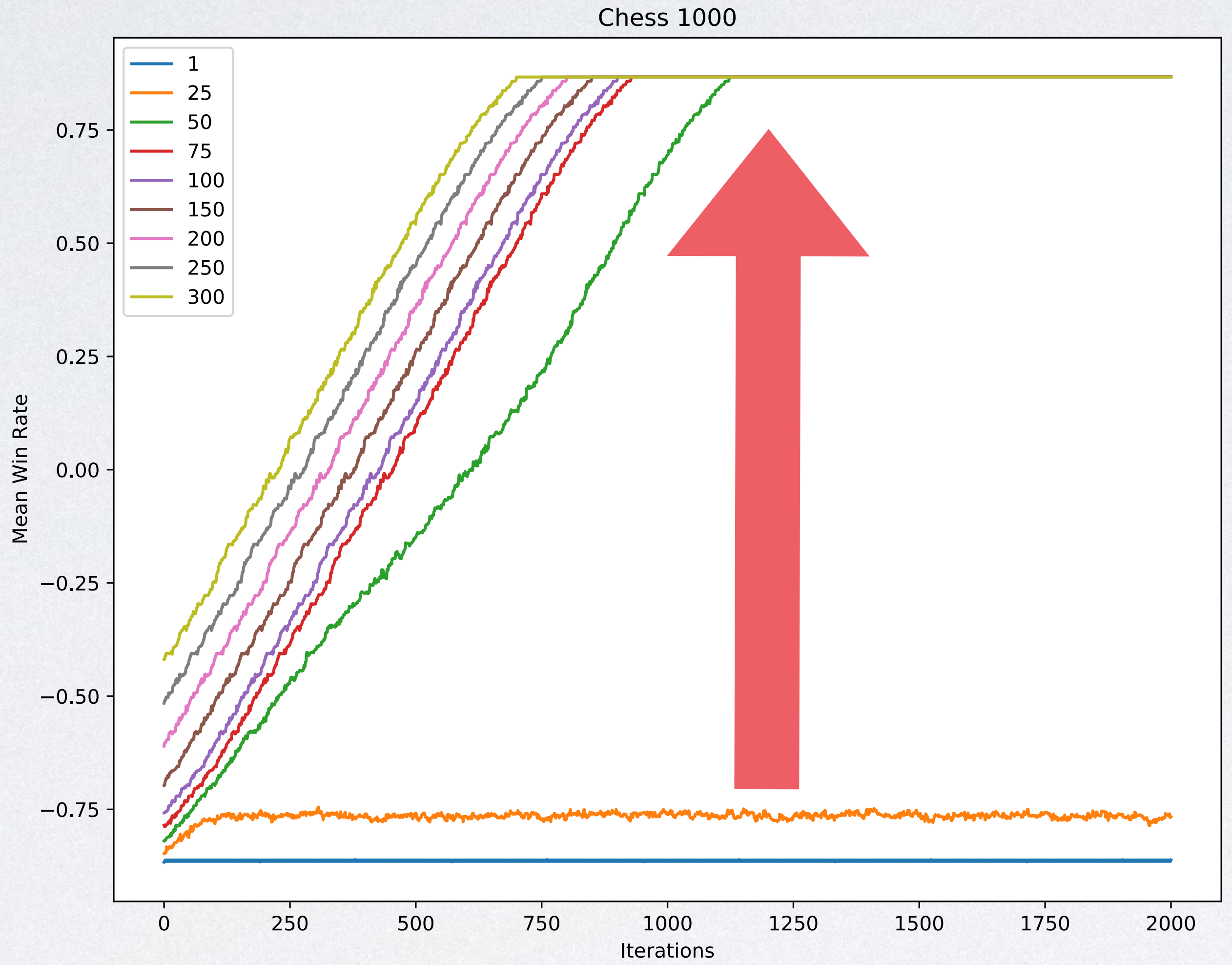
The big question is how does one move efficiently between the layers?

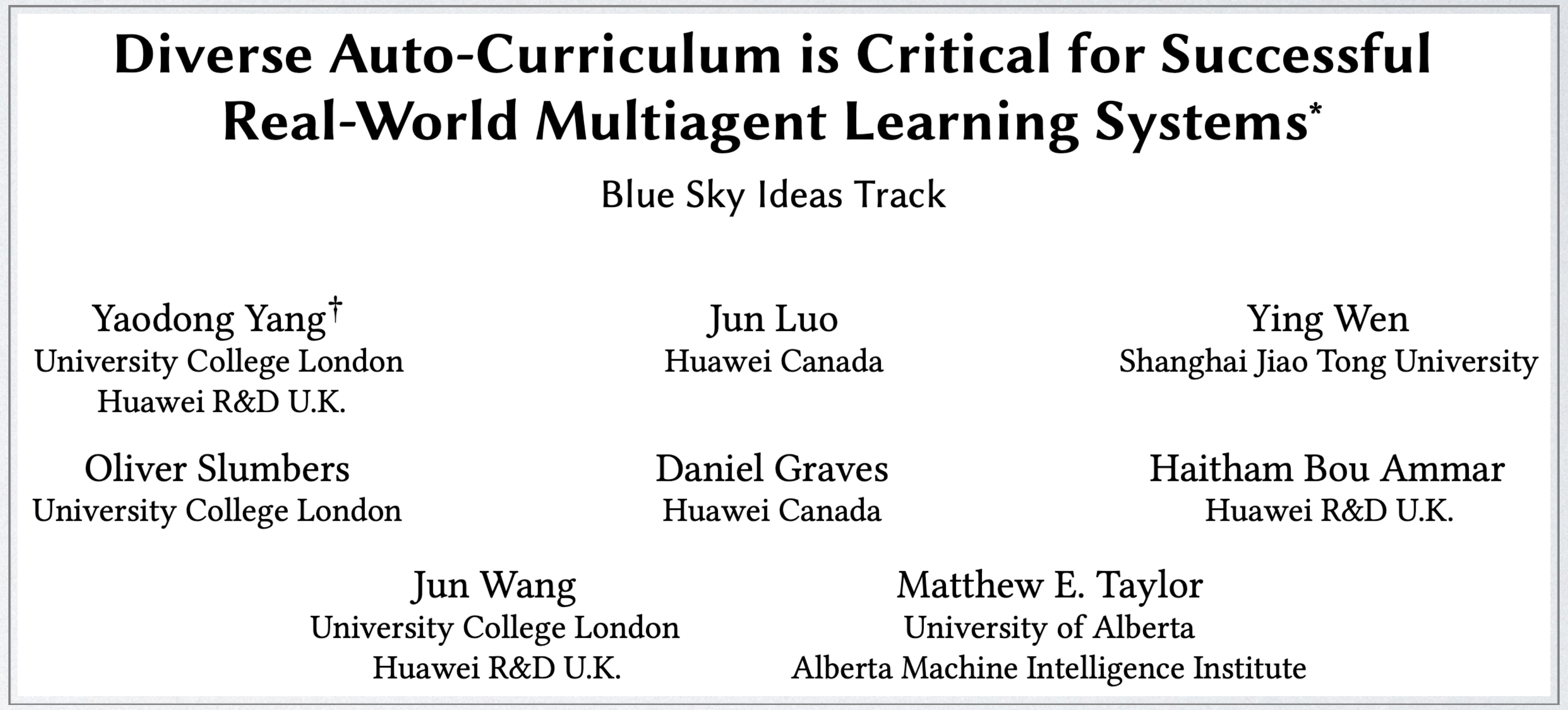
Why is Diversity Important?



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







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

Blue Sky Ideas Track

Yaodong Yang[†]
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.

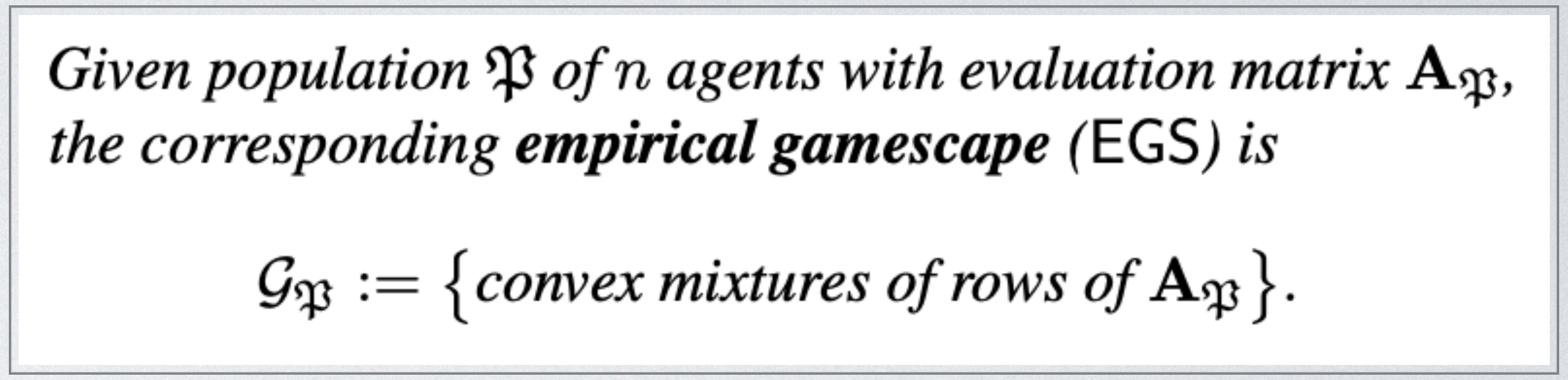
Matthew E. Taylor
University of Alberta
Alberta Machine Intelligence Institute

- 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

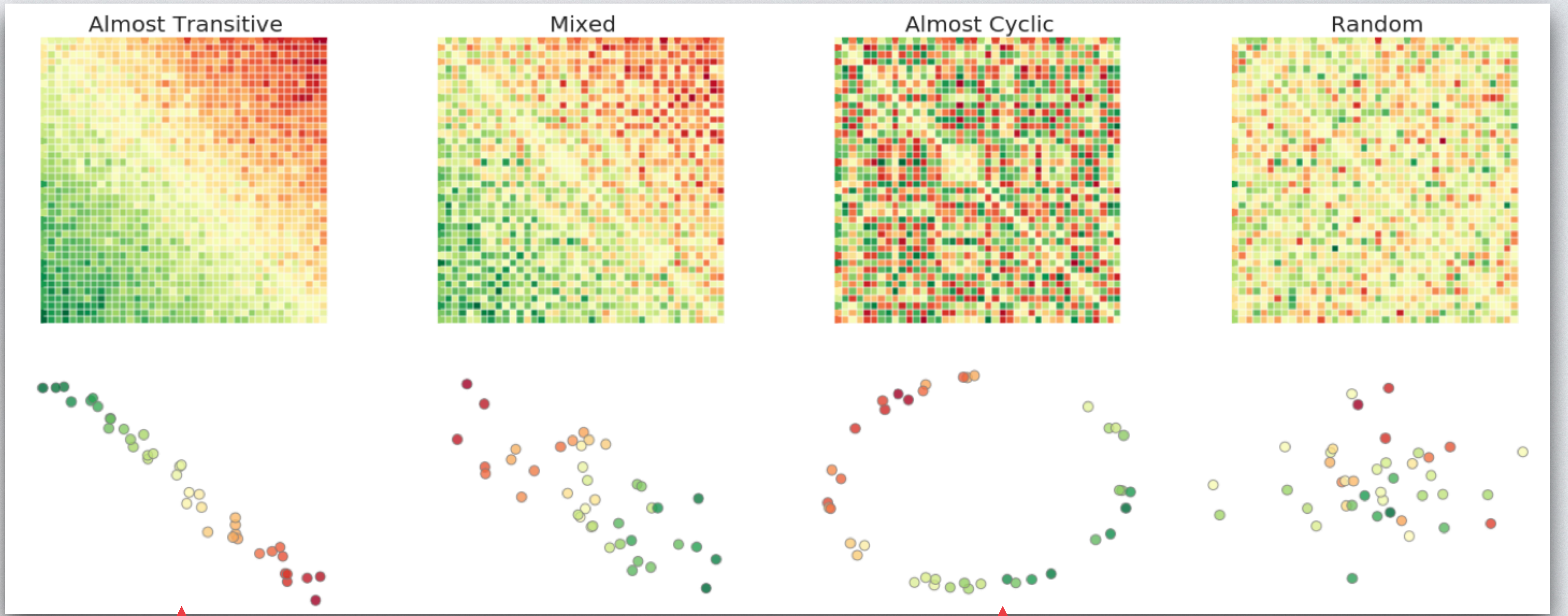
- A crucial component in characterising a population is that of the empirical gamescape
- Measure all ways agents can and are observed to interact with each-other



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

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

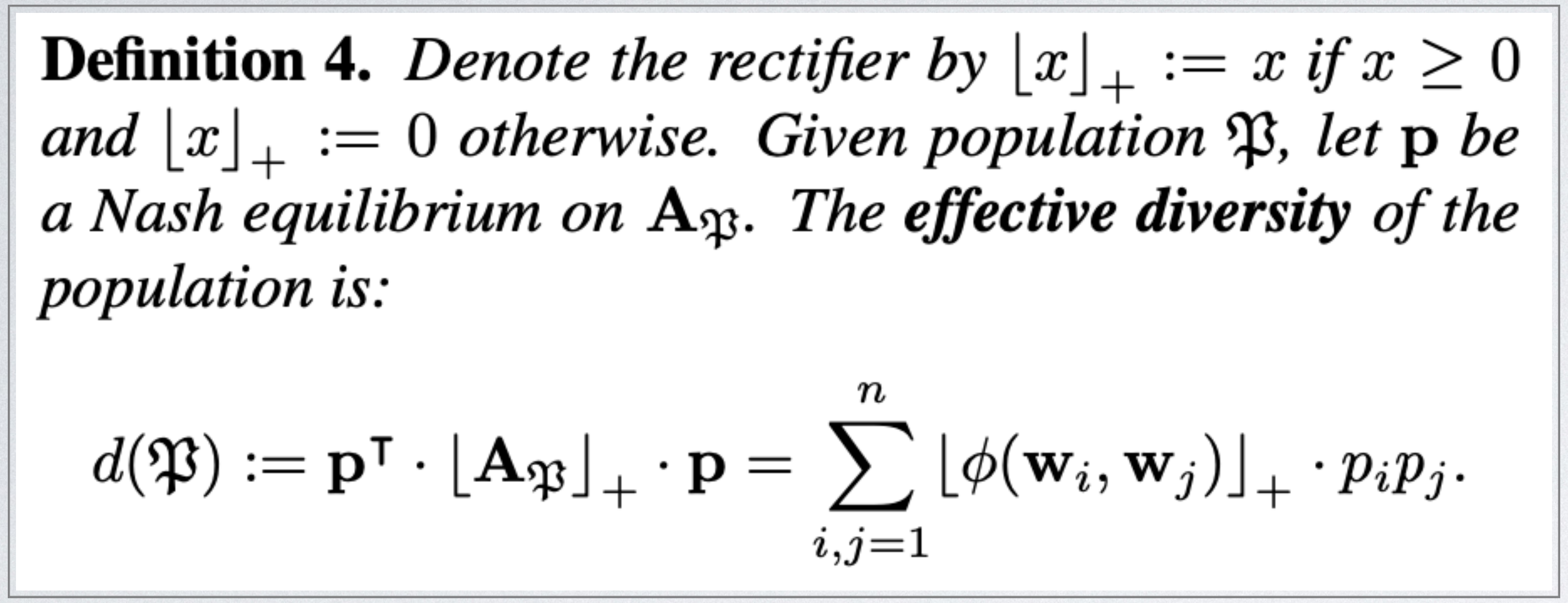
- Schur Decomposition of certain payoff matrices paints an intuitive picture
- Games show an obvious gamescape structure



Obvious linear/transitive structure

Obvious cyclic/non-transitive structure

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



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

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

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

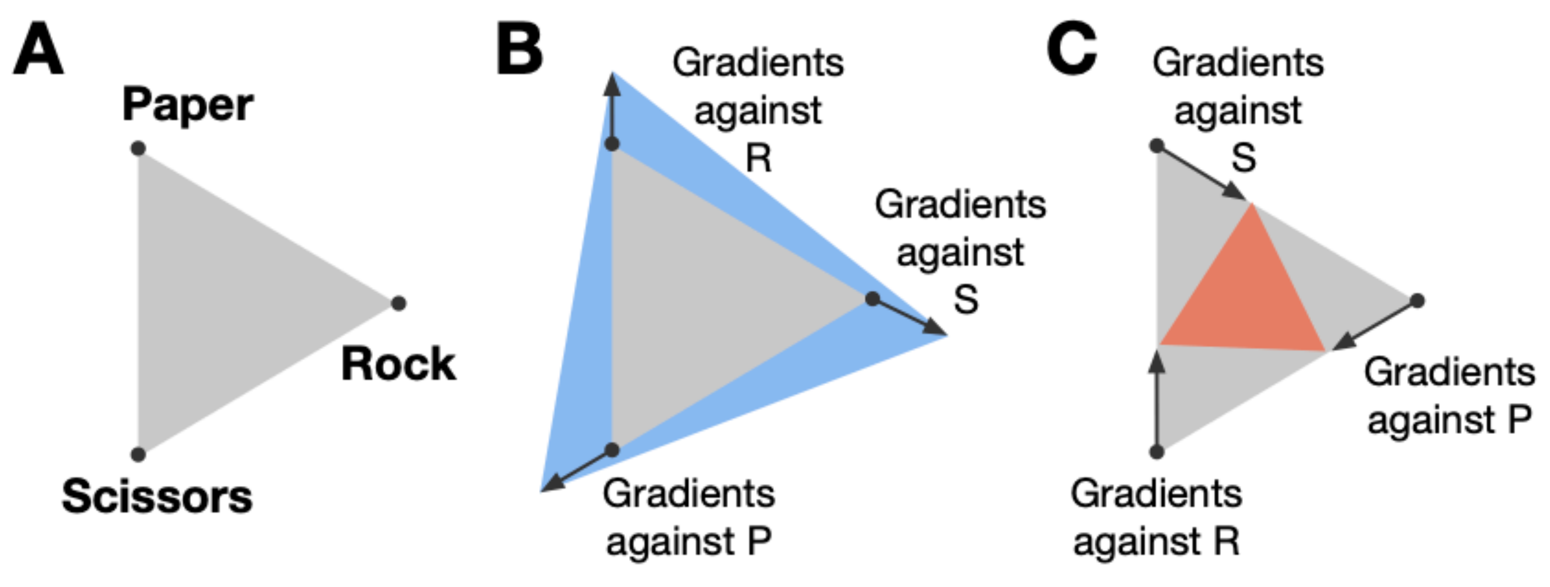
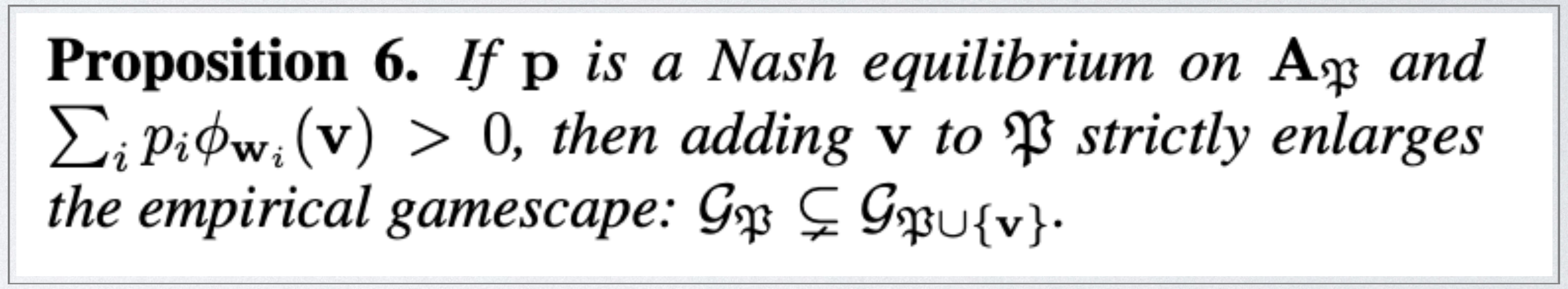


Figure 3. **A:** Rock-paper-scissors. **B:** Gradient updates obtained from PSRO$_{\text{rN}}$, amplifying strengths, grow gamescape (gray to blue). **C:** Gradients obtained by optimizing agents to reduces their losses shrink gamescape (gray to red).

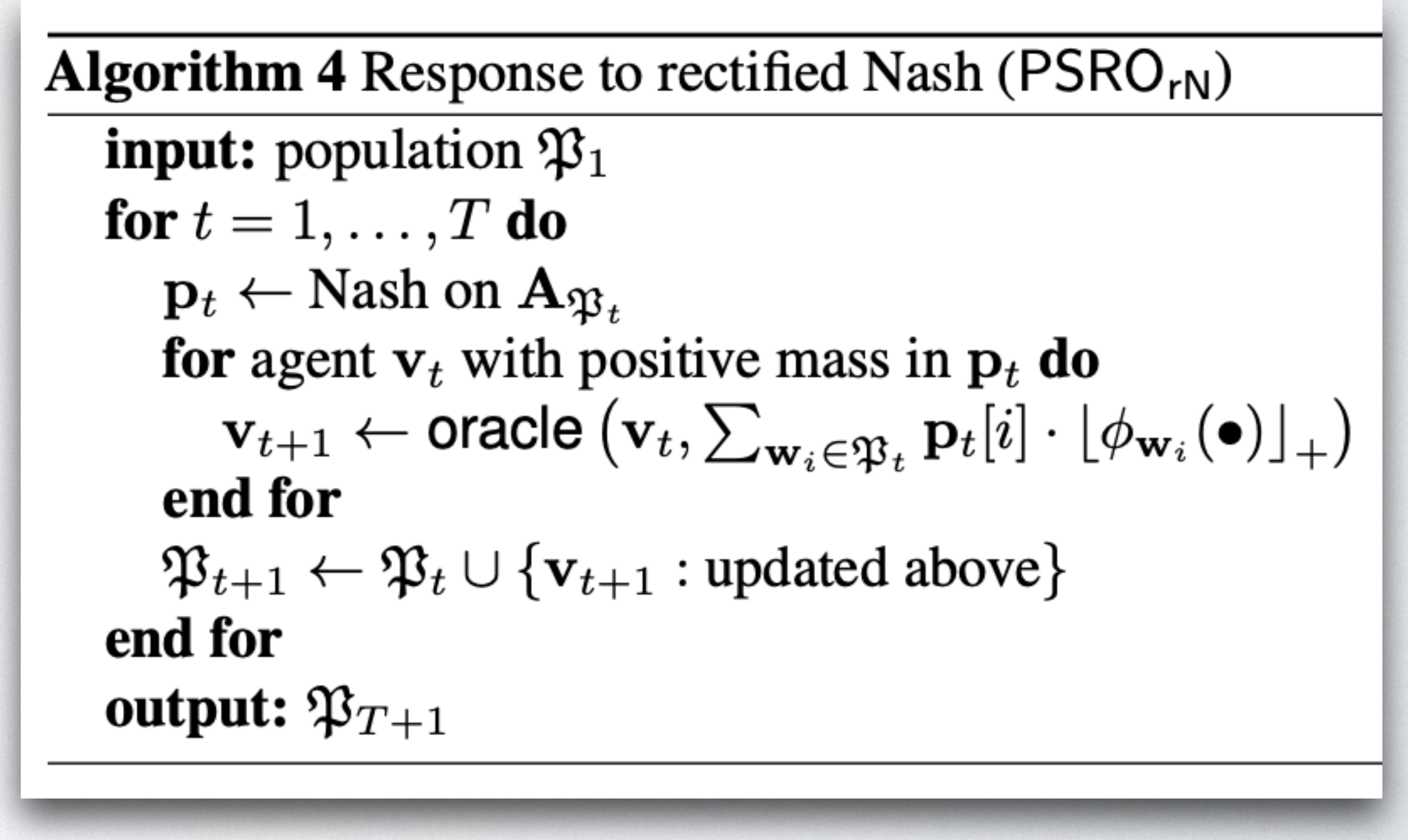
Intuition: improving ones strengths allows for exploration of the strategy space

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

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



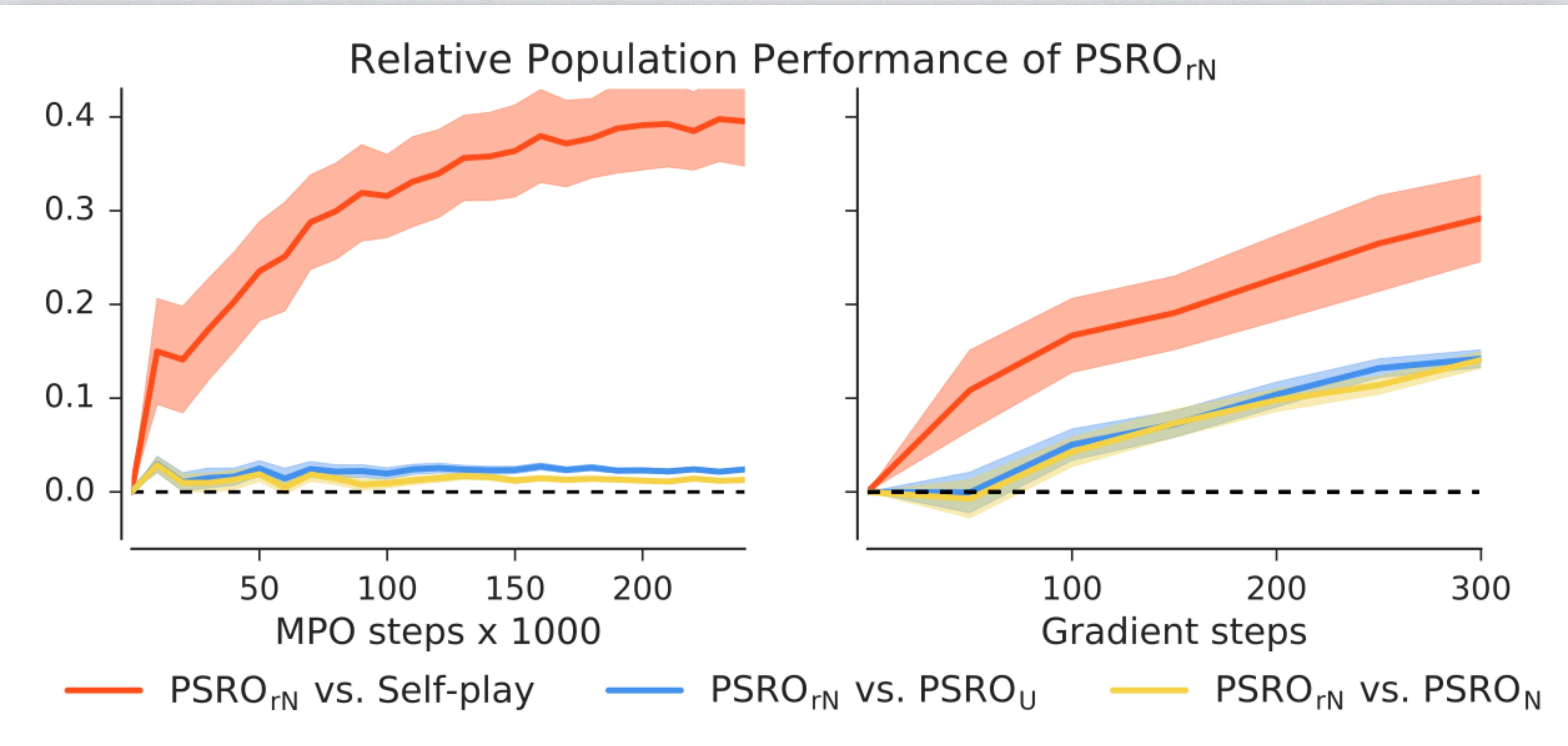
Proposition 6. *If $\mathbf{p}$ is a Nash equilibrium on $\mathbf{A}_{\mathfrak{P}}$ and $\sum_i p_i \phi_{\mathbf{w}_i}(\mathbf{v}) > 0$, then adding $\mathbf{v}$ to $\mathfrak{P}$ strictly enlarges the empirical gamescape: $\mathcal{G}_{\mathfrak{P}} \subsetneq \mathcal{G}_{\mathfrak{P}\cup\{\mathbf{v}\}}$.*

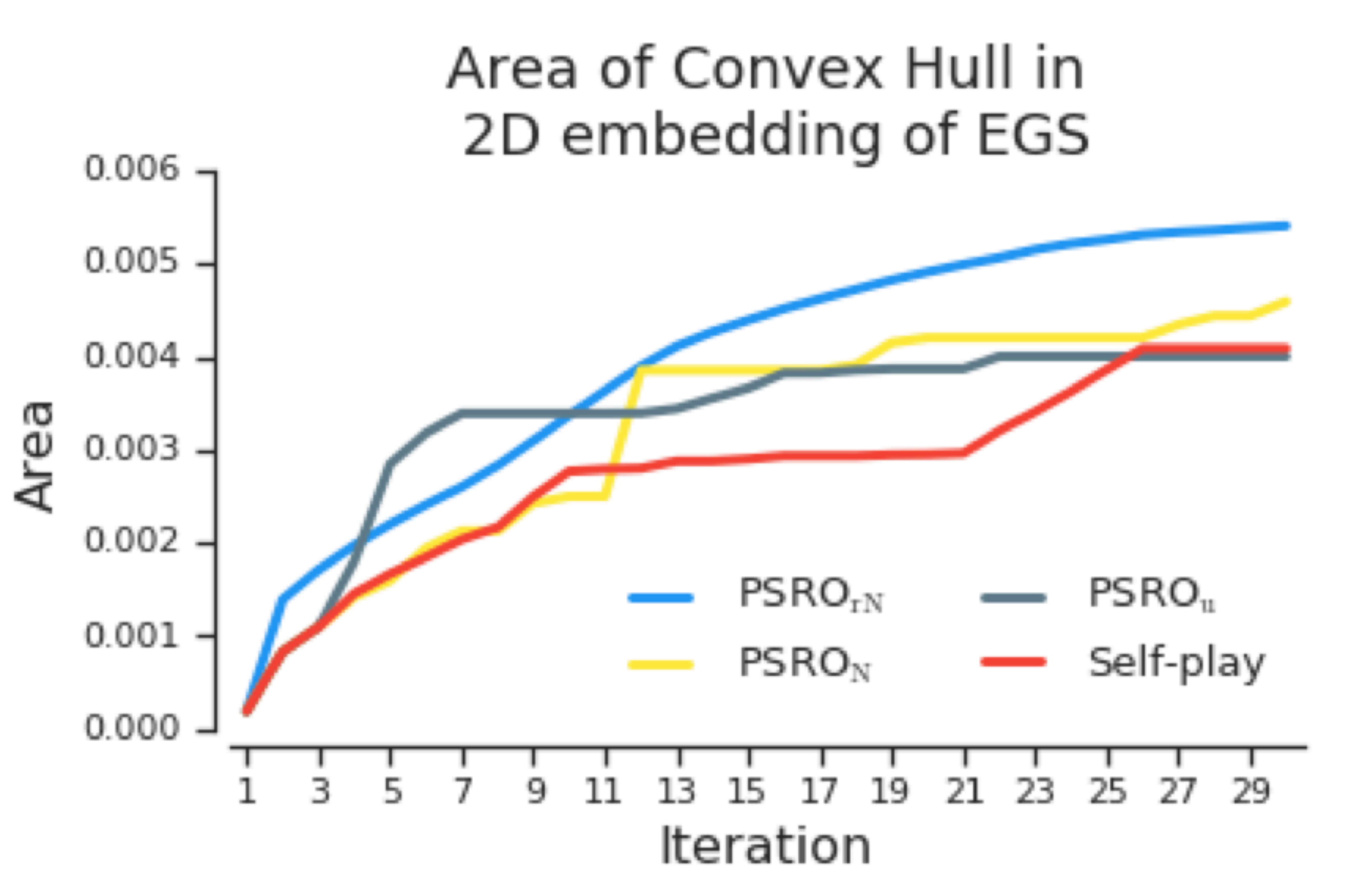


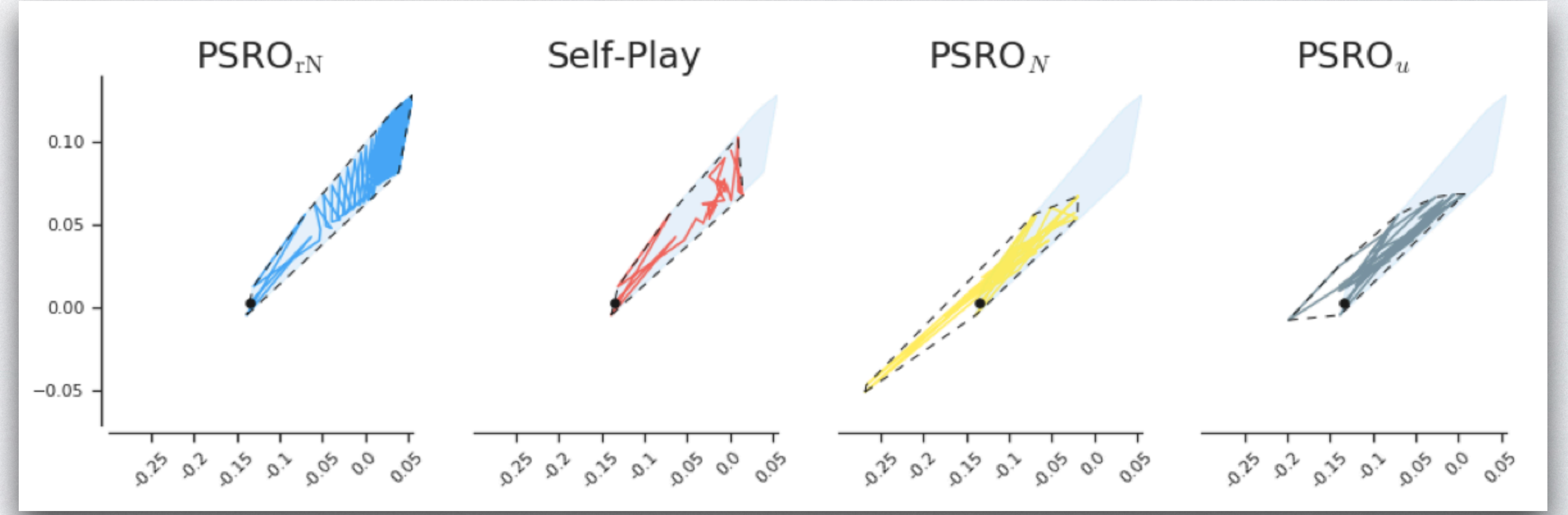
Algorithm 4 Response to rectified Nash (PSRO$_{\text{rN}}$)

```
input: population $\mathfrak{P}_1$
for $t = 1, \ldots, T$ 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
    $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \sum_{\mathbf{w}_i \in \mathfrak{P}_t} \mathbf{p}_t[i] \cdot \lfloor \phi_{\mathbf{w}_i}(\bullet) \rfloor_+\right)$
  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





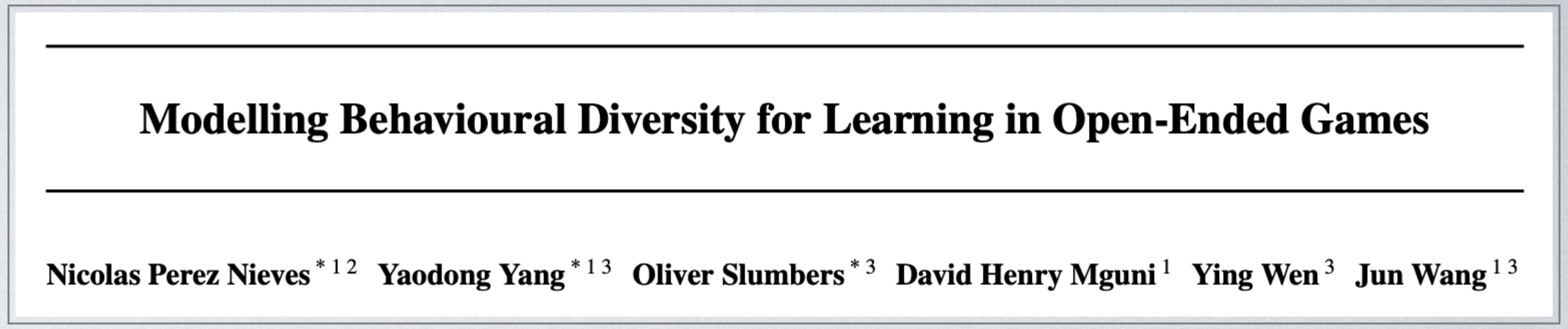


Diversity helps in exploring the strategy space more efficiently and effectively

Contents

- **Rectified Nash**
- **Diverse PSRO**
- **PSRO with Behavioural Diversity**
- **Joint PSRO**
- **Pipeline PSRO**
- **Mixed Oracles / Opponents**
- **Neural Auto-curricula**

Diverse-PSRO

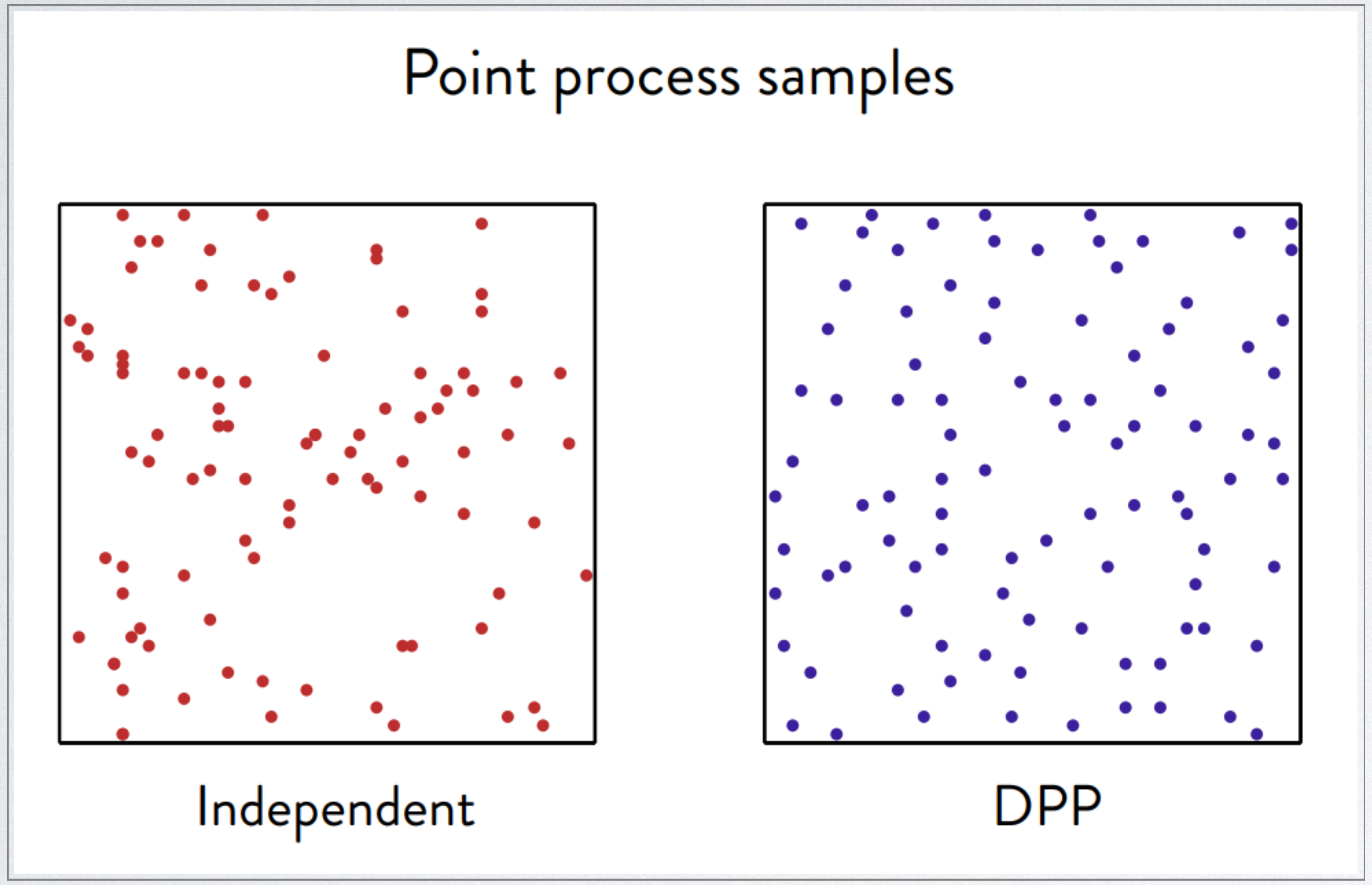


Modelling Behavioural Diversity for Learning in Open-Ended Games

Nicolas Perez Nieves[*12] Yaodong Yang[*13] Oliver Slumbers[*3] David Henry Mguni[1] Ying Wen[3] Jun Wang[13]

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.



Point process samples

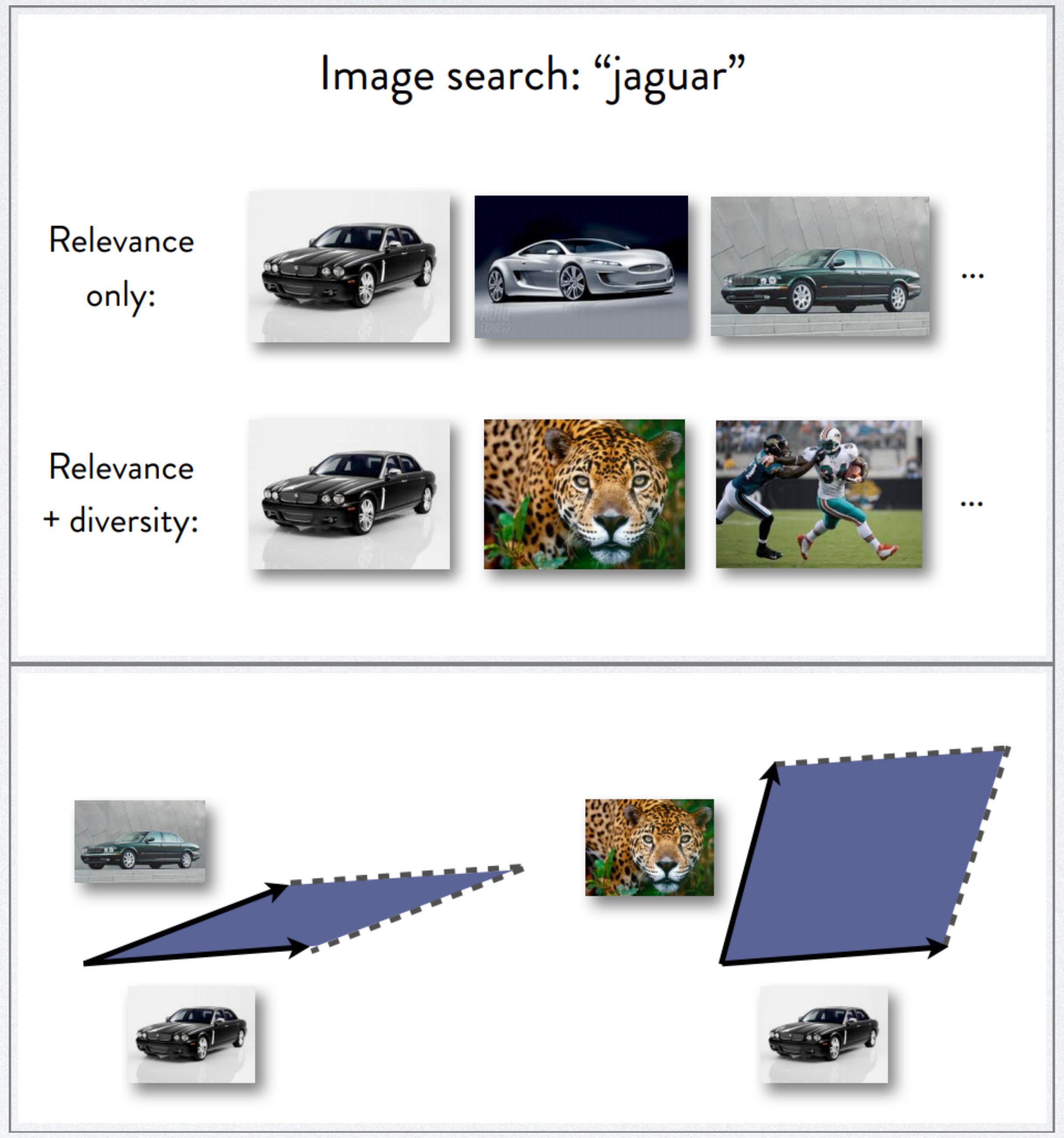
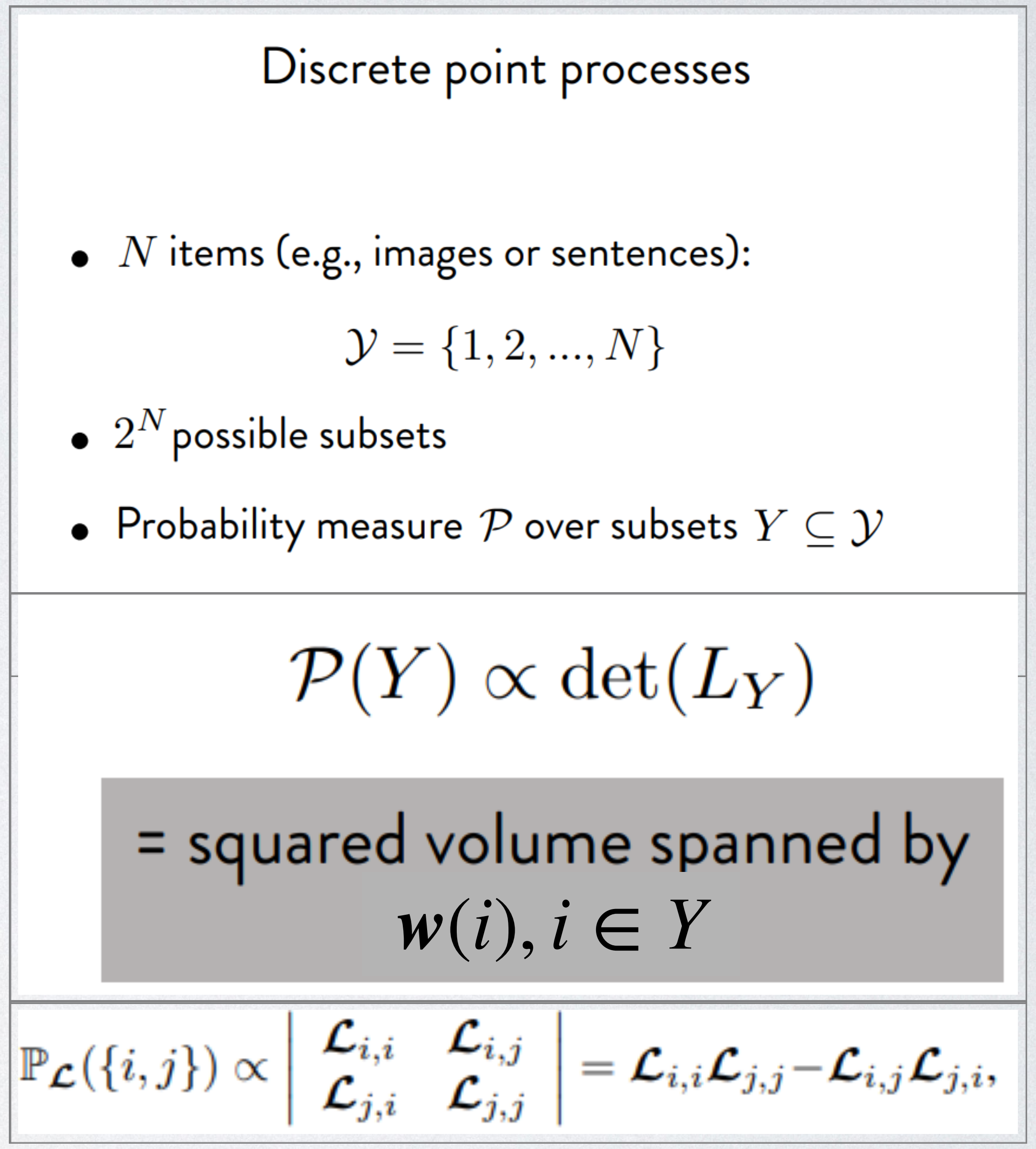


Image search: “jaguar”



Discrete point processes

- N items (e.g., images or sentences):

$$\mathcal{Y} = \{1, 2, ..., N\}$$

- 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 $\boldsymbol{w}(i), i \in Y$

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

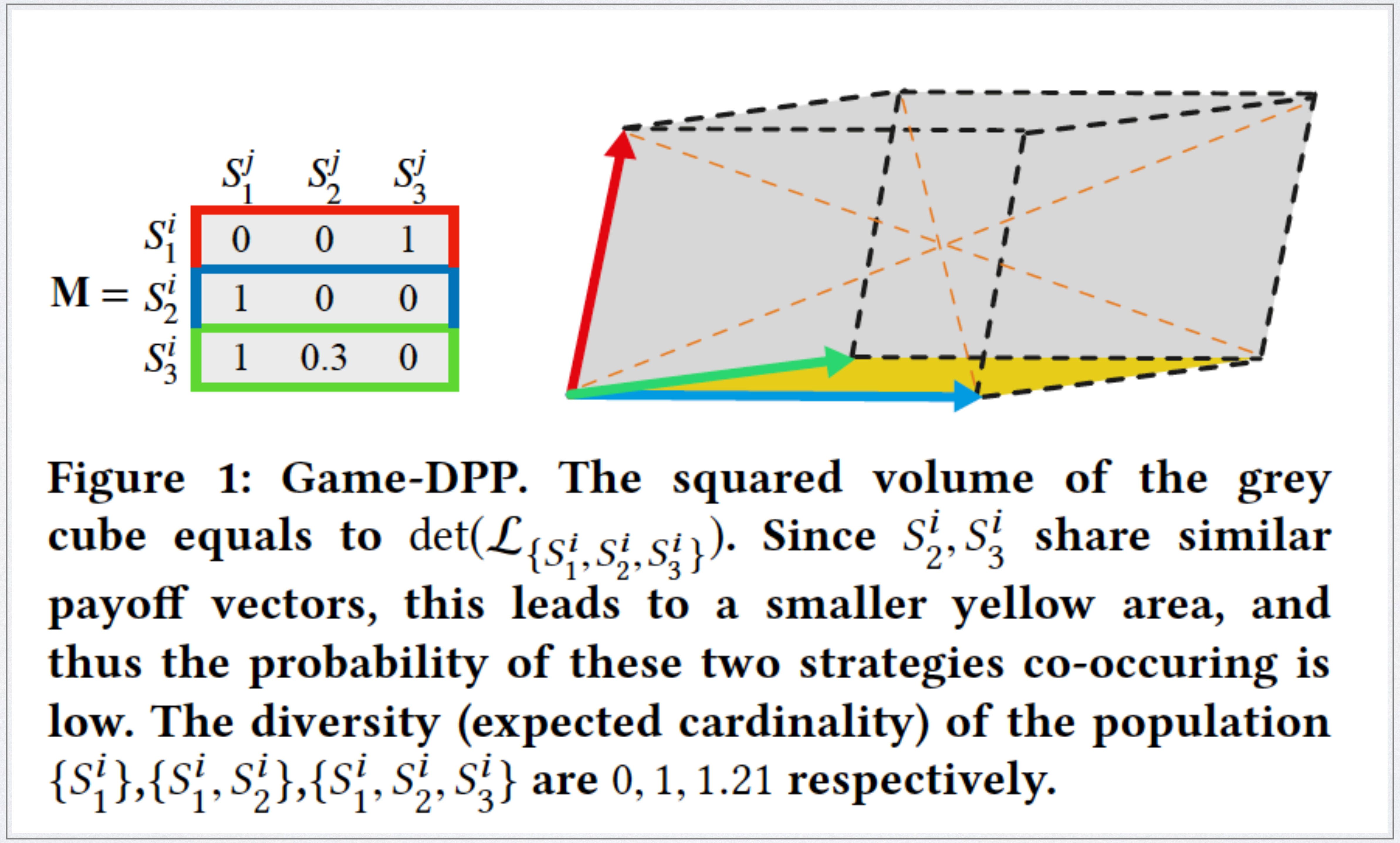
$$\boxed{\text{DPP}(\mathscr{L}) := \mathbb{P}_{\mathscr{L}}(\boldsymbol{Y} = Y) \propto \det(\mathscr{L}_Y) = \text{Vol}^2\left(\{\boldsymbol{w}_i\}_{i \in Y}\right)}$$

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., $\mathscr{L}_{\mathbb{S}} = \mathbf{M}\mathbf{M}^{\top}$.
- The expected cardinality of the DPP is the diversity metric.

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



$$\mathbf{M} = \begin{array}{c|ccc} & S_1^j & S_2^j & S_3^j \\ \hline S_1^i & 0 & 0 & 1 \\ S_2^i & 1 & 0 & 0 \\ S_3^i & 1 & 0.3 & 0 \end{array}$$

Figure 1: Game-DPP. The squared volume of the grey cube equals to $\det(\mathcal{L}_{\{S_1^i,S_2^i,S_3^i\}})$. **Since** S_2^i, S_3^i **share similar payoff vectors, this leads to a smaller yellow area, and thus the probability of these two strategies co-occuring is low. The diversity (expected cardinality) of the population** $\{S_1^i\},\{S_1^i,S_2^i\},\{S_1^i,S_2^i,S_3^i\}$ **are** $0, 1, 1.21$ **respectively.**

Diverse-PSRO

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

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

  - Diverse PSRO

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

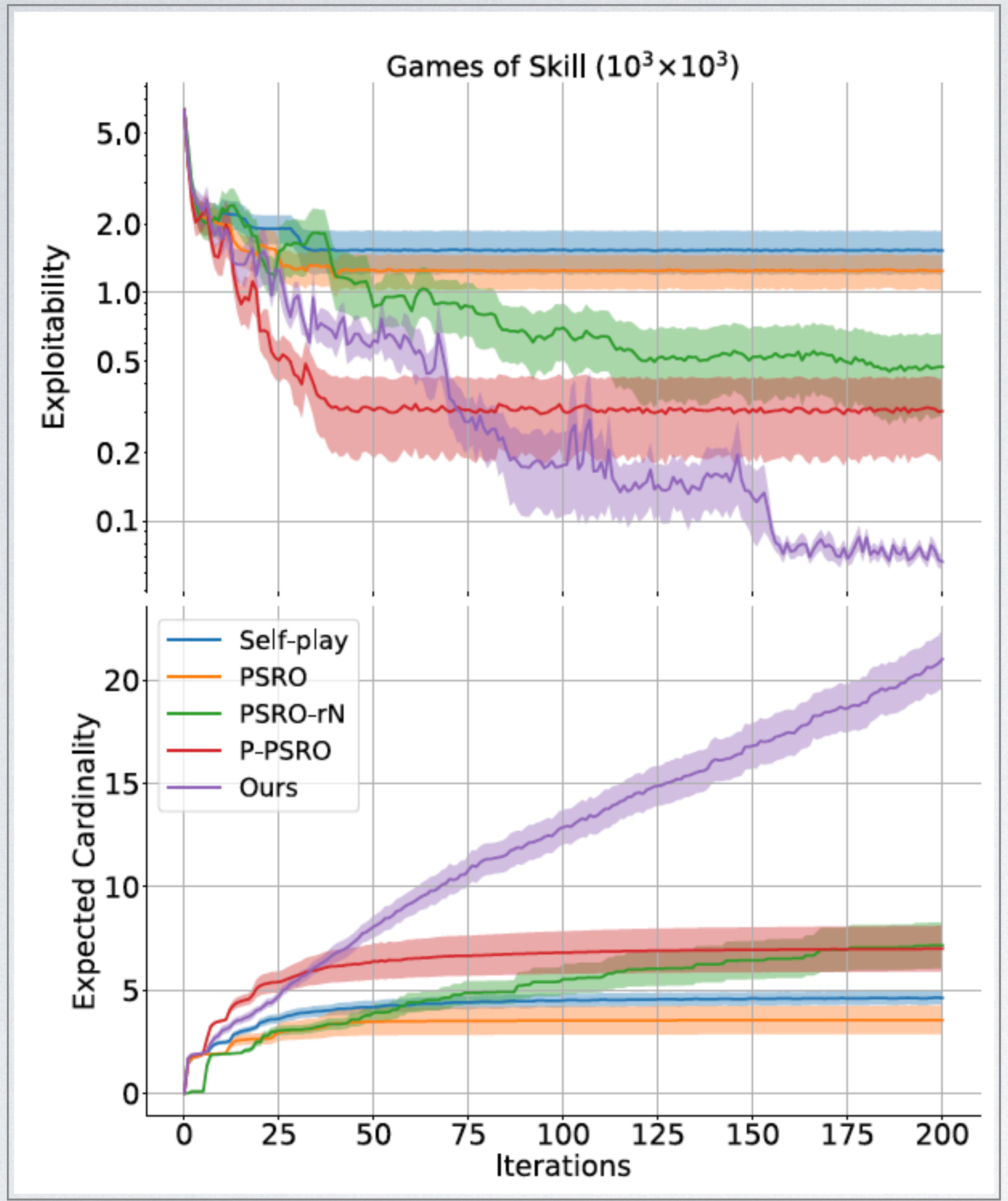
  - Diverse α-PSRO (α-Rank as meta-solver)

$$\mathcal{O}\left(\pi^2\right) = \mathrm{argmax}_{\pi\in\Delta_{S^i}} \mathrm{Tr}\left(I - (\mathscr{L}_{\mathbb{S}^i_t\cup\{\pi\}} + I)^{-1}\right)$$

  - Importantly, we prove that

$$\boxed{\text{Gamescape}\left(\mathbb{S}\right) \subsetneq \text{Gamescape}\left(\mathbb{S} \cup \left\{S_\theta\right\}\right)}$$

Diverse-PSRO



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

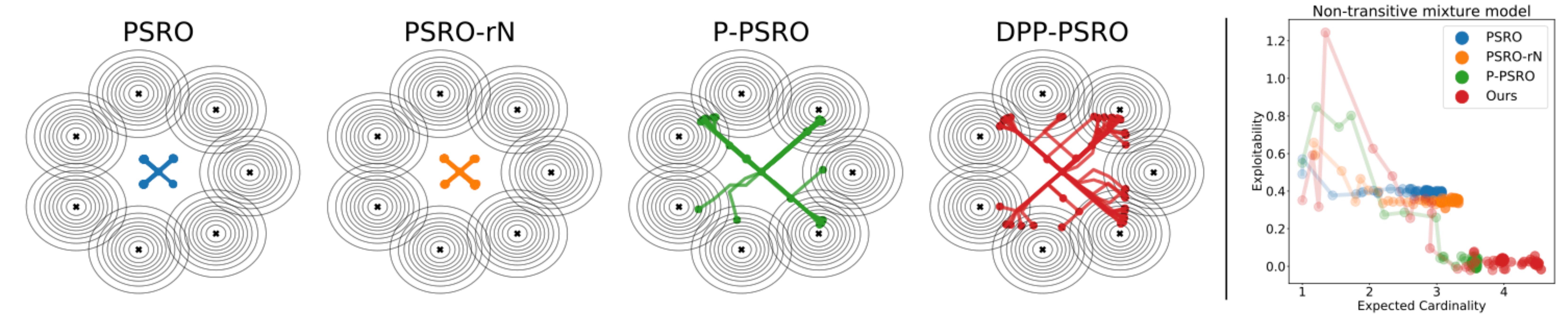


Figure 3. Non-transitive mixture model. Exploration trajectories during training and Performance vs. Diversity comparisons.

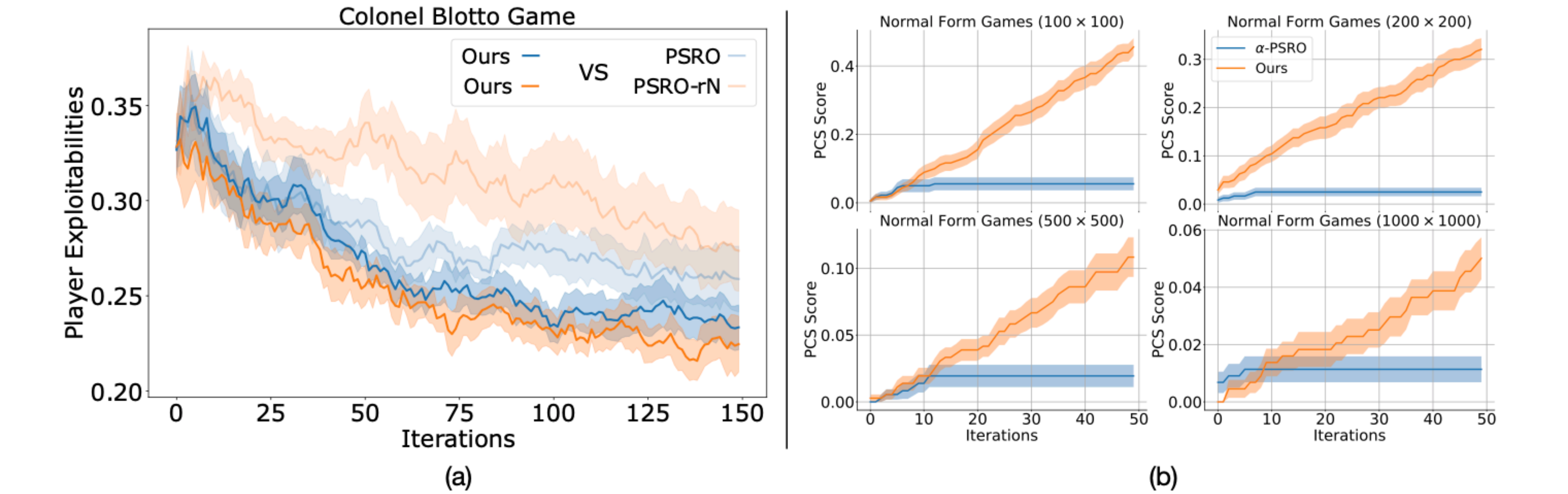
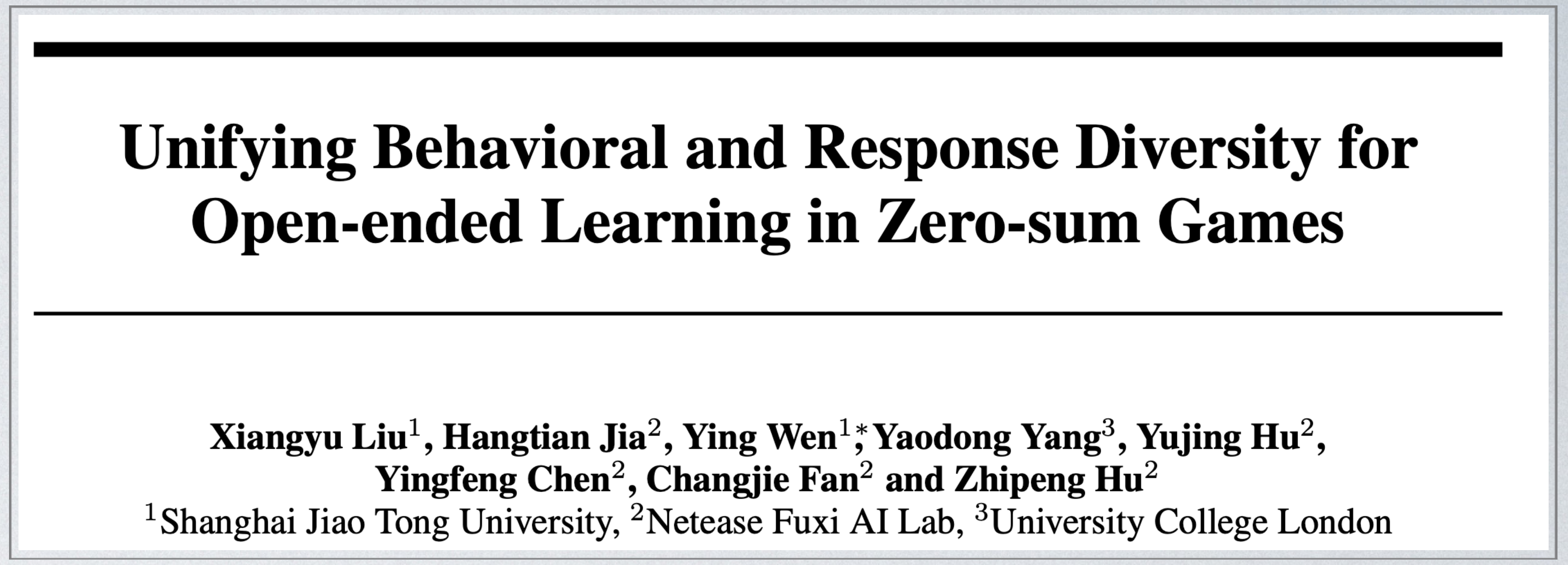


Figure 4. a) Performance of our diverse PSRO *vs.* PSRO, diverse PSRO *vs.* PSRO$_{rN}$ on the Blotto Game, b) PCS-Score comparison of our diverse α-PSRO *vs.* α-PSRO on NFGs with variable sizes.

Contents

- **Rectified Nash**
- **Diverse PSRO**
- **PSRO with Behavioural Diversity**
- **Joint PSRO**
- **Pipeline PSRO**
- **Mixed Oracles / Opponents**
- **Neural Auto-curricula**

Behavioural Diversity + Response Diversity

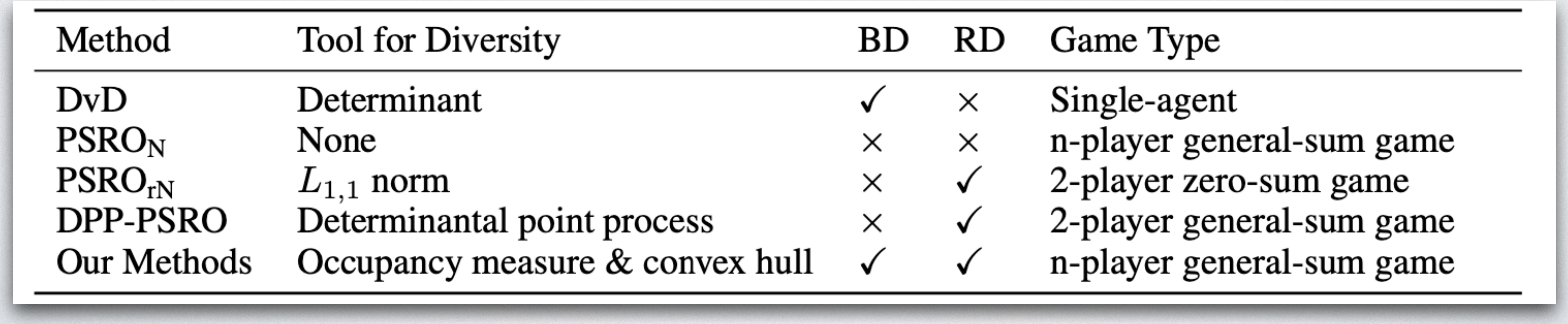


Unifying Behavioral and Response Diversity for Open-ended Learning in Zero-sum Games

Xiangyu Liu[1], Hangtian Jia[2], Ying Wen[1]*, Yaodong Yang[3], Yujing Hu[2], Yingfeng Chen[2], Changjie Fan[2] and Zhipeng Hu[2]

[1]Shanghai Jiao Tong University, [2]Netease Fuxi AI Lab, [3]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_N	None	×	×	n-player general-sum game
PSRO_rN	$L_{1,1}$ 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 π^{M+1} that has a different occupancy measure $\rho_{\boldsymbol{\pi}}(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P\left(s_t = s \mid \boldsymbol{\pi}\right)$ from π_E:

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

- One can train a neural network $f_{\hat{\theta}}$ to fit $(s, \mathbf{a}) \sim \rho_{\boldsymbol{\pi}_E}$, 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).

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

Behavioural Diversity + Response Diversity

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

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

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

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

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

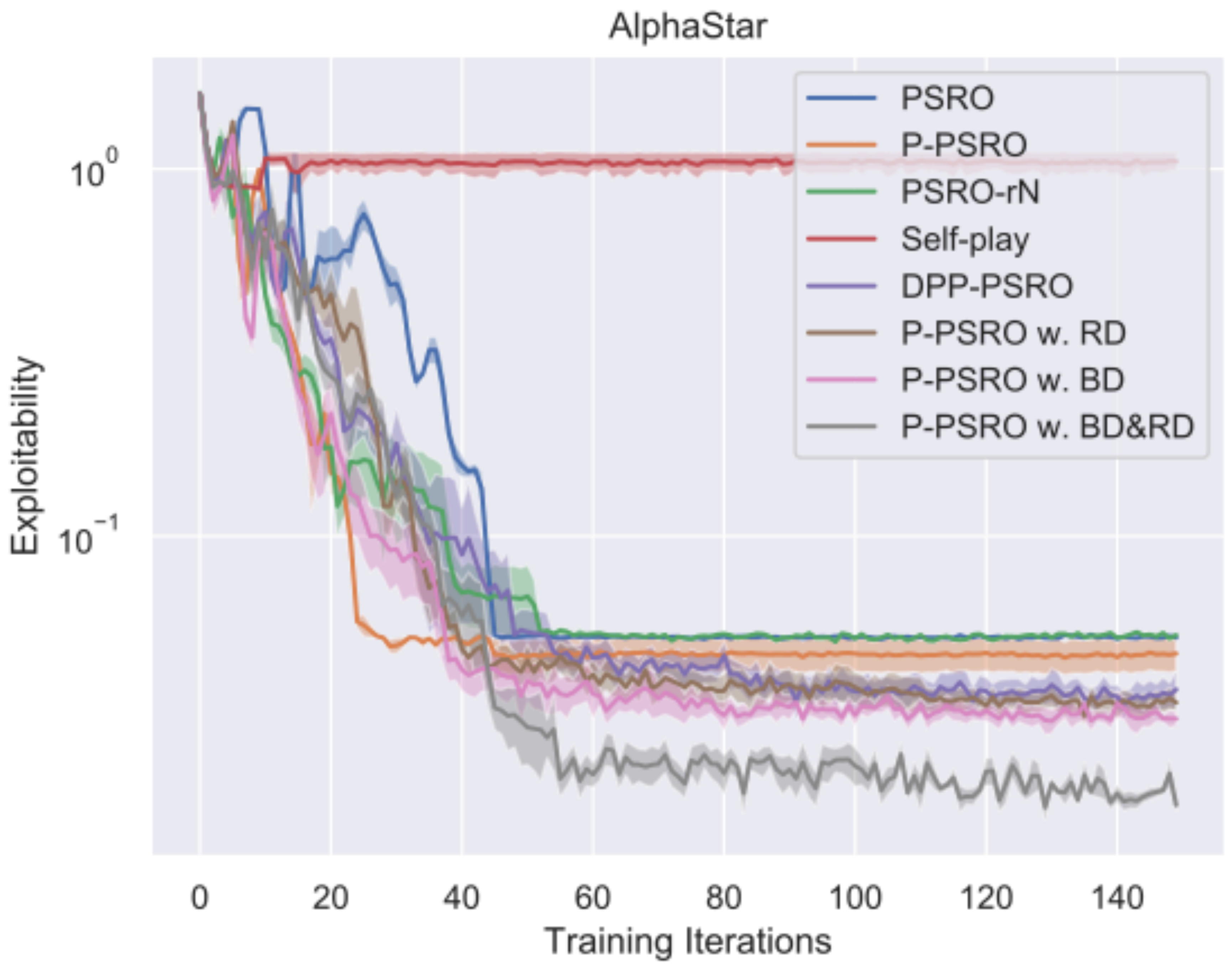
$$\frac{\partial F\left(\pi_i'(\theta)\right)}{\partial \theta} = \left(\frac{\partial \phi_i\left(\pi_i'(\theta), \pi_{-i}^1\right)}{\partial \theta}, \ldots, \frac{\partial \phi_i\left(\pi_i'(\theta), \pi_{-i}^M\right)}{\partial \theta} \right) \frac{\partial F}{\partial \mathbf{a}_{M+1}}$$

the answer: we can train against π_{-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

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



(a)

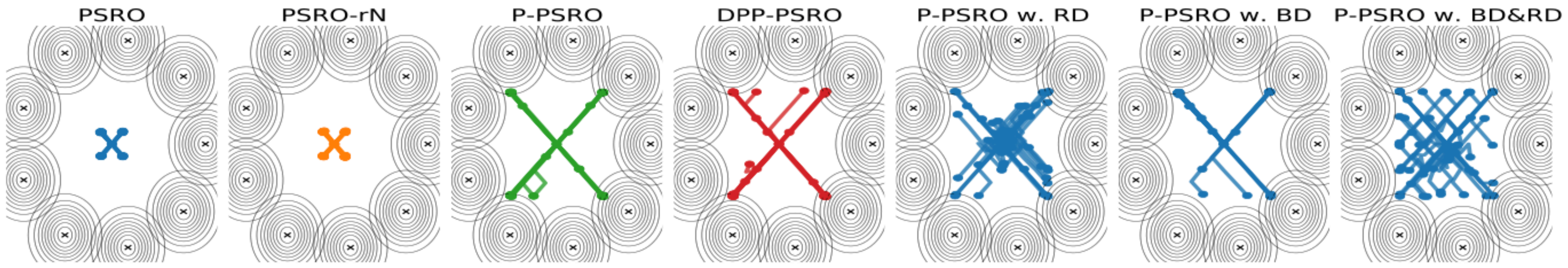


Figure 2: Exploration trajectories during training process on *Non-Transitive Mixture Games*.

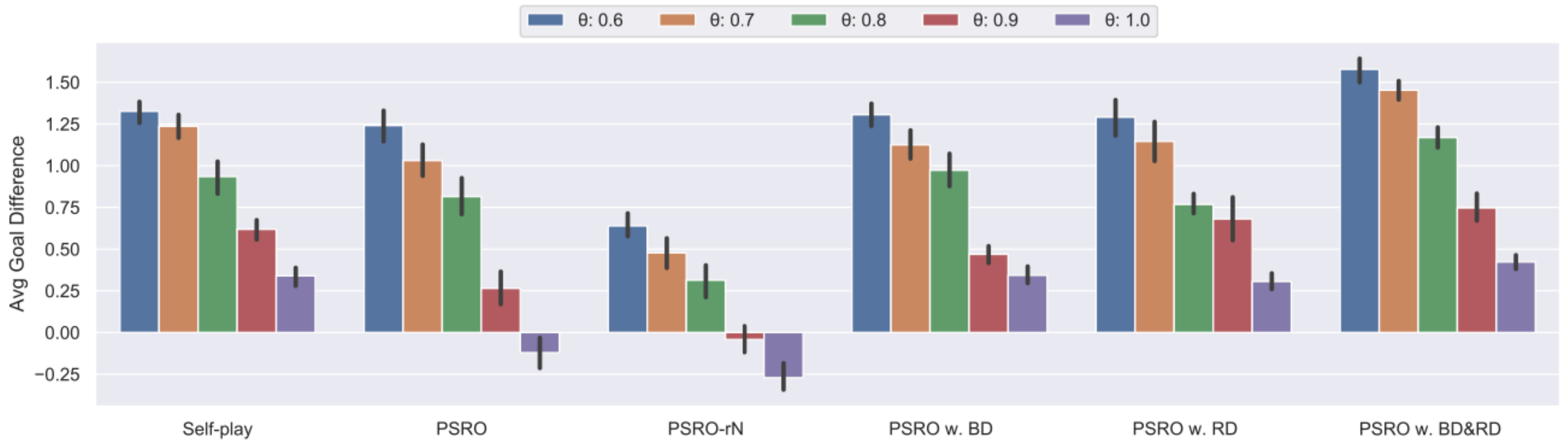


Figure 3: The average goal difference between all the methods and the built-in bots with various difficulty levels θ ($\theta \in [0, 1]$ and larger θ means harder bot) on *Google Research Football*.

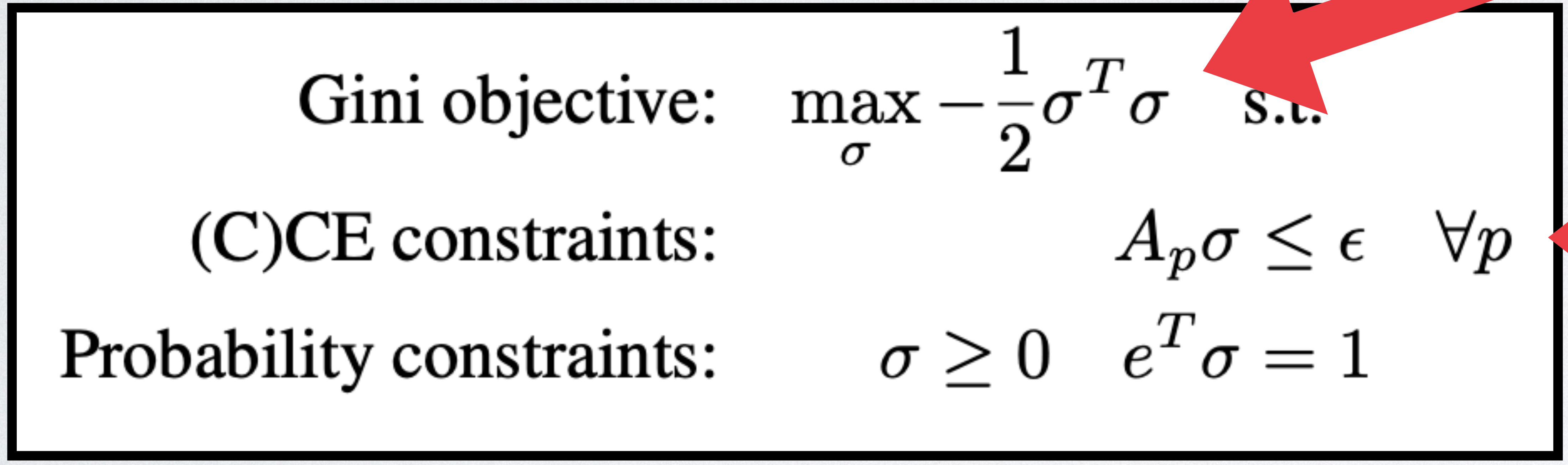
Contents

- **Rectified Nash**
- **Diverse PSRO**
- **PSRO with Behavioural Diversity**
- **Joint PSRO**
- **Pipeline PSRO**
- **Mixed Oracles / Opponents**
- **Neural Auto-curricula**

Joint PSRO [Marris et al. 2021]

- Developed for n-player general-sum extensive-form games, beyond two-player zero-sum

- Maximum Gini Correlated Equilibrium as meta-solver

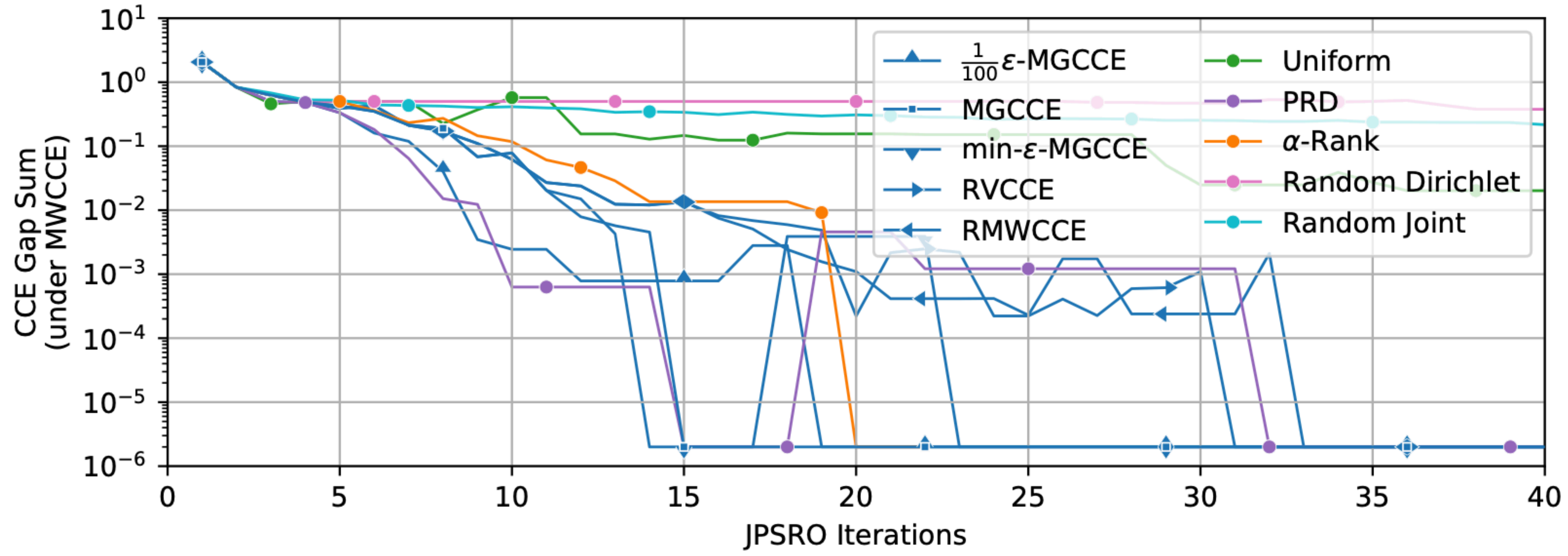


$$\begin{aligned} &\text{Gini objective:} && \max_{\sigma} -\frac{1}{2}\sigma^T\sigma \quad \text{s.t.} \\ &\text{(C)CE constraints:} && A_p\sigma \leq \epsilon \quad \forall p \\ &\text{Probability constraints:} && \sigma \geq 0 \quad e^T\sigma = 1 \end{aligned}$$

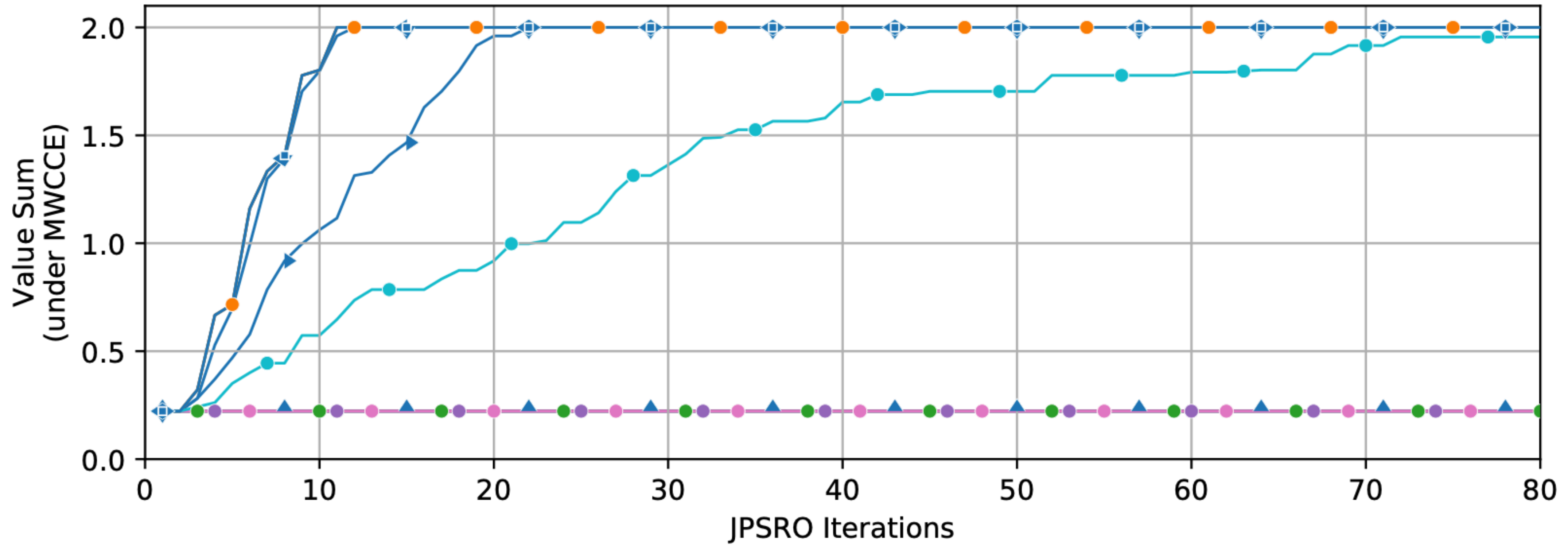
Maximised for a perfectly uniform mixed-strategy

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

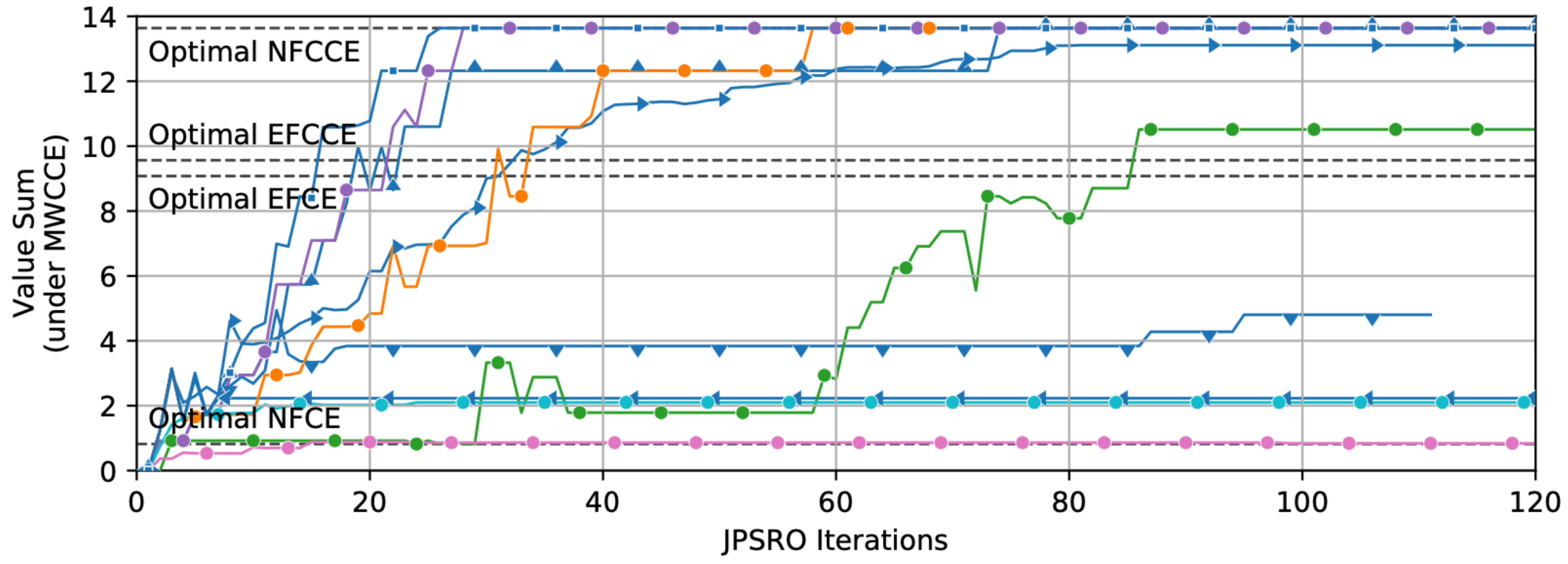
Joint PSRO [Marris et al. 2021] - Results



(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.

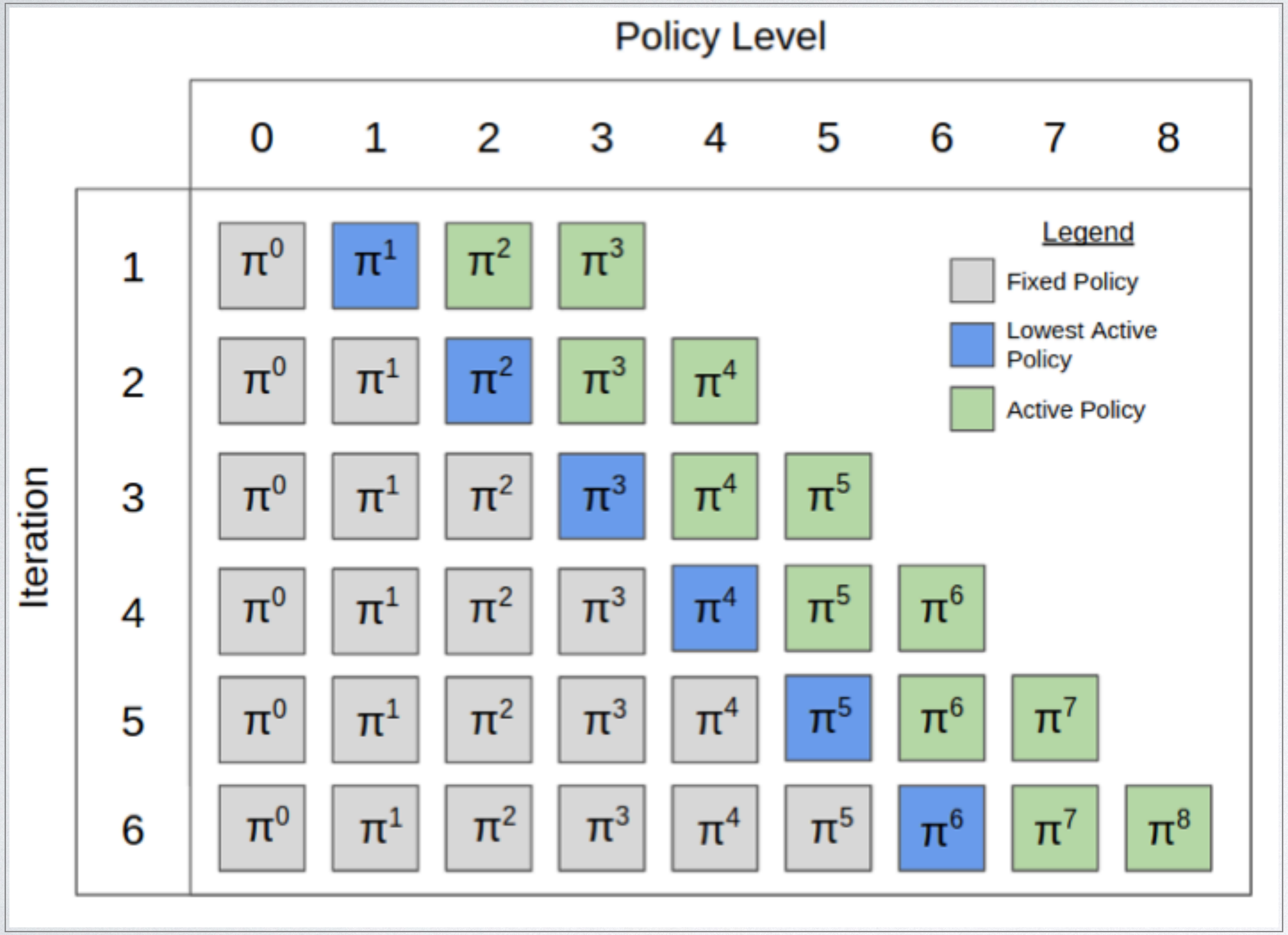


(c) Value sum on Sheriff. The optimal maximum welfare of other solution concepts are included to highlight the appeal of using NFCCE.

Contents

- **Rectified Nash**
- **Diverse PSRO**
- **PSRO with Behavioural Diversity**
- **Joint PSRO**
- **Pipeline PSRO**
- **Mixed Oracles / Opponents**
- **Neural Auto-curricula**

Pipeline PSRO [McAleer 2020] - Algorithm



Policy Level

Iteration	0	1	2	3	4	5	6	7	8
1	π^0	π^1	π^2	π^3					
2	π^0	π^1	π^2	π^3	π^4				
3	π^0	π^1	π^2	π^3	π^4	π^5			
4	π^0	π^1	π^2	π^3	π^4	π^5	π^6		
5	π^0	π^1	π^2	π^3	π^4	π^5	π^6	π^7	
6	π^0	π^1	π^2	π^3	π^4	π^5	π^6	π^7	π^8

Legend

⬜ Fixed Policy

🟦 Lowest Active Policy

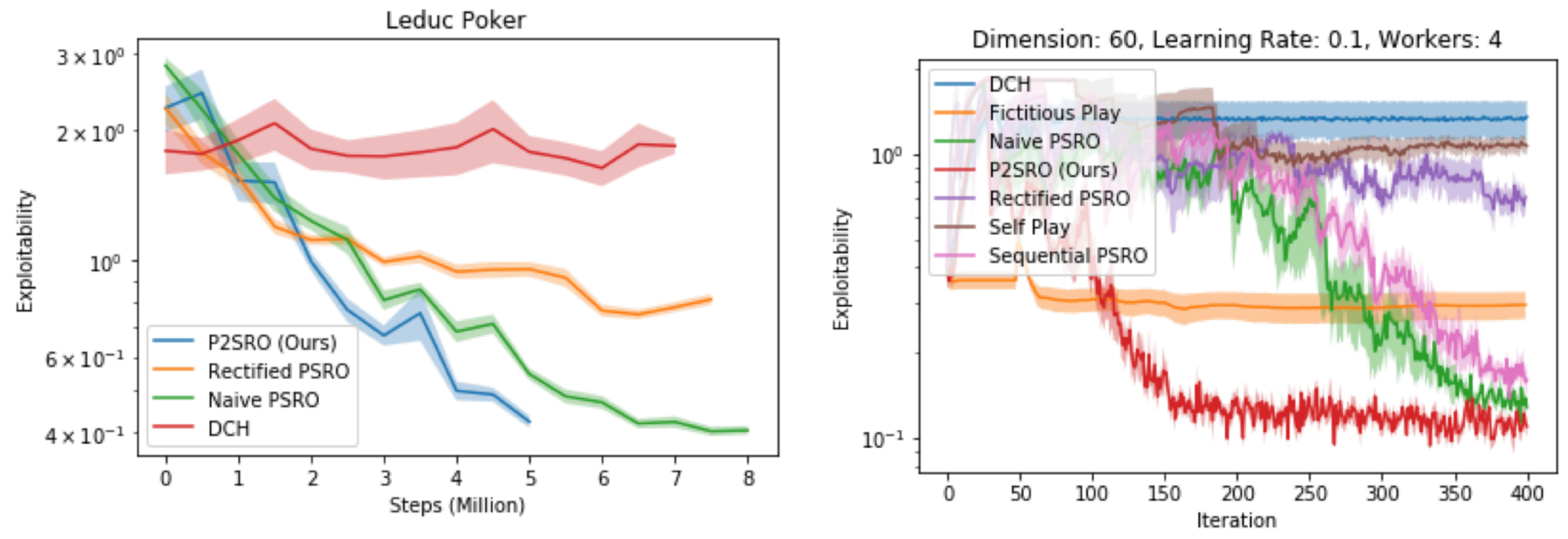
🟩 Active Policy

⬜ π^3 Fixed policies do not train anymore and remain within the fixed population.

🟦 π^3 Lowest active policy trains against the meta-distribution defined by the fixed population

🟩 π^3 Active policies train against the meta-distribution defined by the population of agents below them in the pipeline

Pipeline PSRO [McAleer 2020] - Results



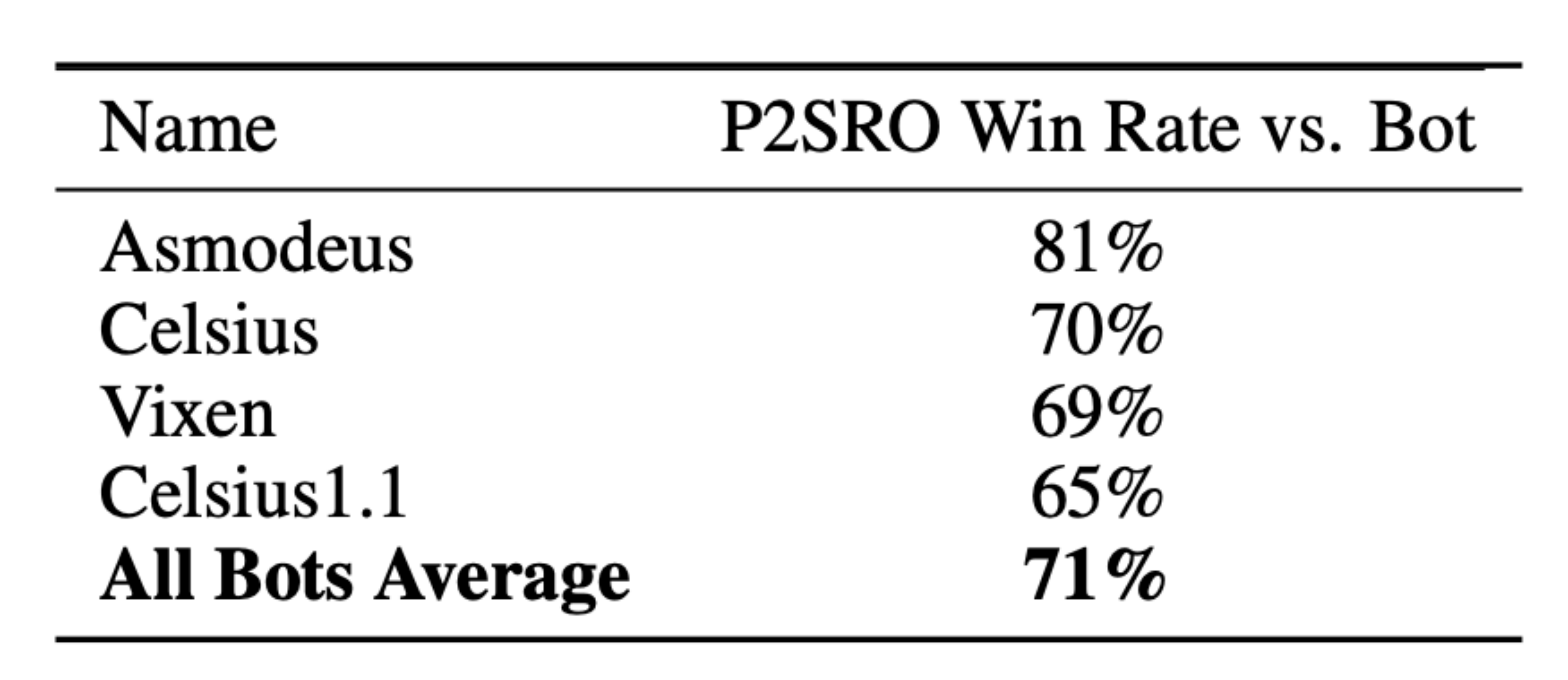
(a) Leduc poker

(b) Random Symmetric Normal Form Games

Figure 2: Exploitability of Algorithms on Leduc poker and Random Symmetric Normal Form Games

- 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

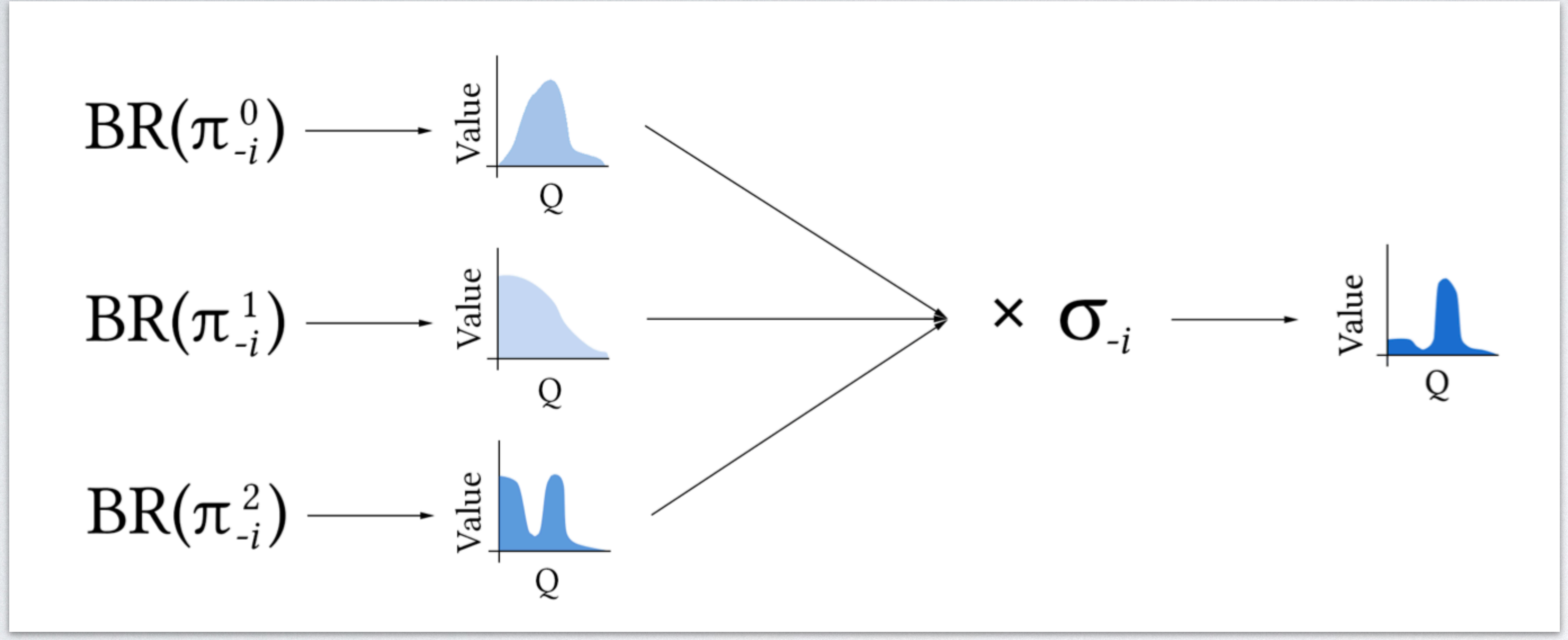


Name	P2SRO Win Rate vs. Bot
Asmodeus	81%
Celsius	70%
Vixen	69%
Celsius1.1	65%
All Bots Average	**71%**

Contents

- **Rectified Nash**
- **Diverse PSRO**
- **PSRO with Behavioural Diversity**
- **Joint PSRO**
- **Pipeline PSRO**
- **Mixed Oracles / Opponents**
- **Neural Auto-curricula**

Q-Mixing [Smith et al. 2020]


$\mathrm{BR}(\pi^0_{-i})$
$\mathrm{BR}(\pi^1_{-i})$
$\mathrm{BR}(\pi^2_{-i})$
Value
Q
Value
Q
Value
Q
$\times\ \sigma_{-i}$
Value
Q

- Learn best-responses to different policies π^t_{-i}
- Transfer knowledge against opponent mixture by weighting Q-values according to current belief of opponent's policy

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

Current brief about opponents' policy

Mixed Oracles [Smith et al. 2021]

- During PSRO how can we transfer experience across iterations?

- Now maintain two populations

$$\Pi_t = \{\pi_1, \pi_2, \ldots, \pi_t\} \qquad \Lambda_t = \{\lambda_1, \lambda_2, \ldots, \lambda_t\}$$

- Where λ_t is a learned best-response to π_t at every iteration, rather than against the meta-distribution

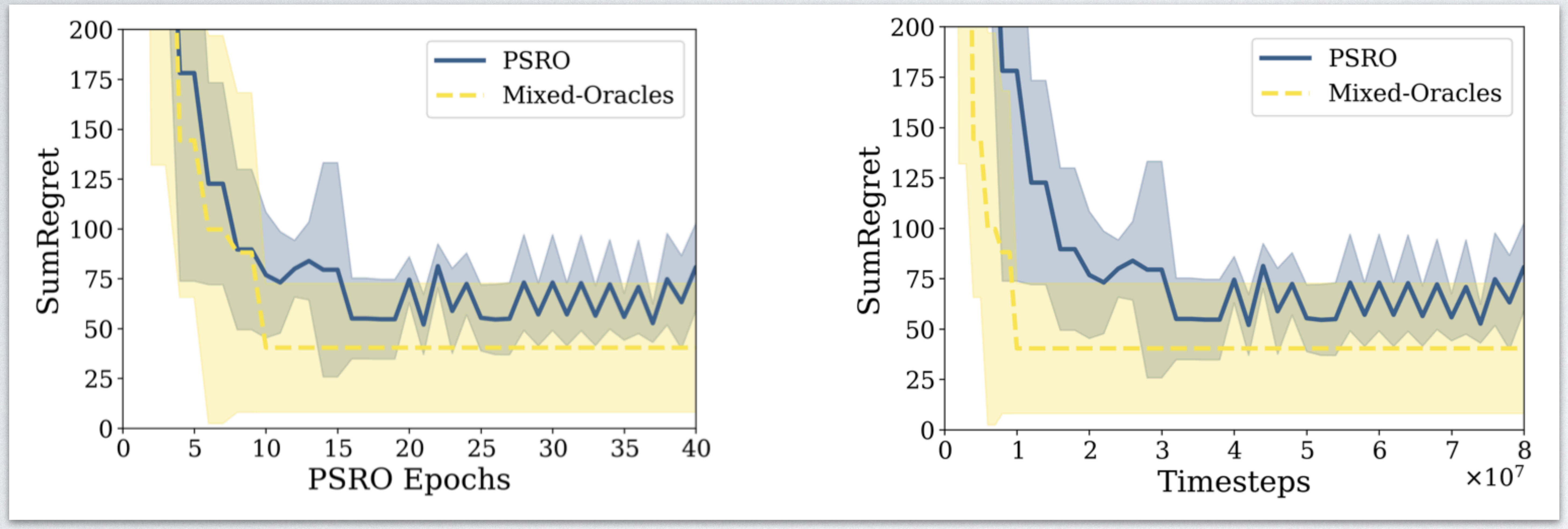
- We now have best-response experience against all policies in the population

- Use Q-mixing to find the new population policy

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

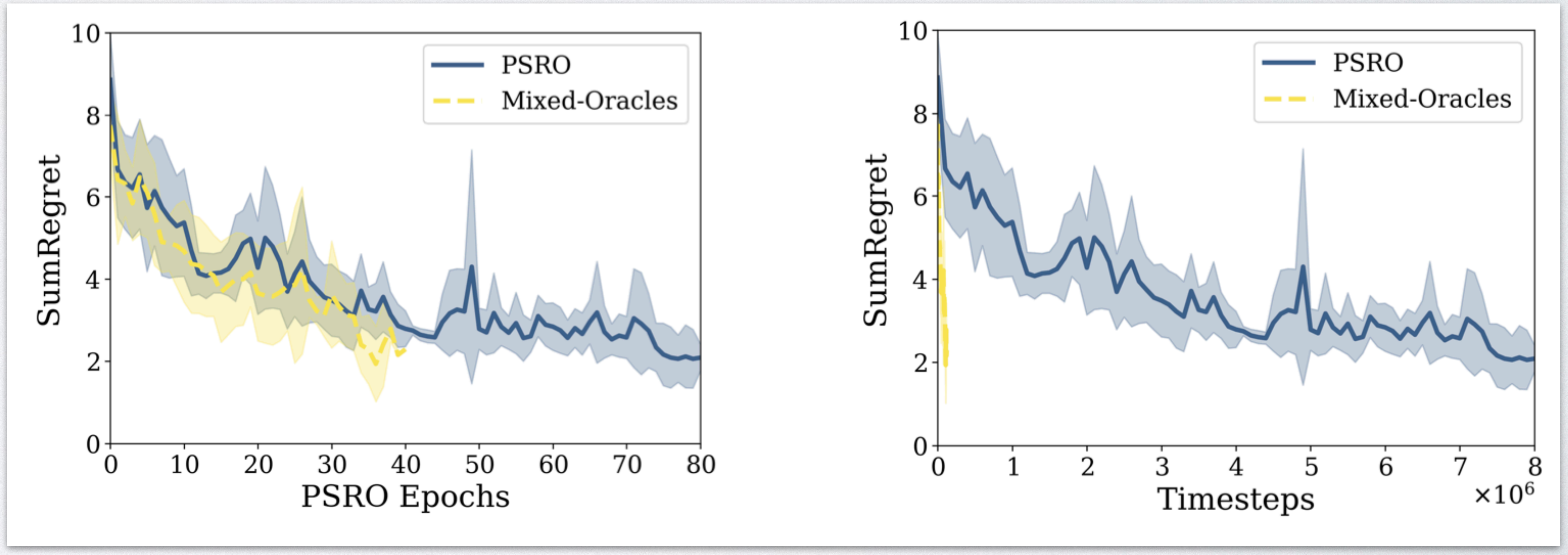
Probability of playing opponent π_{-i}

Mixed Oracles [Smith et al. 2021] - Results

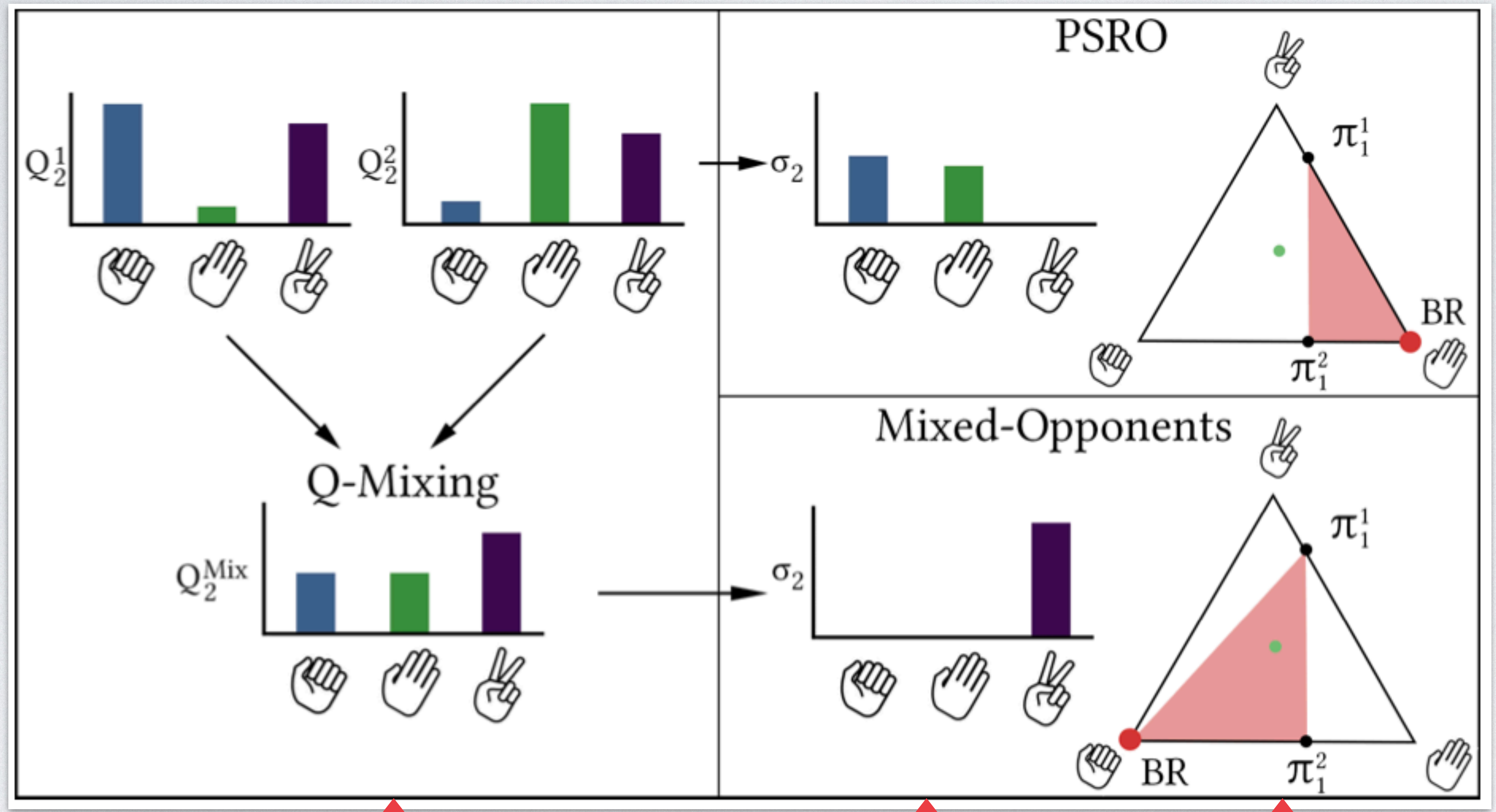


- General-Sum Tragedy of the Commons style game where individual interest is in conflict with the group interest
- Mixed-oracles converges in half the number of PSRO epochs.
- Utilises quarter the number of simulations

- 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

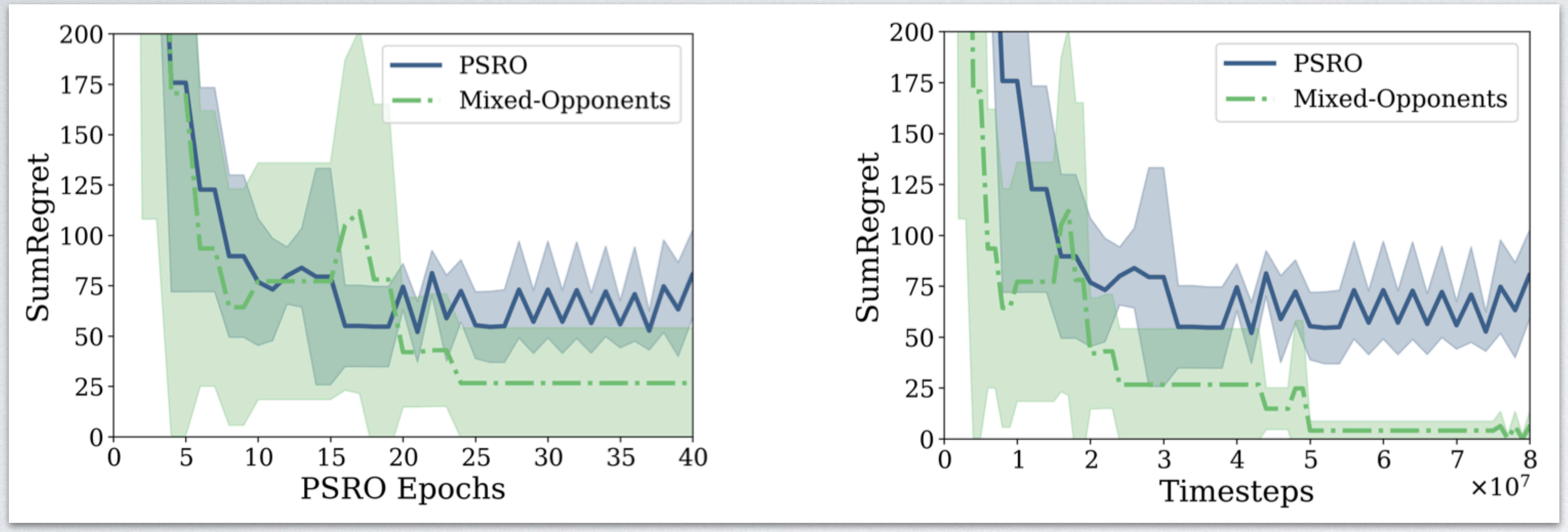
BR includes new strategy Rock

Opponent Policy Q-Values

Opponent Strategy

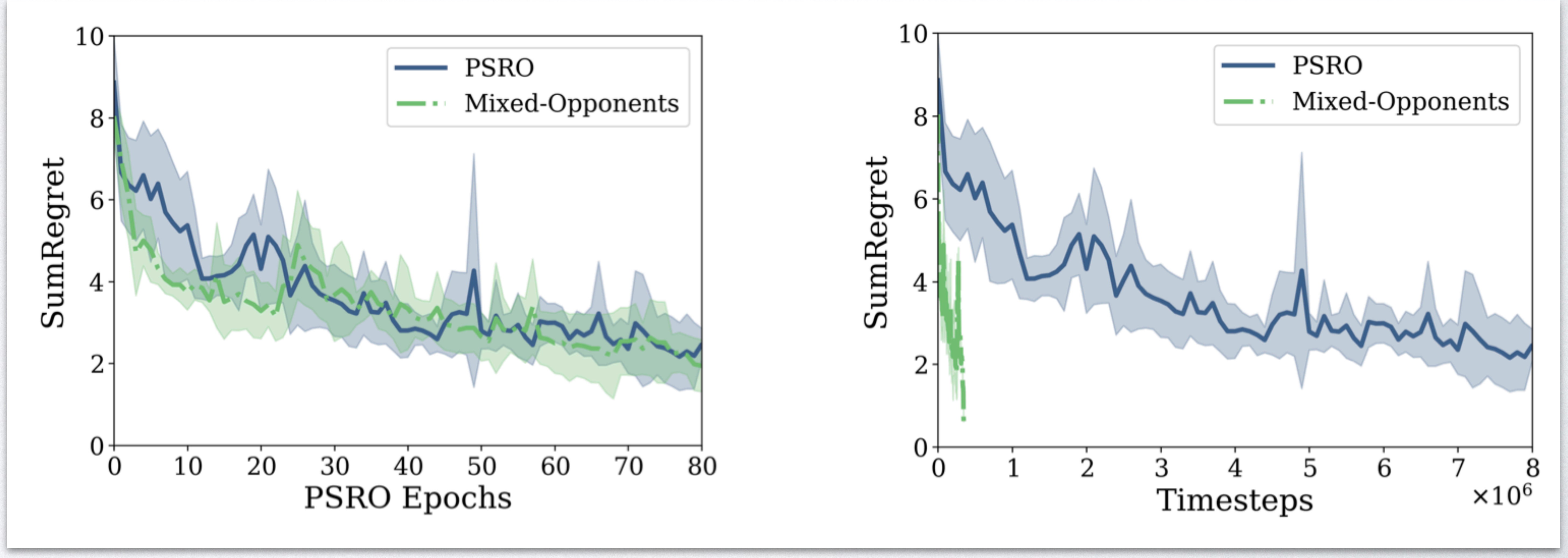
BR to Opponent Strategy

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

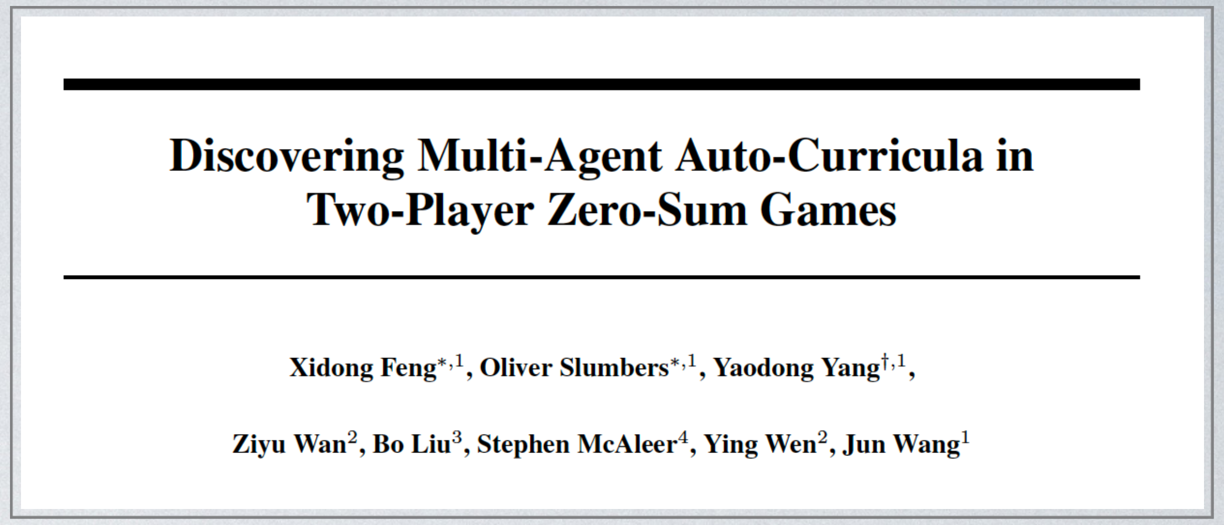
- Leduc Poker
- Mixed-oracles reaches similar performance in similar number of PSRO epochs to PSRO.
- Drastically fewer number of time steps for a comparable solution



Contents

- Rectified Nash
- Diverse PSRO
- PSRO with Behavioural Diversity
- Joint PSRO
- Pipeline PSRO
- Mixed Oracles / Opponents
- Neural Auto-curricula

Neural Auto-Curricula

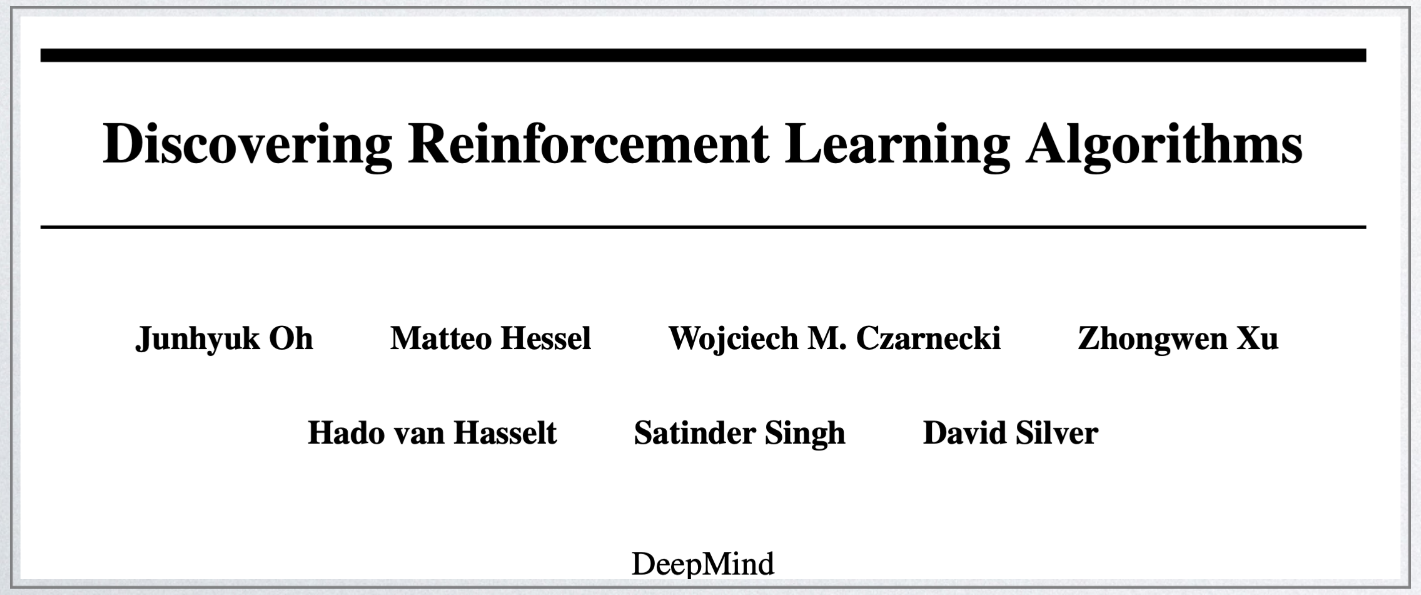


Discovering Multi-Agent Auto-Curricula in Two-Player Zero-Sum Games

Xidong Feng*,1, Oliver Slumbers*,1, Yaodong Yang†,1,

Ziyu Wan2, Bo Liu3, Stephen McAleer4, Ying Wen2, Jun Wang1

- 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

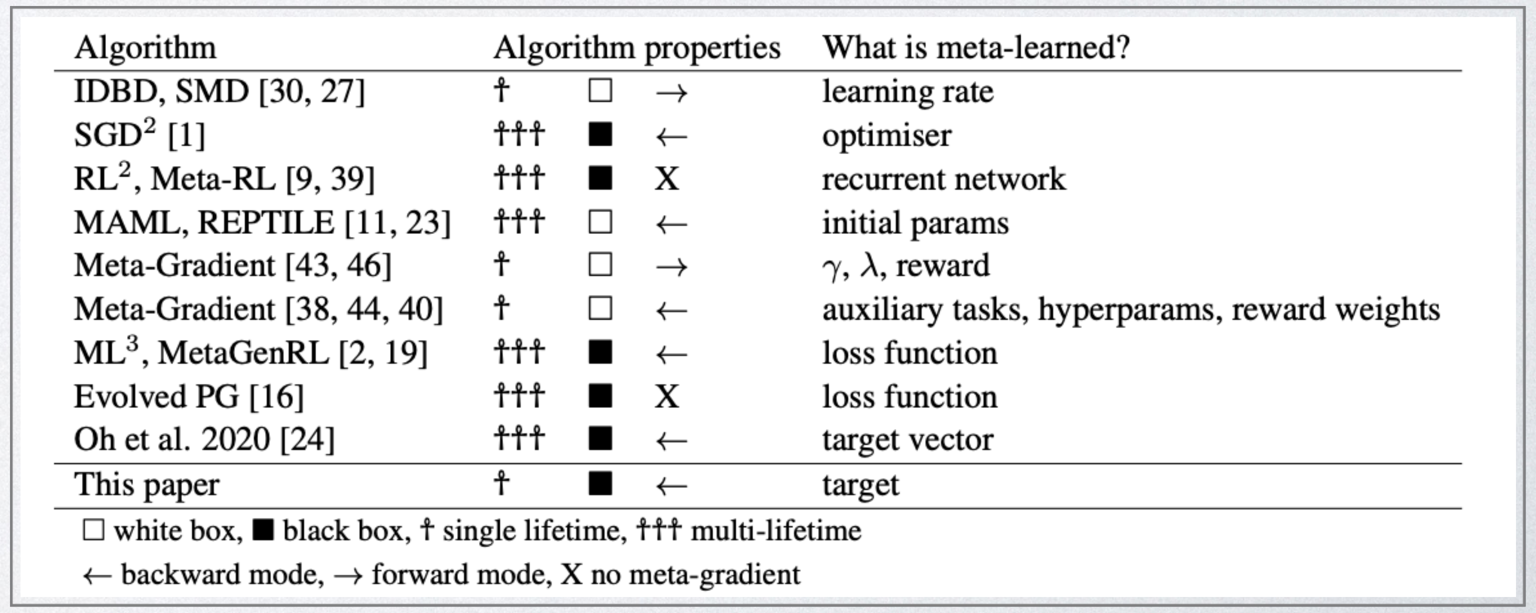
Junhyuk Oh Matteo Hessel Wojciech M. Czarnecki Zhongwen Xu

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,mtthss,junhyuk,baveja,davidsilver}@google.com

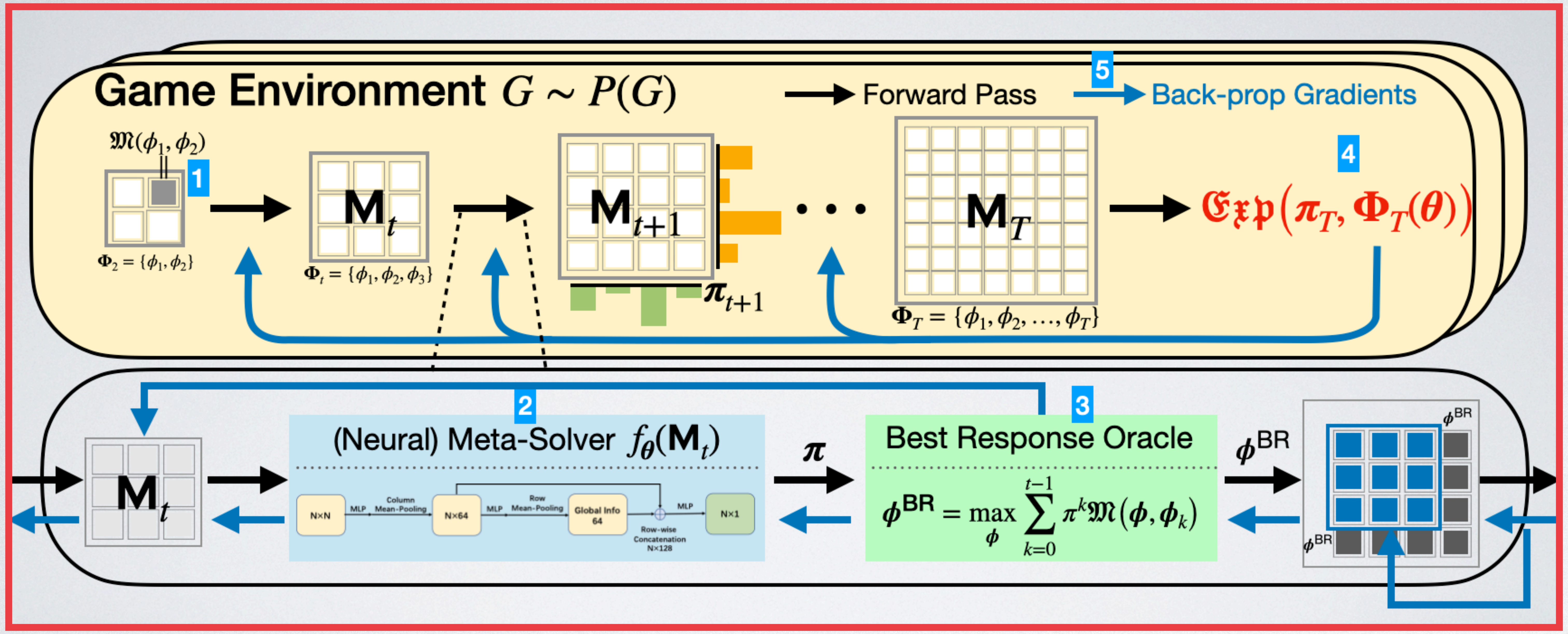


Algorithm	Algorithm properties			What is meta-learned?
IDBD, SMD [30, 27]	†	□	→	learning rate
SGD^2 [1]	†††	■	←	optimiser
RL^2, Meta-RL [9, 39]	†††	■	X	recurrent network
MAML, REPTILE [11, 23]	†††	□	←	initial params
Meta-Gradient [43, 46]	†	□	→	γ, λ, reward
Meta-Gradient [38, 44, 40]	†	□	←	auxiliary tasks, hyperparams, reward weights
ML^3, 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

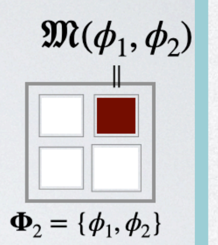
Neural Auto-Curricula Framework

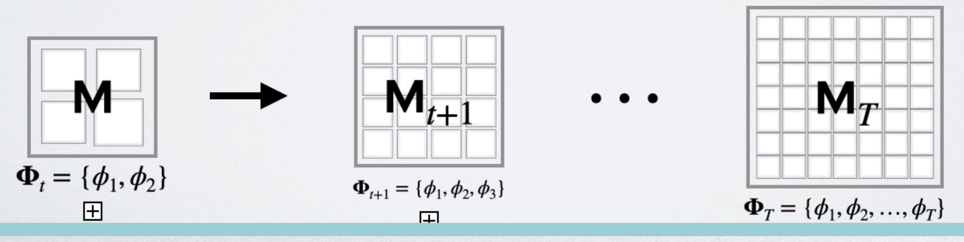


Game Environment $G \sim P(G)$
Forward Pass
5
Back-prop Gradients
$\mathfrak{M}(\phi_1, \phi_2)$
1
$\Phi_2 = \{\phi_1, \phi_2\}$
$\mathbf{M}_t$
$\Phi_t = \{\phi_1, \phi_2, \phi_3\}$
$\mathbf{M}_{t+1}$
π_{t+1}
$\mathbf{M}_T$
$\Phi_T = \{\phi_1, \phi_2, \dots, \phi_T\}$
4
$\mathfrak{Exp}(\pi_T, \Phi_T(\theta))$
2
(Neural) Meta-Solver $f_\theta(\mathbf{M}_t)$
N×N
MLP
Column Mean-Pooling
N×64
MLP
Row Mean-Pooling
Global Info 64
MLP
Row-wise Concatenation N×128
N×1
π
3
Best Response Oracle
$\phi^{BR} = \max_\phi \sum_{k=0}^{t-1} \pi^k \mathfrak{M}(\phi, \phi_k)$
ϕ^{BR}
ϕ^{BR}
ϕ^{BR}

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 $\mathfrak{M}(\phi_i, \phi_j)$
  - Payoff matrix between *agents* in a population amenable to GT analysis
  - The goal of these algorithms is to expand the populations Φ iteratively

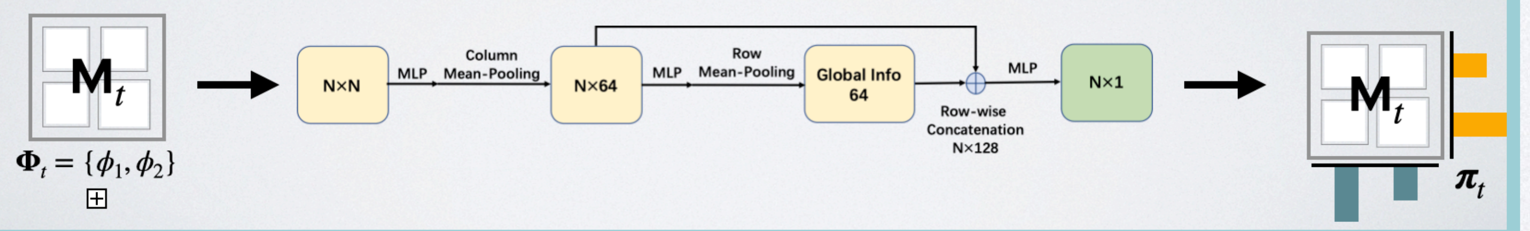




$\mathbf{M}$
$\Phi_t = \{\phi_1, \phi_2\}$
$\mathbf{M}_{t+1}$
$\Phi_{t+1} = \{\phi_1, \phi_2, \phi_3\}$
$\mathbf{M}_T$
$\Phi_T = \{\phi_1, \phi_2, \dots, \phi_T\}$

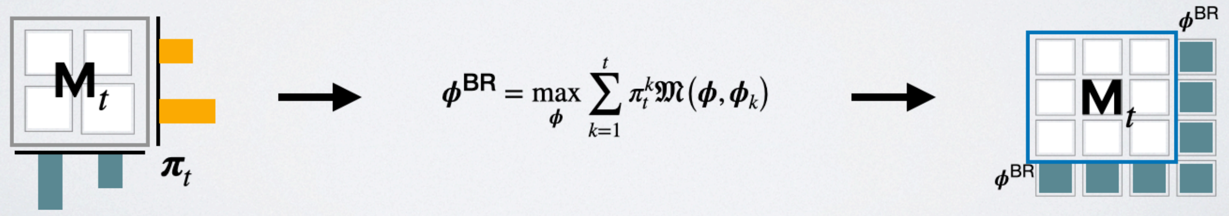
2 The Meta-Solver

- 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 θ maps $f_\theta : \mathbf{M}_t \rightarrow [0,1]^t$ so that $\pi_t = f_\theta(\mathbf{M}_t)$



3 The Best-Response Oracle

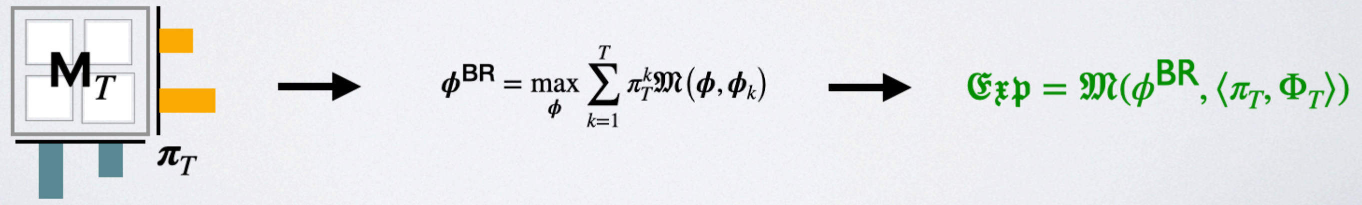
- Algorithm component that controls the iterative expansion of the population
  - Given a curriculum $\pi_t \in \Delta_{|\Phi_t|}$ the goal becomes to solve a best-response to this distribution
  - Goal is the following: $\phi_t^{\text{BR}} = \text{argmax}_\phi \sum_{k=1}^{t} \pi_t^k \mathfrak{M}(\phi,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



$\mathbf{M}_t$
π_t
$\phi^{\text{BR}} = \max_\phi \sum_{k=1}^{t} \pi_t^k \mathfrak{M}(\phi, \phi_k)$
$\mathbf{M}_t$
ϕ^{BR}
ϕ^{BR}

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 Φ_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



$\mathbf{M}_T$
π_T
$\phi^{\text{BR}} = \max_\phi \sum_{k=1}^{T} \pi_t^k \mathfrak{M}(\phi, \phi_k)$
$\mathfrak{Exp} = \mathfrak{M}(\phi^{\text{BR}}, \langle \pi_T, \Phi_T \rangle)$

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:

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

  - What does the gradient boil down to then?

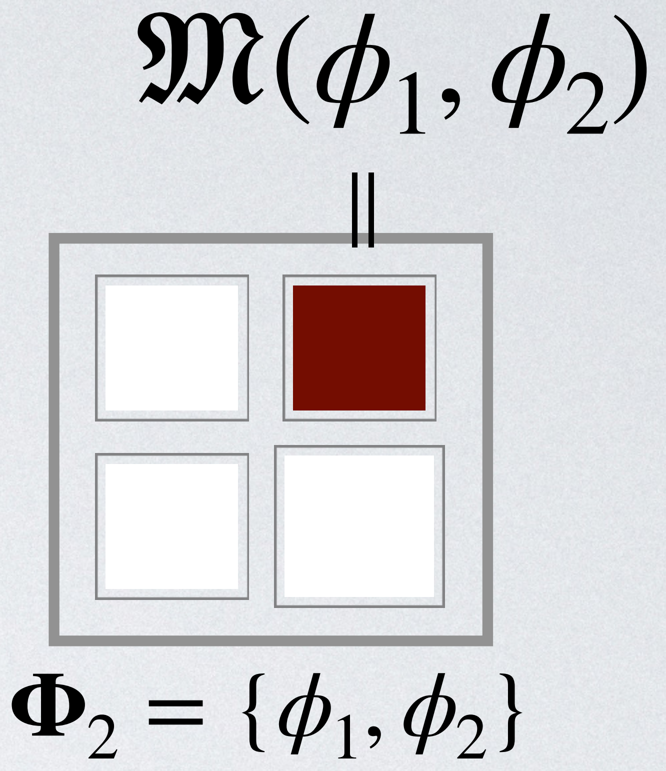
$$\nabla_\theta J(\theta) = \mathbb{E}_G \Big[\frac{\partial \mathfrak{M}_{T+1}}{\partial \phi_{T+1}^{\text{BR}}} \boxed{\frac{\partial \phi_{T+1}^{\text{BR}}}{\partial \theta}} + \frac{\partial \mathfrak{M}_{T+1}}{\partial \pi_T} \frac{\partial \pi_T}{\partial \theta} + \frac{\partial \mathfrak{M}_{T+1}}{\partial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

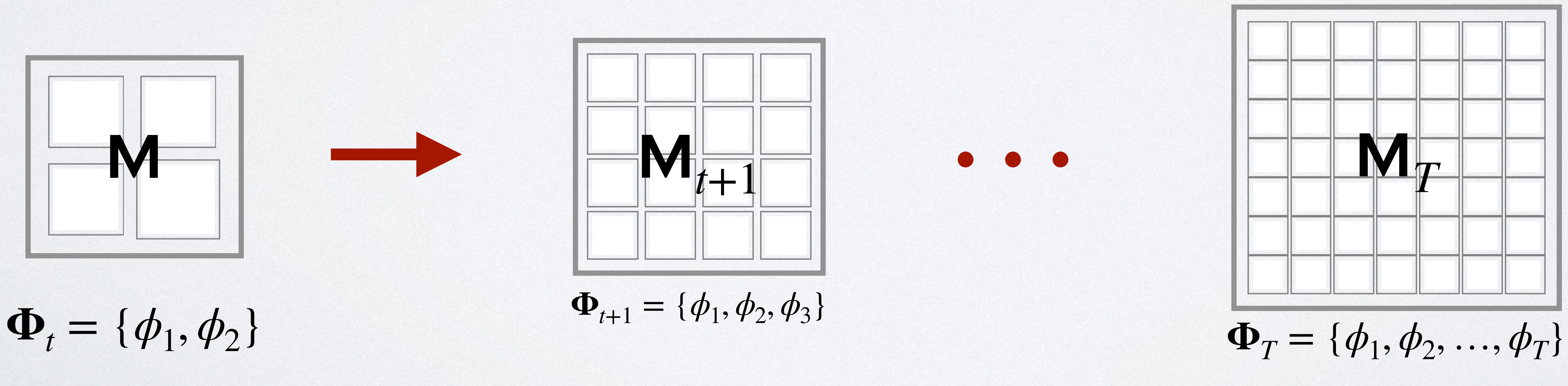
Gradient of most interest decomposes to

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

1 The Meta-Game

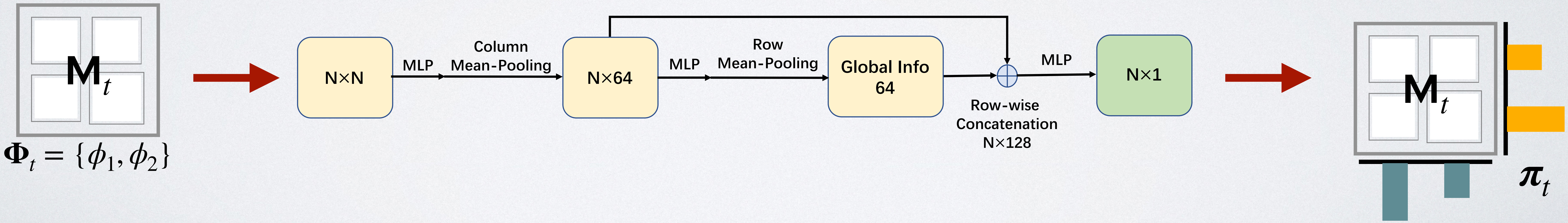
- Main component of population-based methods - **The meta-game**
  - An agent is a mapping $\phi : \boldsymbol{S} \times \boldsymbol{A} \rightarrow [0,1]$
  - The payoff for agent *i* vs. agent *j* is defined as $\mathfrak{M}(\phi_i, \phi_j)$
  - Payoff matrix between *agents* in a population amenable to GT analysis
  - The goal of these algorithms is to expand the populations Φ iteratively




$\mathbf{M}$
$\Phi_t = \{\phi_1, \phi_2\}$
$\mathbf{M}_{t+1}$
$\Phi_{t+1} = \{\phi_1, \phi_2, \phi_3\}$
$\mathbf{M}_T$
$\Phi_T = \{\phi_1, \phi_2, \ldots, \phi_T\}$

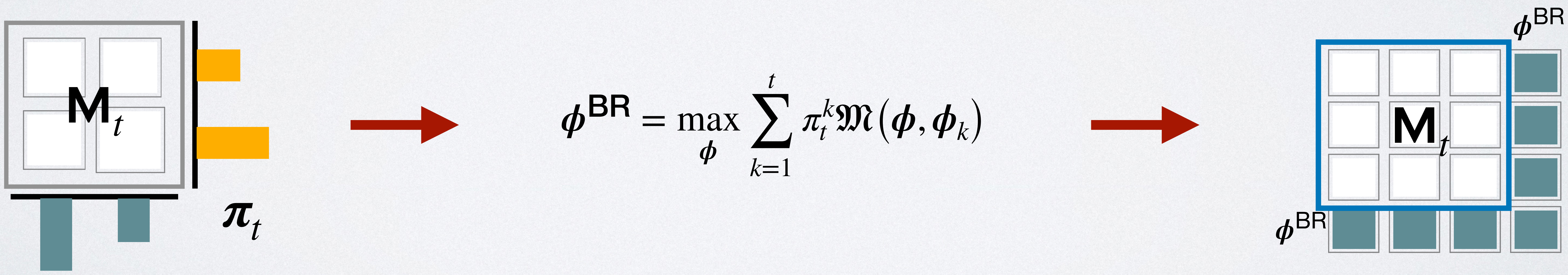
2 The Meta-Solver

- 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 θ 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: $\phi_t^{\text{BR}} = \text{argmax}_\phi \sum_{k=1}^{t} \pi_t^k \mathfrak{M}(\phi, \phi_k)$
    - Perform the optimisation in anyway desired, but this will impact the meta-gradient calculation

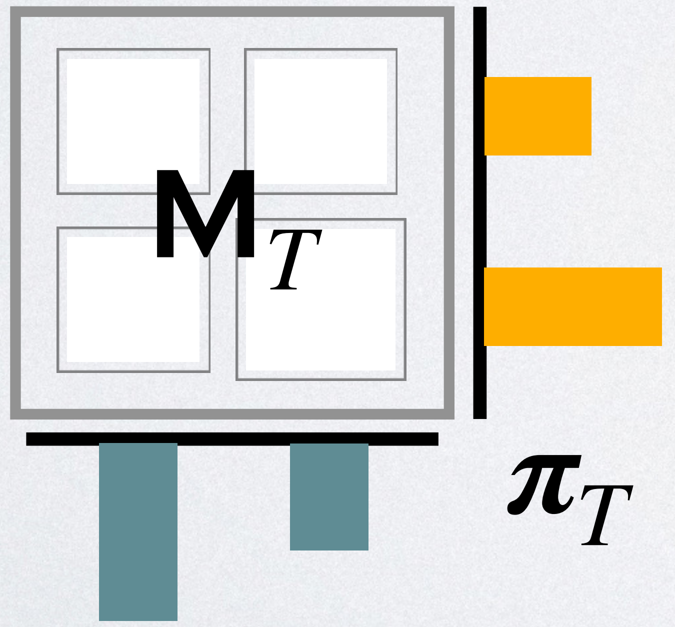


$$\boldsymbol{\phi}^{\text{BR}} = \max_{\boldsymbol{\phi}} \sum_{k=1}^{t} \pi_t^k \mathfrak{M}(\boldsymbol{\phi}, \boldsymbol{\phi}_k)$$

$\mathbf{M}_t$
$\boldsymbol{\pi}_t$
$\boldsymbol{\phi}^{\text{BR}}$
$\mathbf{M}_t$
$\boldsymbol{\phi}^{\text{BR}}$

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 Φ_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



$$\xrightarrow{\quad} \quad \phi^{\mathrm{BR}} = \max_{\phi} \sum_{k=1}^{T} \pi_T^k \mathfrak{M}(\phi, \phi_k) \quad \xrightarrow{\quad} \quad \mathfrak{Exp} = \mathfrak{M}(\phi^{\mathrm{BR}}, \langle \pi_T, \Phi_T \rangle)$$

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:

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

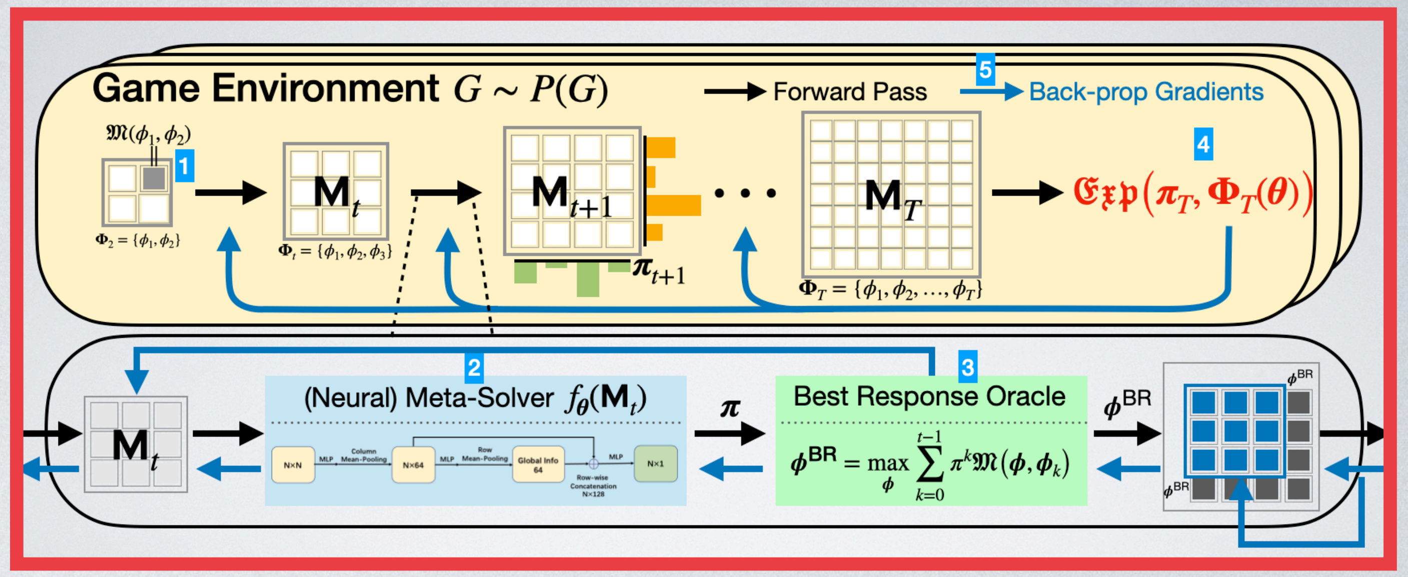
  - What does the gradient boil down to then?

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

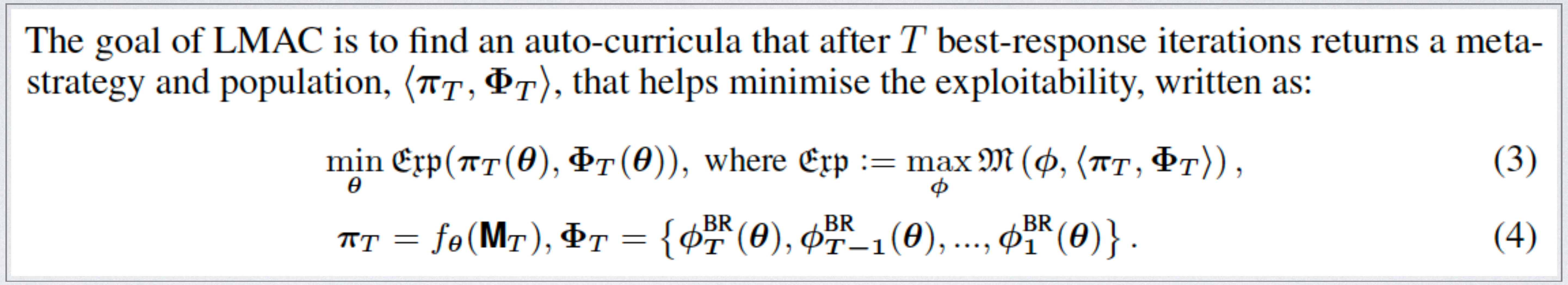
Gradient of most interest decomposes to

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

Neural Auto-Curricula Recap


Game Environment $G \sim P(G)$
Forward Pass
5
Back-prop Gradients
$\mathfrak{M}(\phi_1, \phi_2)$
1
$\Phi_2 = \{\phi_1, \phi_2\}$
$\mathbf{M}_t$
$\Phi_t = \{\phi_1, \phi_2, \phi_3\}$
$\mathbf{M}_{t+1}$
π_{t+1}
$\mathbf{M}_T$
$\Phi_T = \{\phi_1, \phi_2, \ldots, \phi_T\}$
4
$\mathfrak{Exp}(\pi_T, \Phi_T(\theta))$
2
(Neural) Meta-Solver $f_\theta(\mathbf{M}_t)$
N×N
MLP
Column Mean-Pooling
N×64
MLP
Row Mean-Pooling
Global Info 64
MLP
N×1
Row-wise Concatenation N×128
π
3
Best Response Oracle
$\phi^{\text{BR}} = \max_{\phi} \sum_{k=0}^{t-1} \pi^k \mathfrak{M}(\phi, \phi_k)$
ϕ^{BR}

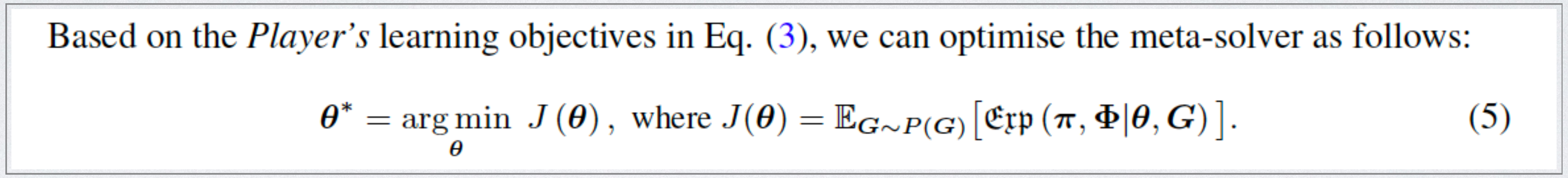
- The objective is given by:



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

$$\min_{\boldsymbol{\theta}} \mathfrak{Exp}(\boldsymbol{\pi}_T(\boldsymbol{\theta}), \boldsymbol{\Phi}_T(\boldsymbol{\theta})), \text{ where } \mathfrak{Exp} := \max_{\boldsymbol{\phi}} \mathfrak{M}\left(\boldsymbol{\phi}, \langle \boldsymbol{\pi}_T, \boldsymbol{\Phi}_T \rangle\right), \tag{3}$$

$$\boldsymbol{\pi}_T = f_{\boldsymbol{\theta}}(\mathbf{M}_T), \boldsymbol{\Phi}_T = \left\{\boldsymbol{\phi}_T^{\text{BR}}(\boldsymbol{\theta}), \boldsymbol{\phi}_{T-1}^{\text{BR}}(\boldsymbol{\theta}), \ldots, \boldsymbol{\phi}_1^{\text{BR}}(\boldsymbol{\theta})\right\}. \tag{4}$$



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

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \text{ where } J(\boldsymbol{\theta}) = \mathbb{E}_{G \sim P(G)}\left[\mathfrak{Exp}(\boldsymbol{\pi}, \boldsymbol{\Phi} | \boldsymbol{\theta}, G)\right]. \tag{5}$$

- When optimising the meta-solver θ, the type of best-response oracle matters due to back-propagation!
  - one-step gradient descent oracle

$$\phi_{t+1}^{\text{BR}} = \phi_0 + \alpha \frac{\partial \mathfrak{M}(\phi_0, \langle \pi_t, \Phi_t \rangle)}{\partial \phi_0}, \ \frac{\partial \boldsymbol{\phi}_{t+1}^{\text{BR}}}{\partial \pi_t} = \alpha \frac{\partial^2 \mathfrak{M}\left(\boldsymbol{\phi}_0, \langle \pi_t, \Phi_t \rangle\right)}{\partial \boldsymbol{\phi}_0 \partial \pi_t}, \ \frac{\partial \boldsymbol{\phi}_{t+1}^{\text{BR}}}{\partial \Phi_t} = \alpha \frac{\partial^2 \mathfrak{M}\left(\boldsymbol{\phi}_0, \langle \pi_t, \Phi_t \rangle\right)}{\partial \boldsymbol{\phi}_0 \partial \Phi_t}.$$

  - N-step gradient descent oracle (via implicit gradient)

$$\frac{\partial \boldsymbol{\phi}_{t+1}^{\text{BR}}}{\partial \Phi_t} = -\left[\frac{\partial^2 \mathfrak{M}(\boldsymbol{\phi}_{t+1}^{\text{BR}}, \langle \pi_t, \boldsymbol{\Phi}_t \rangle)}{\partial \boldsymbol{\phi}_{t+1}^{\text{BR}} \partial \boldsymbol{\phi}_{t+1}^{\text{BR}^T}}\right]^{-1} \frac{\partial^2 \mathfrak{M}(\boldsymbol{\phi}_{t+1}^{\text{BR}}, \langle \pi_t, \boldsymbol{\Phi}_t \rangle)}{\partial \boldsymbol{\phi}_{t+1}^{\text{BR}} \partial \boldsymbol{\Phi}_t}$$

  - policy-gradient based oracle (via DICE)

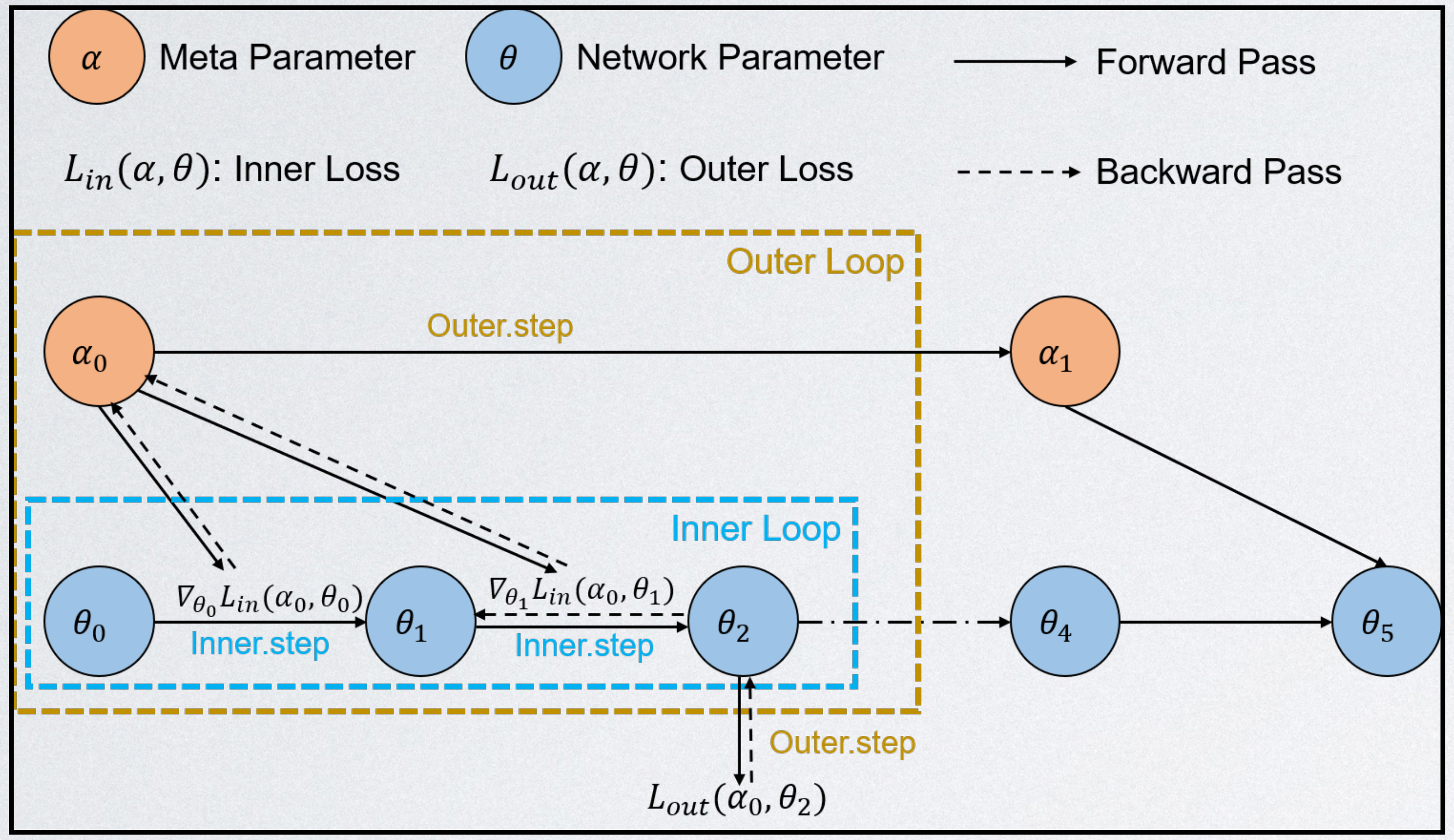
$$\boldsymbol{\phi}_1 = \boldsymbol{\phi}_0 + \alpha \frac{\partial \mathcal{J}^{\text{DICE}}}{\partial \boldsymbol{\phi}_0}, \text{ where } \mathcal{J}^{\text{DICE}} = \sum_{k=0}^{H-1} \left(\prod_{k'=0}^{k} \frac{\pi_{\boldsymbol{\phi}_1}\left(a_{k'}^1 \mid s_{k'}^1\right) \pi_{\boldsymbol{\phi}_2}\left(a_{k'}^2 \mid s_{k'}^2\right)}{\perp\left(\pi_{\boldsymbol{\phi}_1}\left(a_{k'}^1 \mid s_{k'}^1\right) \pi_{\boldsymbol{\phi}_2}\left(a_{k'}^2 \mid s_{k'}^2\right)\right)} \right) r_k^1$$

  - general type of oracle (via ES)

$$\nabla_\theta \hat{J}_\sigma(\theta) = \mathbb{E}_{G \sim P(G), \epsilon \sim \mathcal{N}(0,I)}\left[\frac{1}{\sigma}\left(\mathfrak{Exp}_T(\pi_T, \Phi_T) \Big| \theta + \epsilon, G\right)\epsilon\right]$$

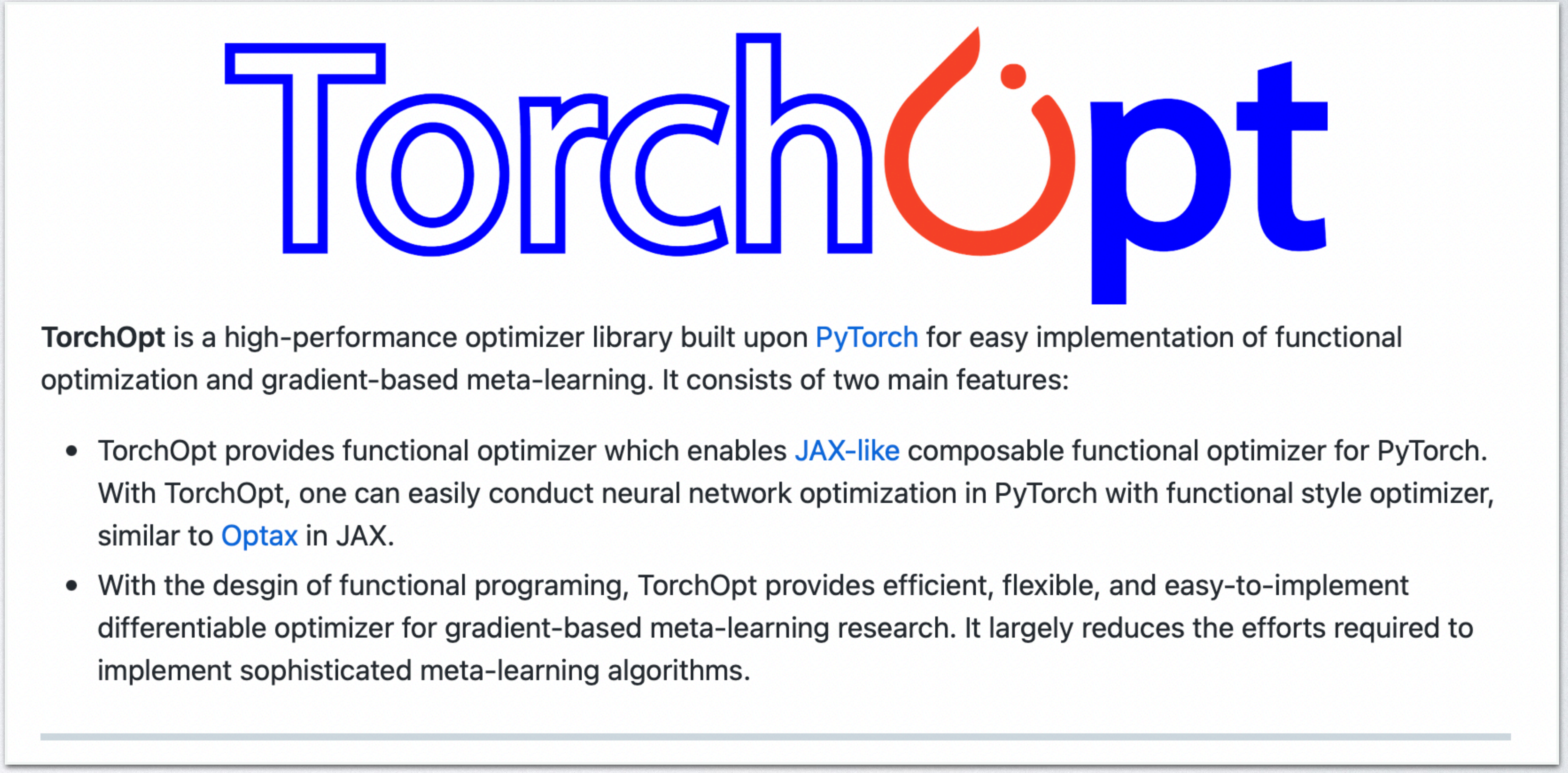
The TorchOpt Project

- Computing meta-gradient in meta-RL is troublesome, we offer a JAX-like functional programming tool for PyTroch.



α Meta Parameter
θ Network Parameter
Forward Pass
$L_{in}(\alpha,\theta)$: Inner Loss
$L_{out}(\alpha,\theta)$: Outer Loss
Backward Pass
Outer Loop
Outer.step
α_0
α_1
Inner Loop
θ_0
$\nabla_{\theta_0} L_{in}(\alpha_0,\theta_0)$
Inner.step
θ_1
$\nabla_{\theta_1} L_{in}(\alpha_0,\theta_1)$
Inner.step
θ_2
θ_4
θ_5
Outer.step
$L_{out}(\alpha_0,\theta_2)$

https://github.com/metaopt/TorchOpt



TorchOpt

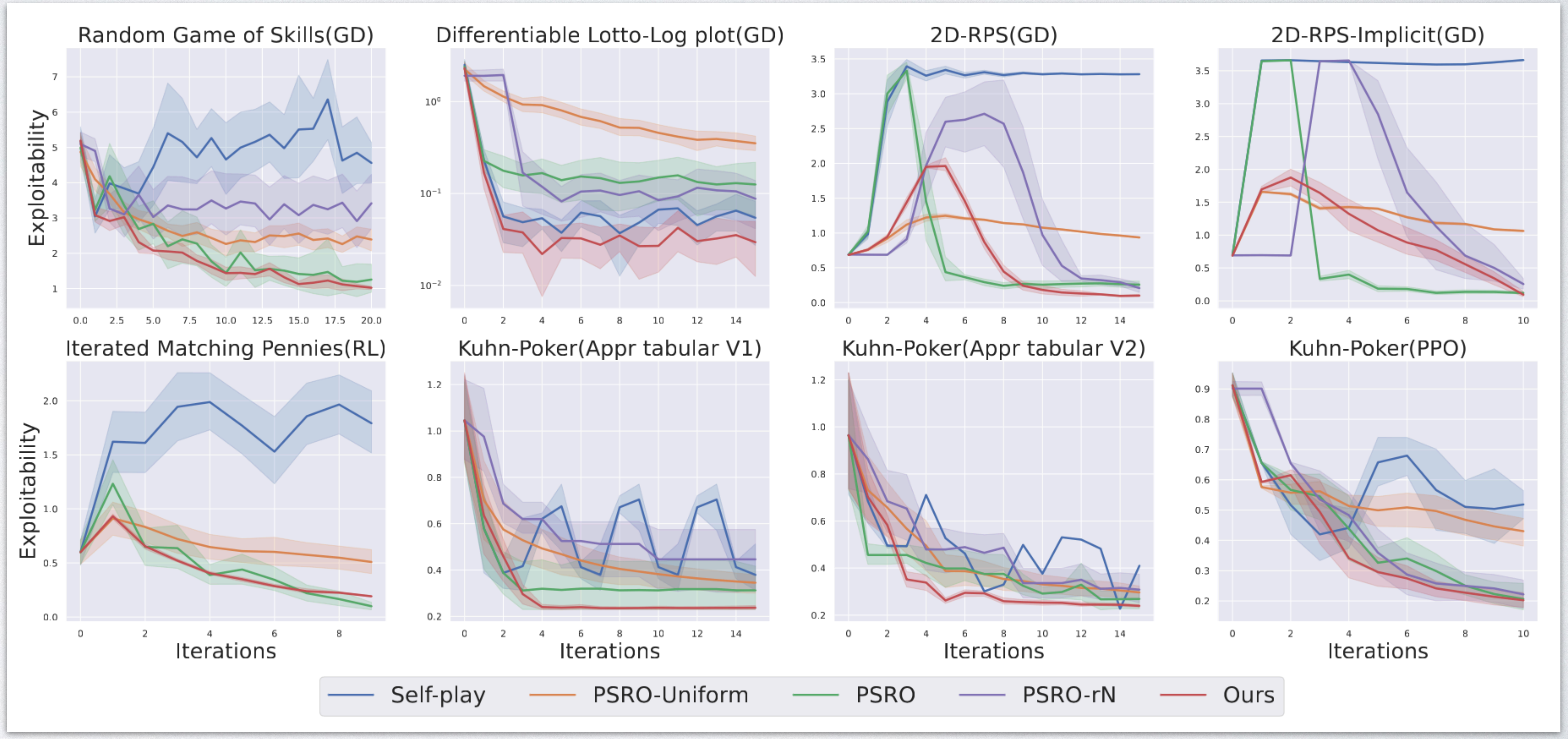
TorchOpt is a high-performance optimizer library built upon PyTorch for easy implementation of functional optimization and gradient-based meta-learning. It consists of two main features:

- TorchOpt provides functional optimizer which enables JAX-like composable functional optimizer for PyTorch. With TorchOpt, one can easily conduct neural network optimization in PyTorch with functional style optimizer, similar to Optax in JAX.
- With the desgin of functional programing, TorchOpt provides efficient, flexible, and easy-to-implement differentiable optimizer for gradient-based meta-learning research. It largely reduces the efforts required to implement sophisticated meta-learning algorithms.

Neural Auto-Curricula Results

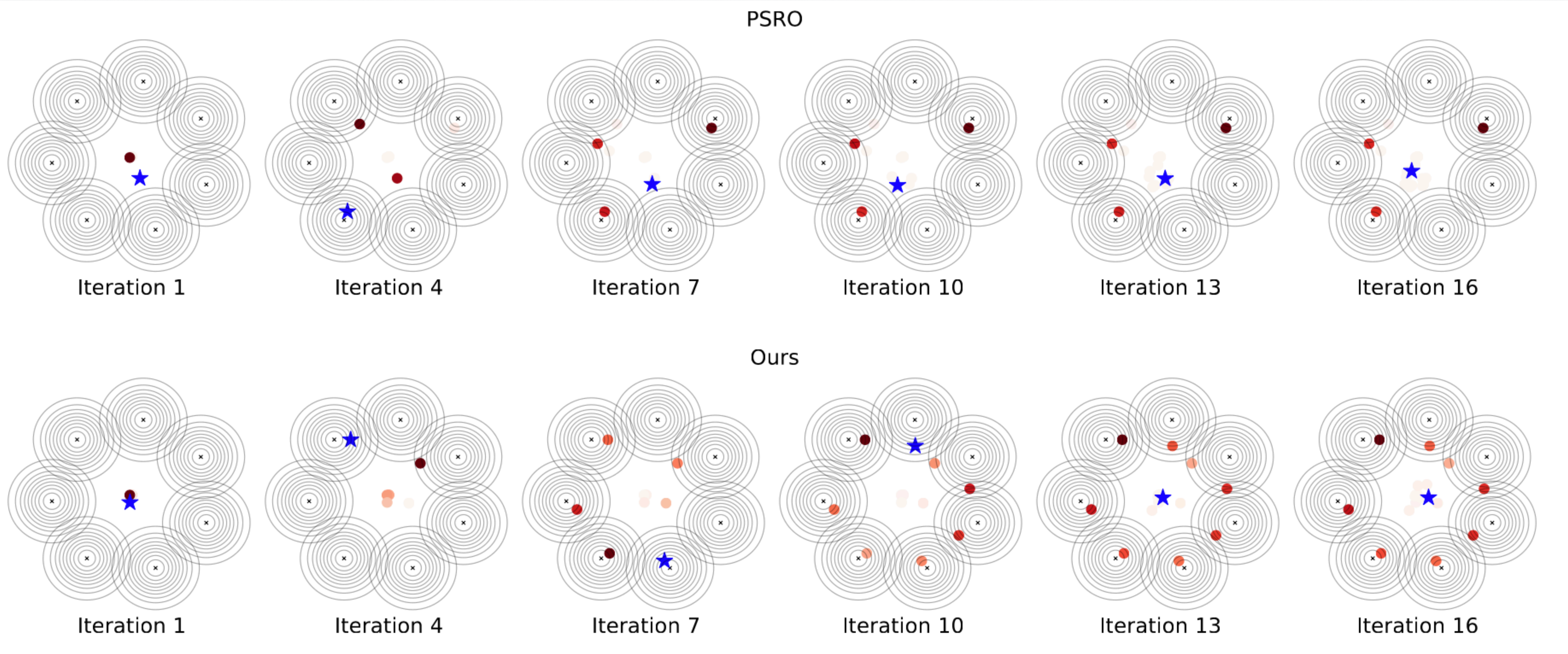
- Is our method any good on the environments where it is trained?
  - Due to long-trajectory issues, we also focus on the approximate best-response setting

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



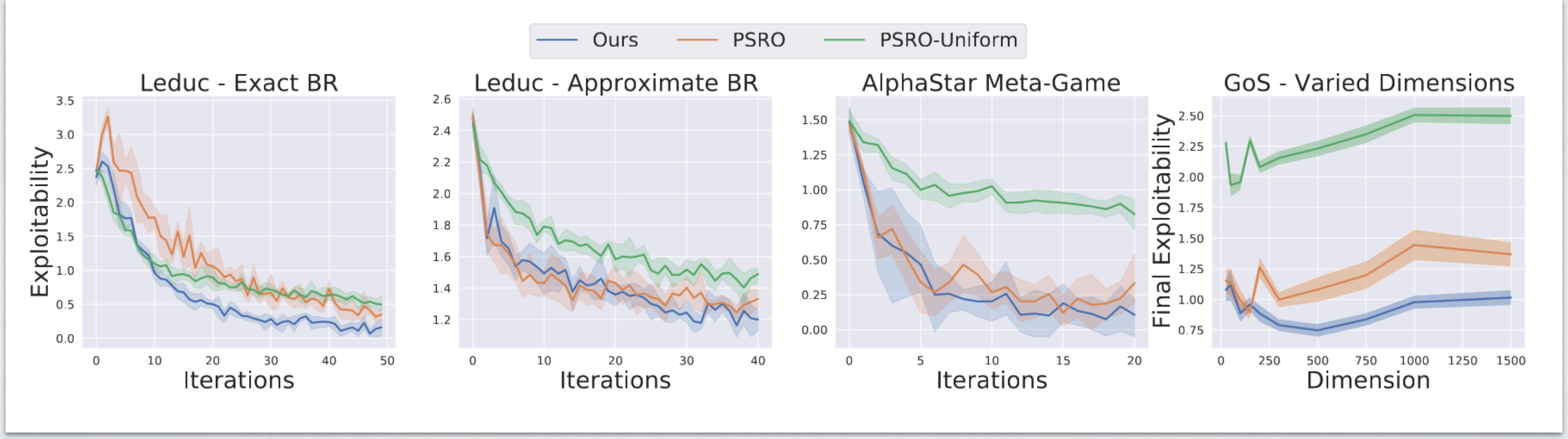
Neural Auto-Curricula Results

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

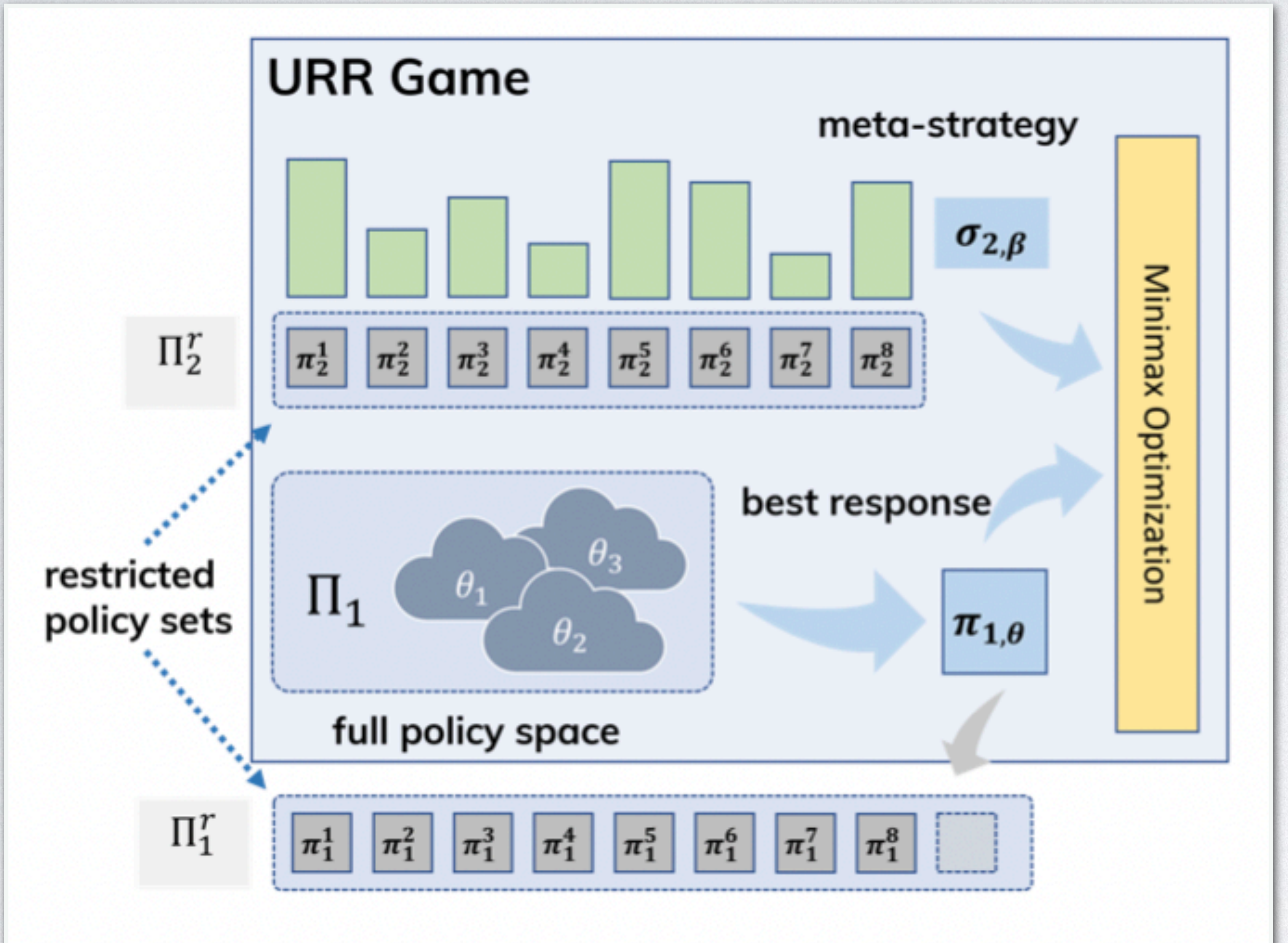


Neural Auto-Curricula Results

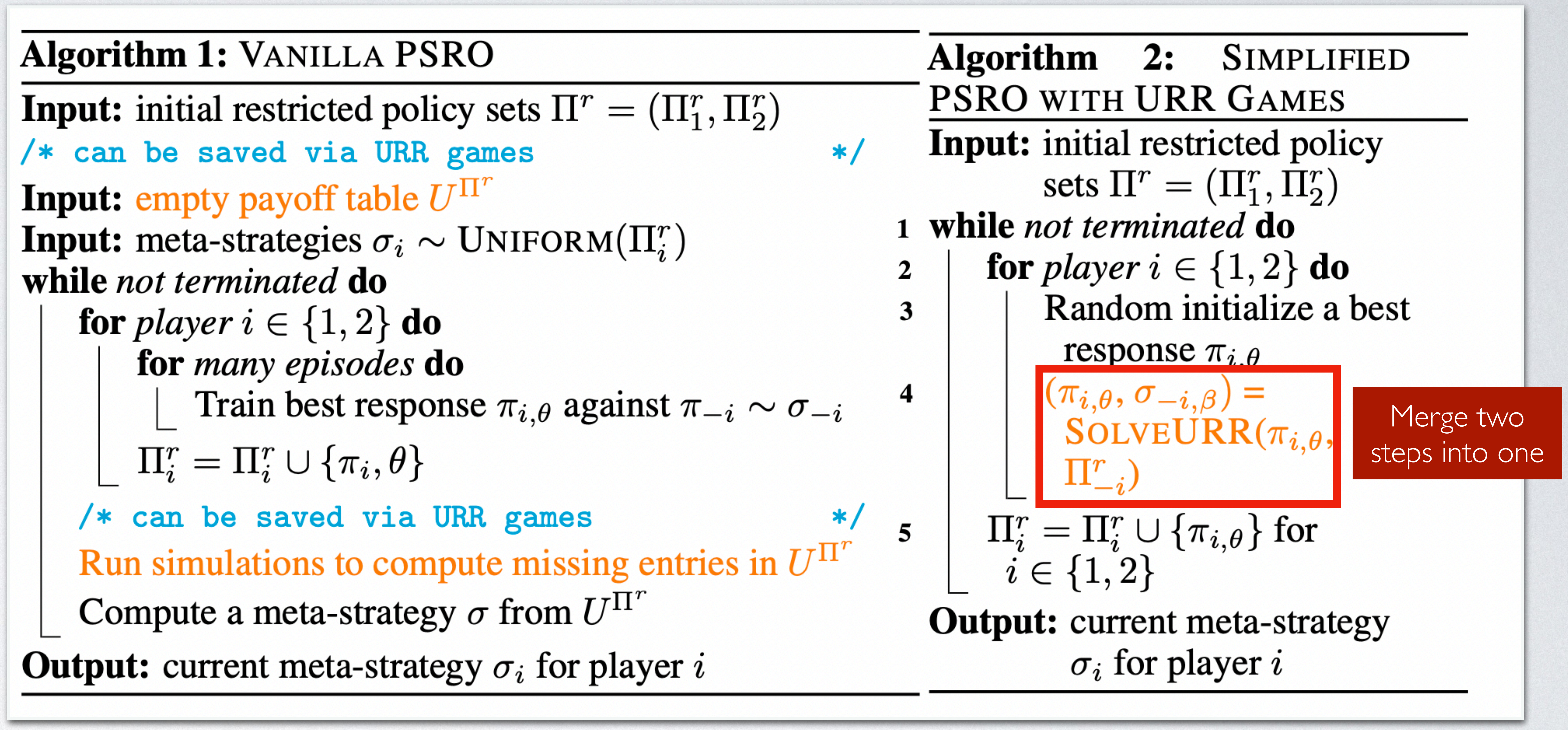
- Can the learned solver generalise over different games?
  - the most promising and striking aspect of NAC - Train on small games and generalise to large games, e.g., train on Kukn Poker and test on Leduc Poker



Efficient PSRO [Ming et. al .2022]



(a) Player 1's URR game dynamics

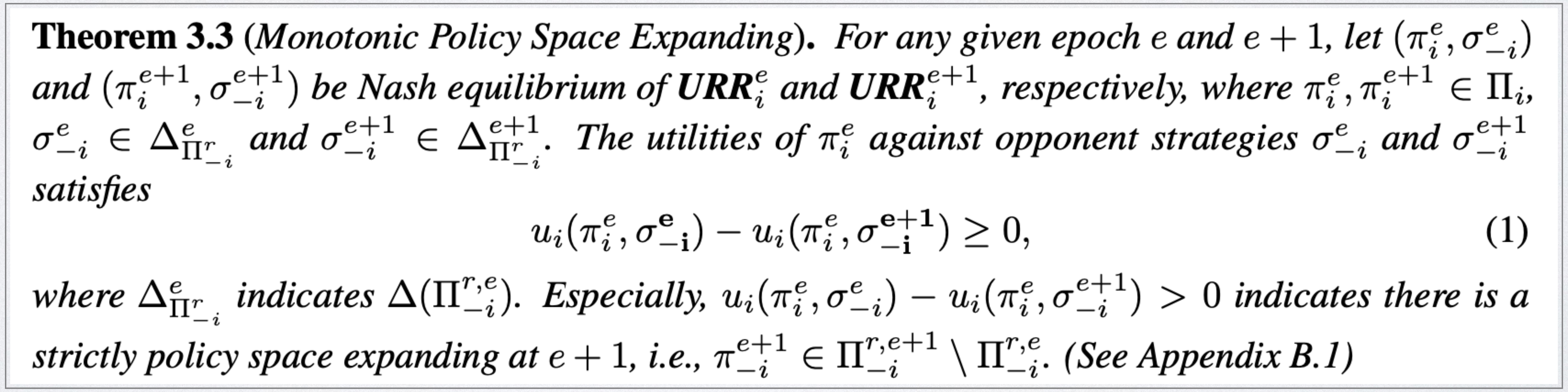


```
Algorithm 1: VANILLA PSRO
Input: initial restricted policy sets $\Pi^r = (\Pi_1^r, \Pi_2^r)$
/* can be saved via URR games */
Input: empty payoff table $U^{\Pi^r}$
Input: meta-strategies $\sigma_i \sim$ UNIFORM$(\Pi_i^r)$
while not terminated do
    for player $i \in \{1, 2\}$ do
        for many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

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

Merge two steps into one

Monotonic Improvement Guarantee

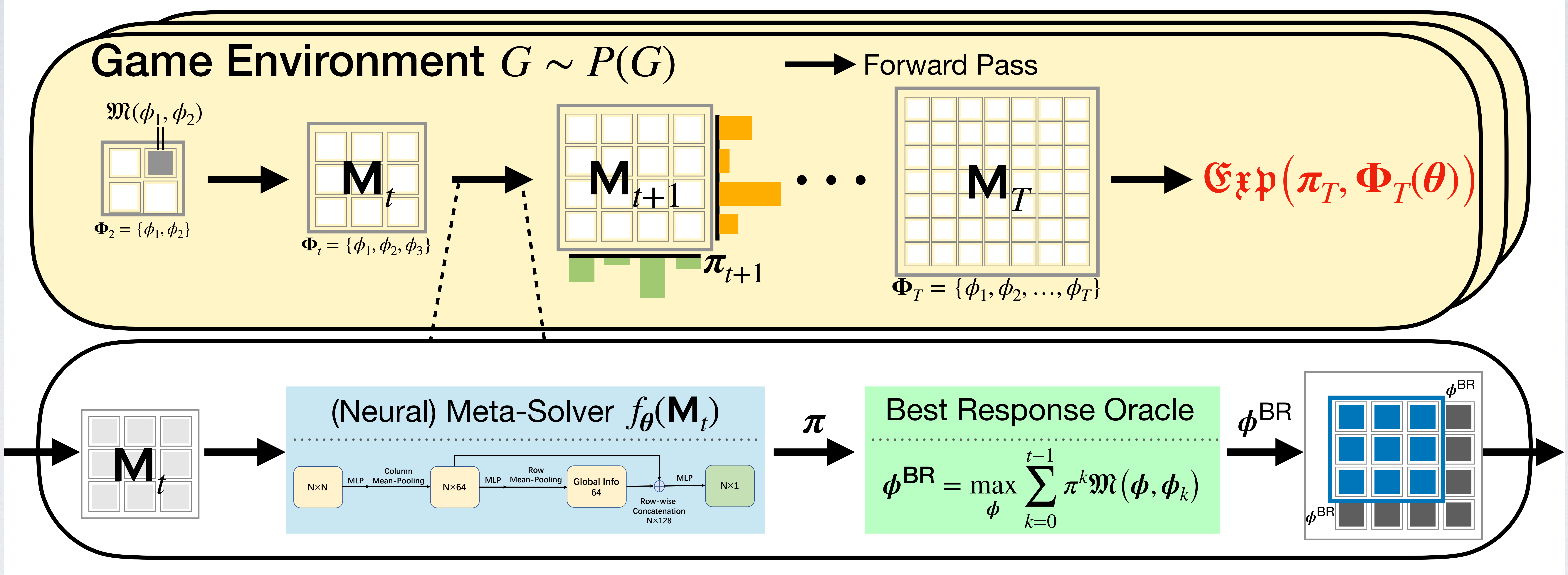


Theorem 3.3 (*Monotonic Policy Space Expanding*). *For any given epoch e and $e+1$, let (π_i^e, σ_{-i}^e) and $(\pi_i^{e+1}, \sigma_{-i}^{e+1})$ be Nash equilibrium of* ***URR***$_i^e$ *and* ***URR***$_i^{e+1}$*, respectively, where $\pi_i^e, \pi_i^{e+1} \in \Pi_i$, $\sigma_{-i}^e \in \Delta_{\Pi_{-i}^r}^e$ and $\sigma_{-i}^{e+1} \in \Delta_{\Pi_{-i}^r}^{e+1}$. The utilities of π_i^e against opponent strategies σ_{-i}^e and σ_{-i}^{e+1} satisfies*

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

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

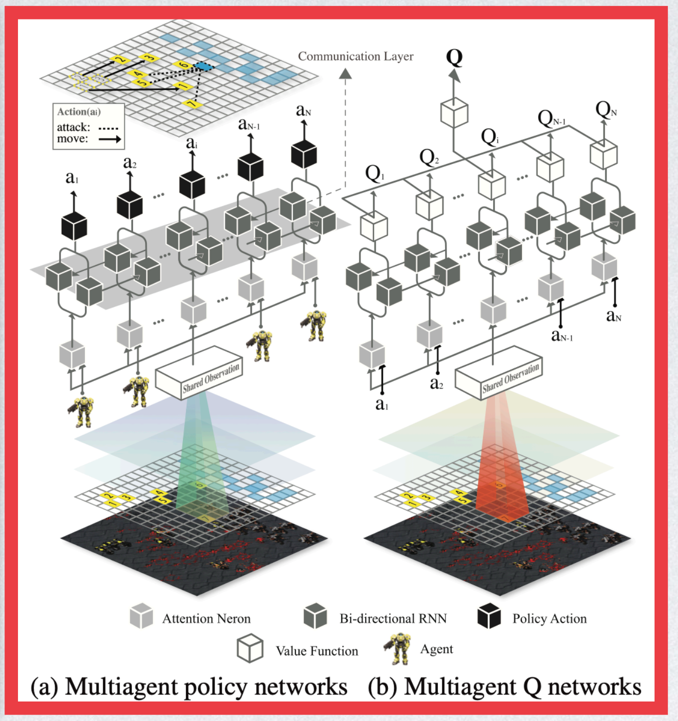
Summary: PSRO Incorporate Many Variants


Game Environment $G \sim P(G)$
Forward Pass
$\mathfrak{M}(\phi_1, \phi_2)$
$\Phi_2 = \{\phi_1, \phi_2\}$
$\mathbf{M}_t$
$\Phi_t = \{\phi_1, \phi_2, \phi_3\}$
$\mathbf{M}_{t+1}$
π_{t+1}
$\mathbf{M}_T$
$\Phi_T = \{\phi_1, \phi_2, \ldots, \phi_T\}$
$\mathfrak{Exp}\left(\pi_T, \Phi_T(\theta)\right)$
$\mathbf{M}_t$
(Neural) Meta-Solver $f_\theta(\mathbf{M}_t)$
N×N
MLP
Column Mean-Pooling
N×64
MLP
Row Mean-Pooling
Global Info 64
Row-wise Concatenation N×128
MLP
N×1
π
Best Response Oracle
$\phi^{\text{BR}} = \max_{\phi} \sum_{k=0}^{t-1} \pi^k \mathfrak{M}(\phi, \phi_k)$
ϕ^{BR}
ϕ^{BR}
ϕ^{BR}

Elo rating
Correlated equilibrium
Nash equilibrium
Replicator dynamics
α-Rank/α^α-Rank

naive self-play
fictitious play
double oracle/PSRO
PSRO-Nash
α-PSRO
JPSRO

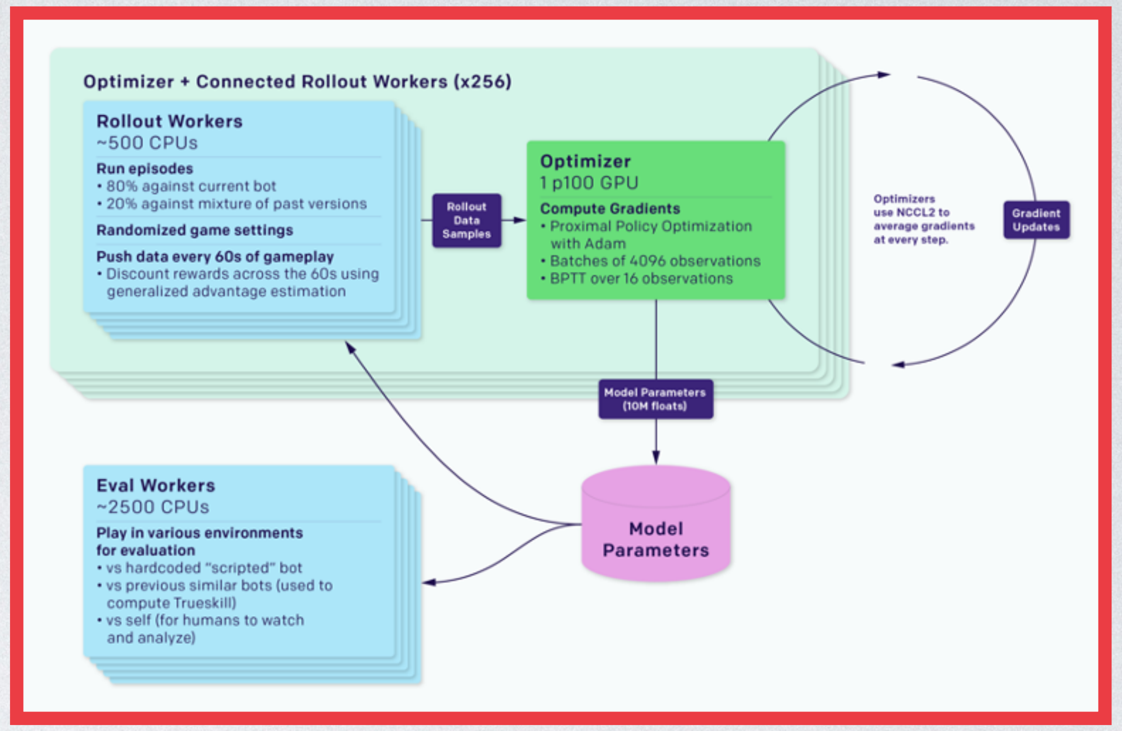
Training Population of RL Agents Require Powerful AI Systems



StarCraft micro-management

BiCNet, deep MARL methods

1-2 GPUs, **1-2** days


Optimizer + Connected Rollout Workers (x256)
Rollout Workers
~500 CPUs
Run episodes
• 80% against current bot
• 20% against mixture of past versions
Randomized game settings
Push data every 60s of gameplay
• Discount rewards across the 60s using generalized advantage estimation
Rollout Data Samples
Optimizer
1 p100 GPU
Compute Gradients
• Proximal Policy Optimization with Adam
• Batches of 4096 observations
• BPTT over 16 observations
Optimizers use NCCL2 to average gradients at every step.
Gradient Updates
Model Parameters (10M floats)
Model Parameters
Eval Workers
~2500 CPUs
Play in various environments for evaluation
• vs hardcoded "scripted" bot
• vs previous similar bots (used to compute Trueskill)
• vs self (for humans to watch and analyze)

Dota 2 full game (OpenAI Five)

Population-based + Rapid training system

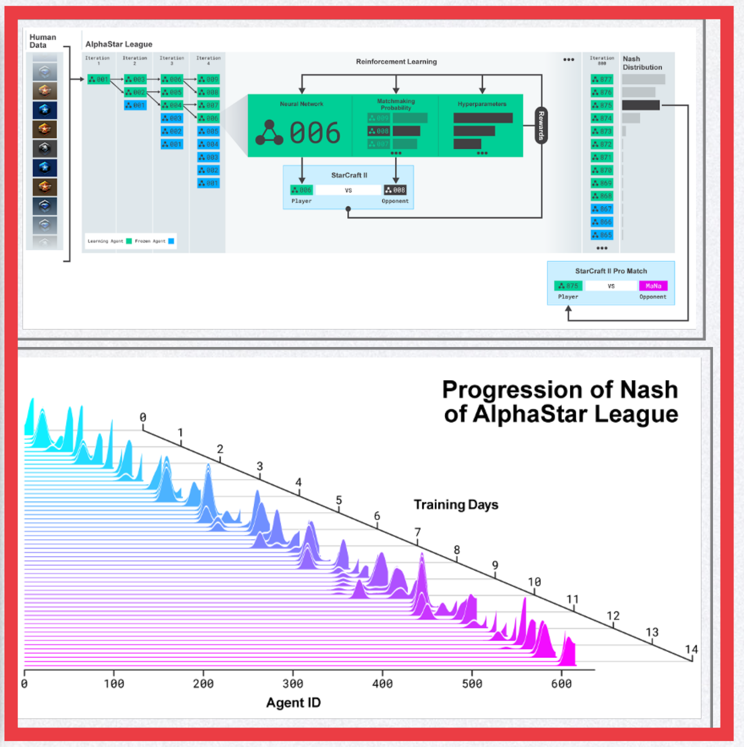
128,000 CPUs, **100** GPUs, **180** years of plays per day

2017.1

2019.1

2019.4

2020.11

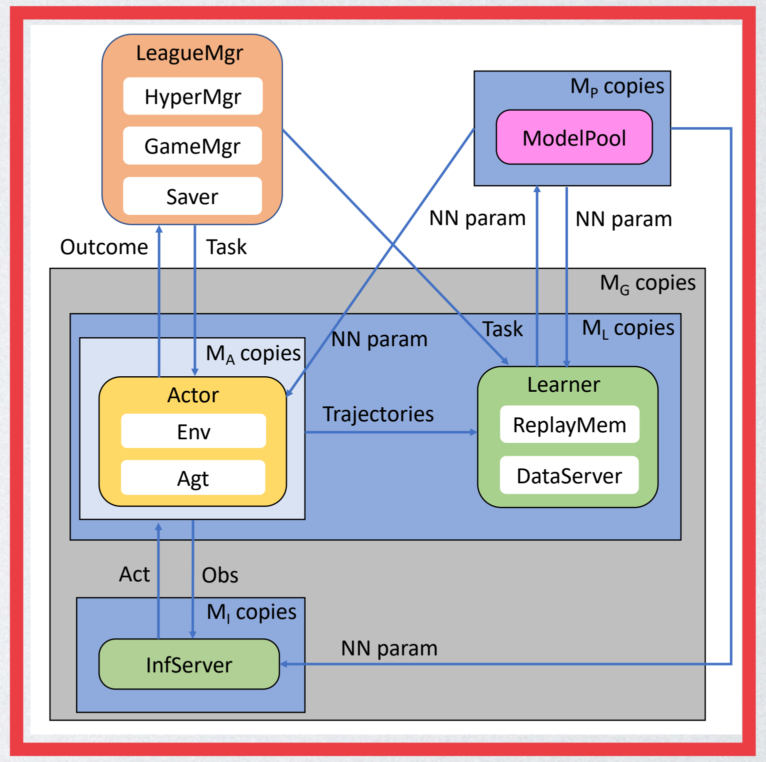


StarCraft full game (AlphaStar)

Populating-based Training

Training for single agent costs **14** days, **16** TPUs/Agent,

200 years of real-time play.



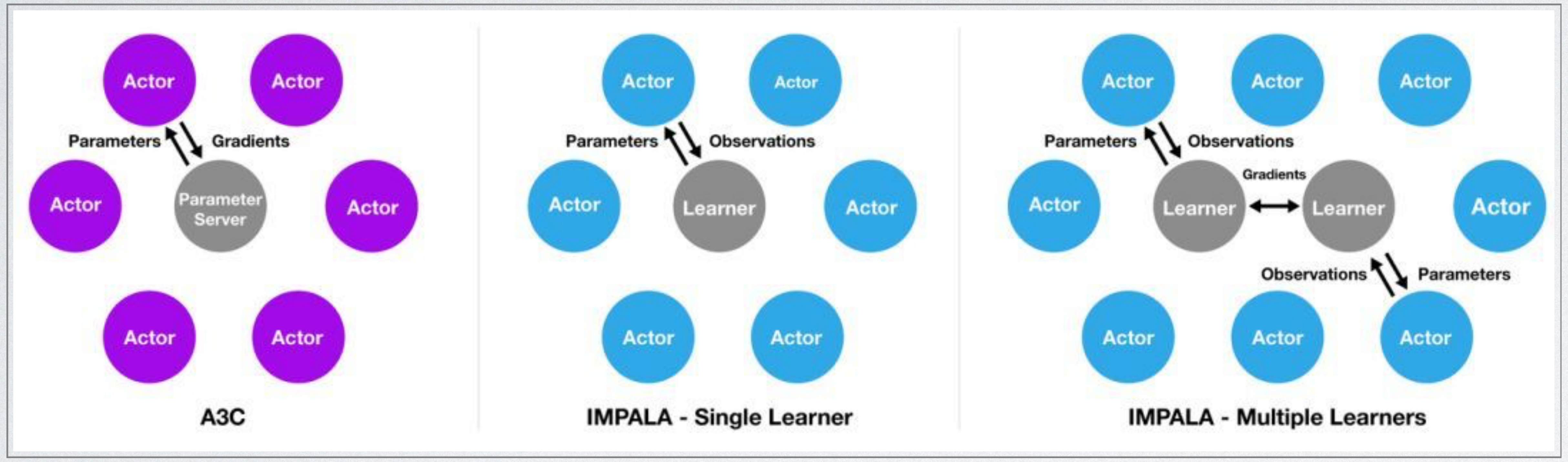
王者荣耀 (绝悟)

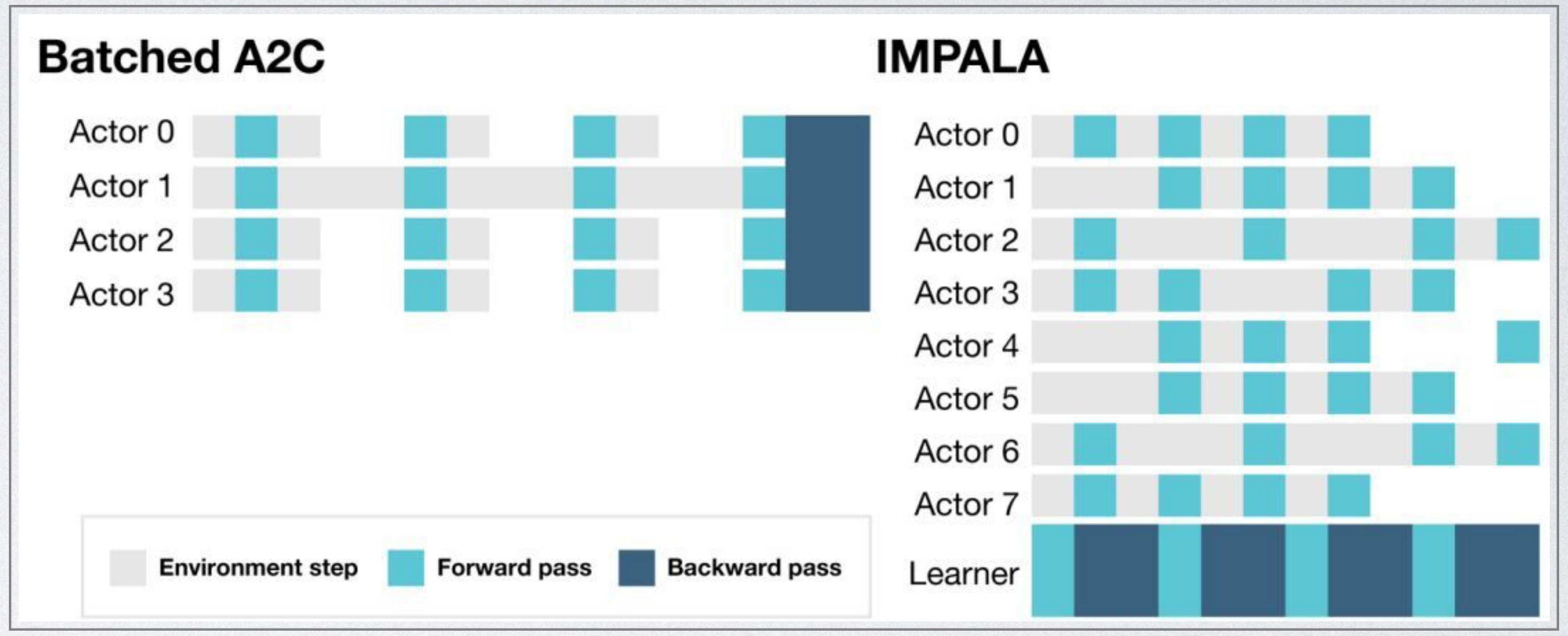
Populating-based Competitive Self-play + Policy distillation

35,000 CPUs, **320** GPUs, begin to converge after **336** hours

Training Population of RL Agents Require Powerful AI Systems

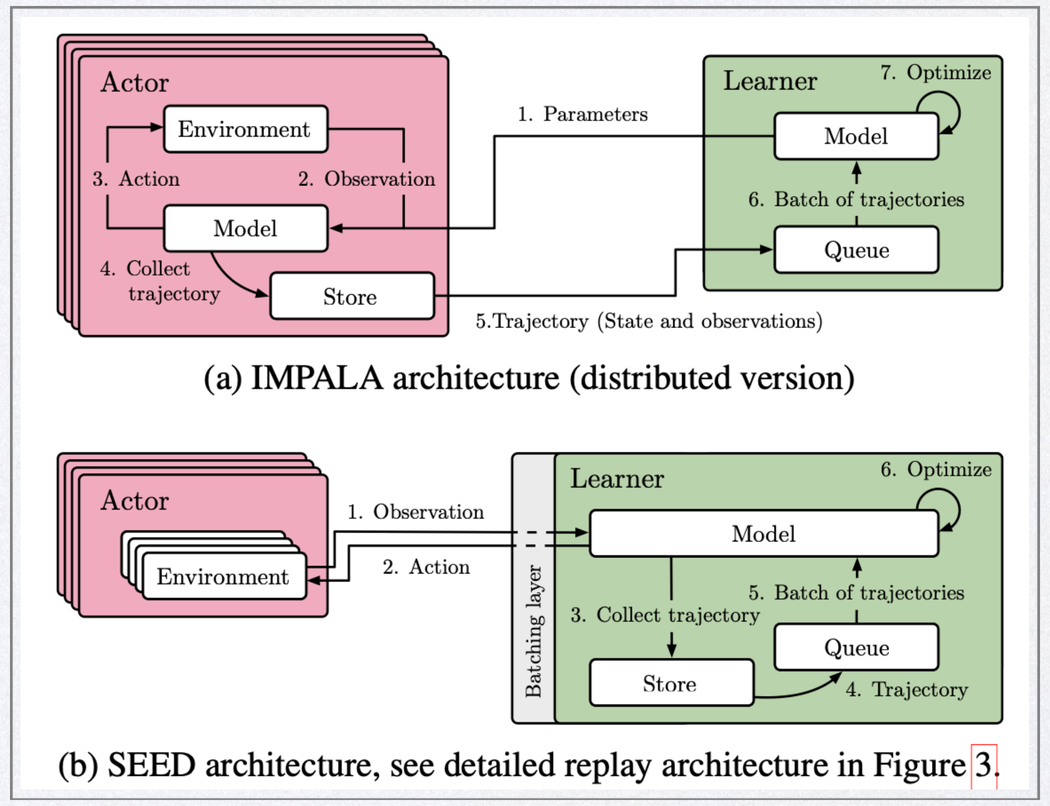
- Distributed RL systems has made substantial progress nowadays.
  - A rule-of-thumb is that: decoupling learning and rollout enables great speedups.





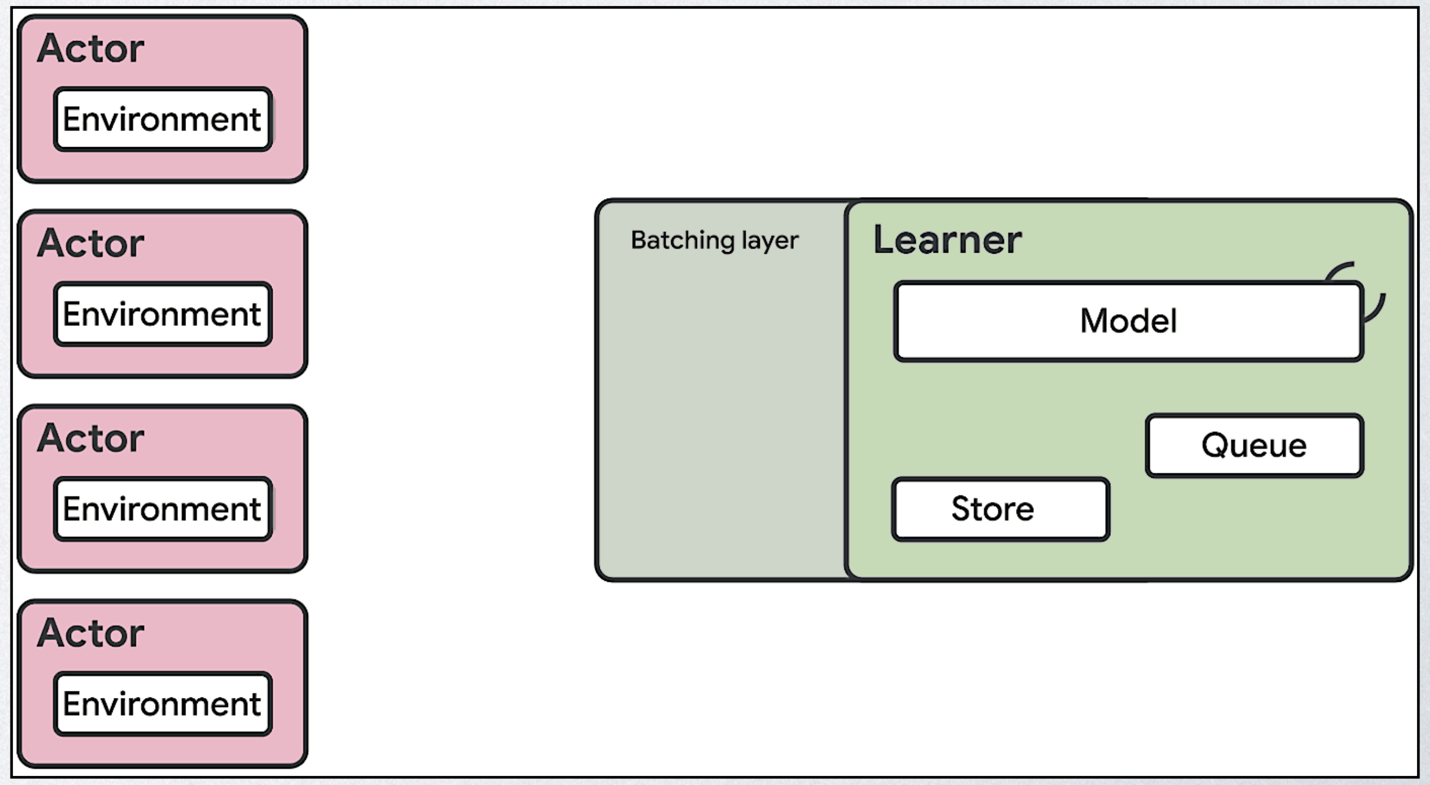
https://cloud.tencent.com/developer/article/1119569

- 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



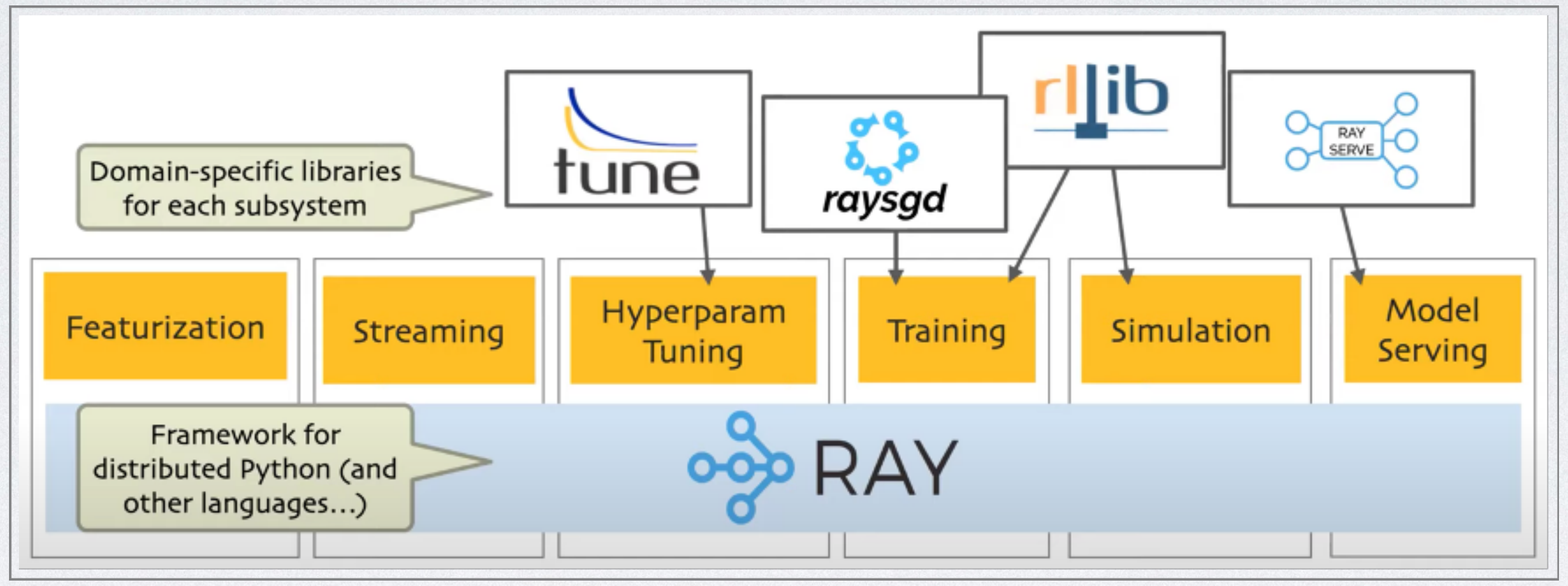
(a) IMPALA architecture (distributed version)

(b) SEED architecture, see detailed replay architecture in Figure 3.



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.



Domain-specific libraries for each subsystem

tune

raysgd

rllib

RAY SERVE

Featurization

Streaming

Hyperparam Tuning

Training

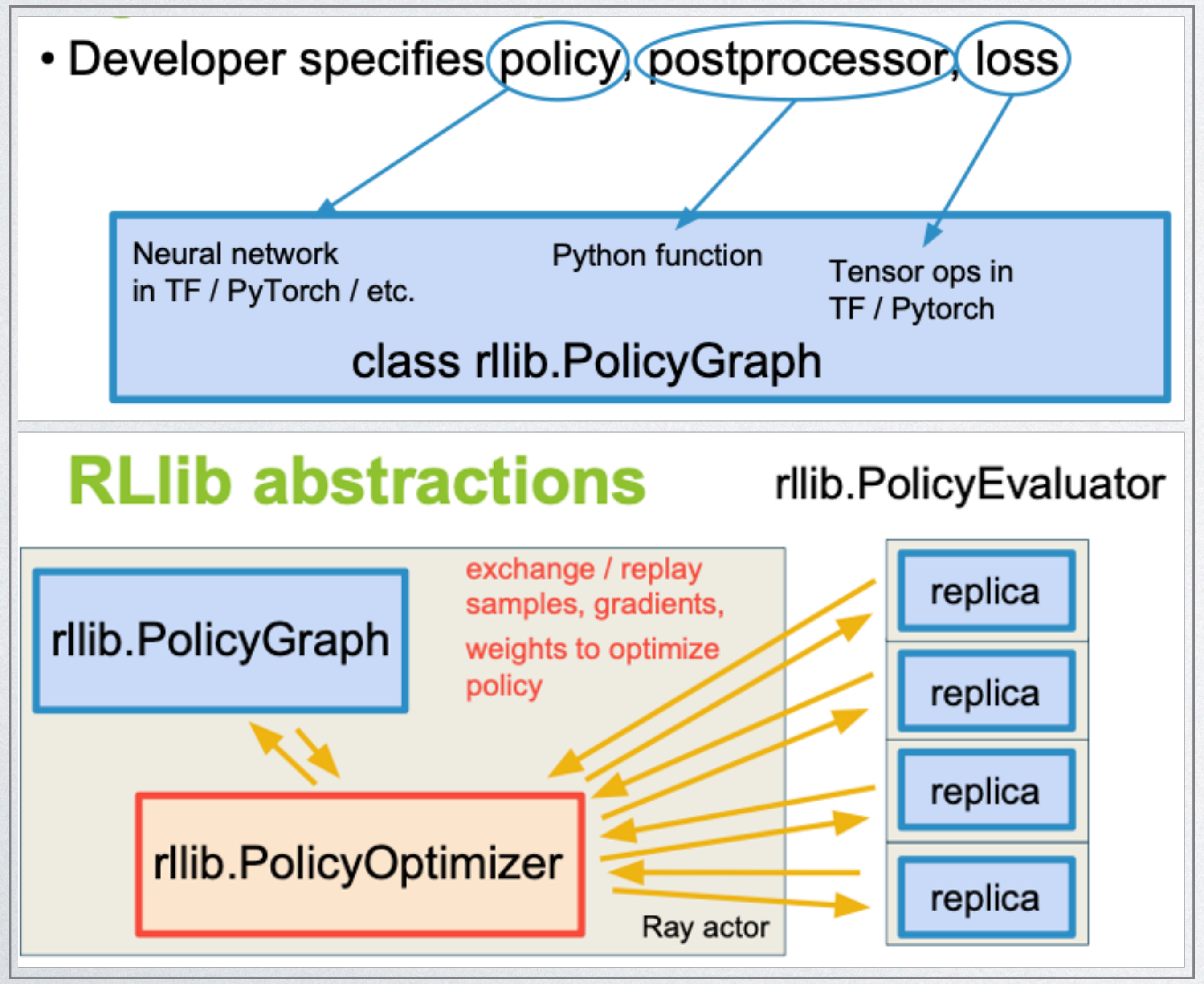
Simulation

Model Serving

Framework for distributed Python (and other languages…)

RAY

  - The implementation of RLlib uses RAY for RL applications.



• Developer specifies policy, postprocessor, loss

Neural network in TF / PyTorch / etc.

Python function

Tensor ops in TF / Pytorch

class rllib.PolicyGraph

RLlib abstractions

rllib.PolicyEvaluator

rllib.PolicyGraph

exchange / replay samples, gradients, weights to optimize policy

rllib.PolicyOptimizer

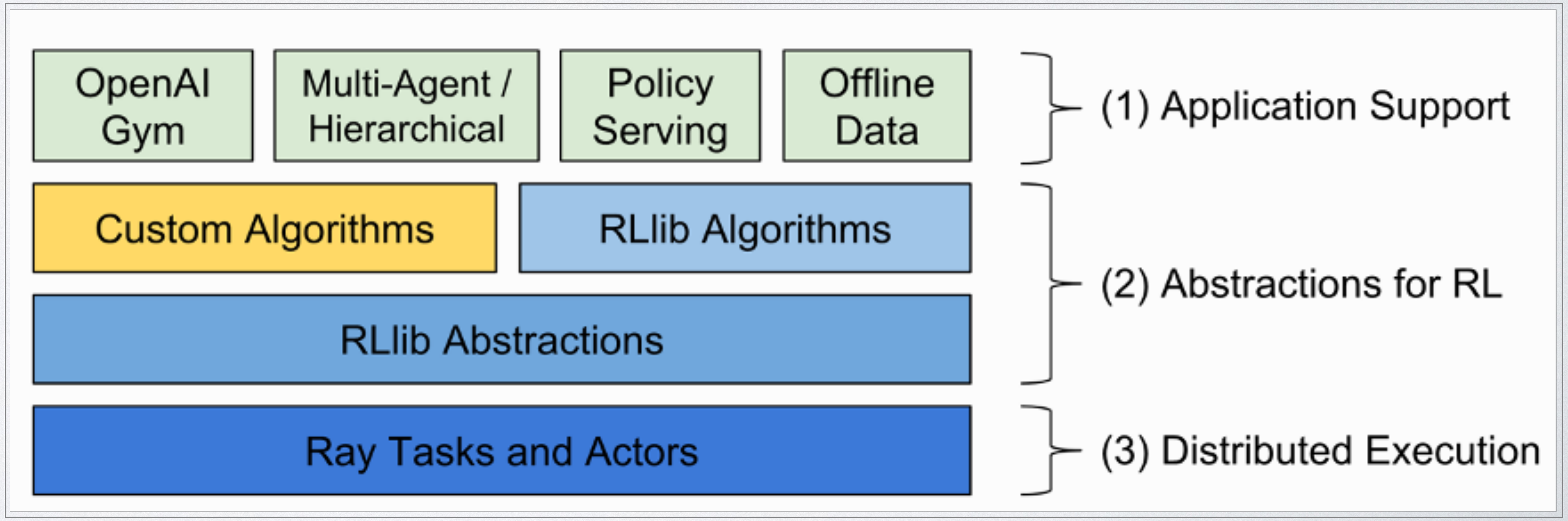
Ray actor

replica

replica

replica

replica



OpenAI Gym

Multi-Agent / Hierarchical

Policy Serving

Offline Data

(1) Application Support

Custom Algorithms

RLlib Algorithms

RLlib Abstractions

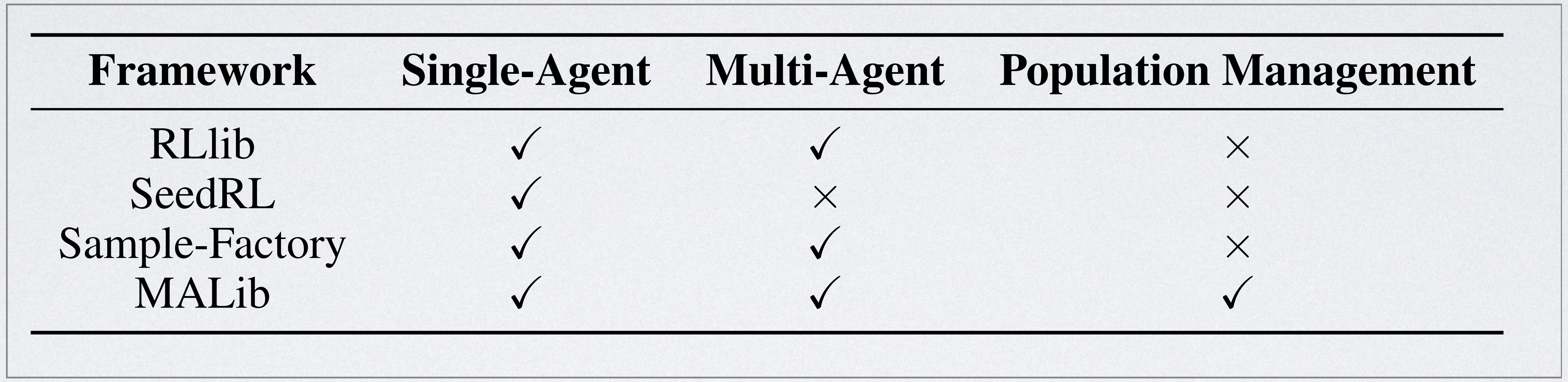
(2) Abstractions for RL

Ray Tasks and Actors

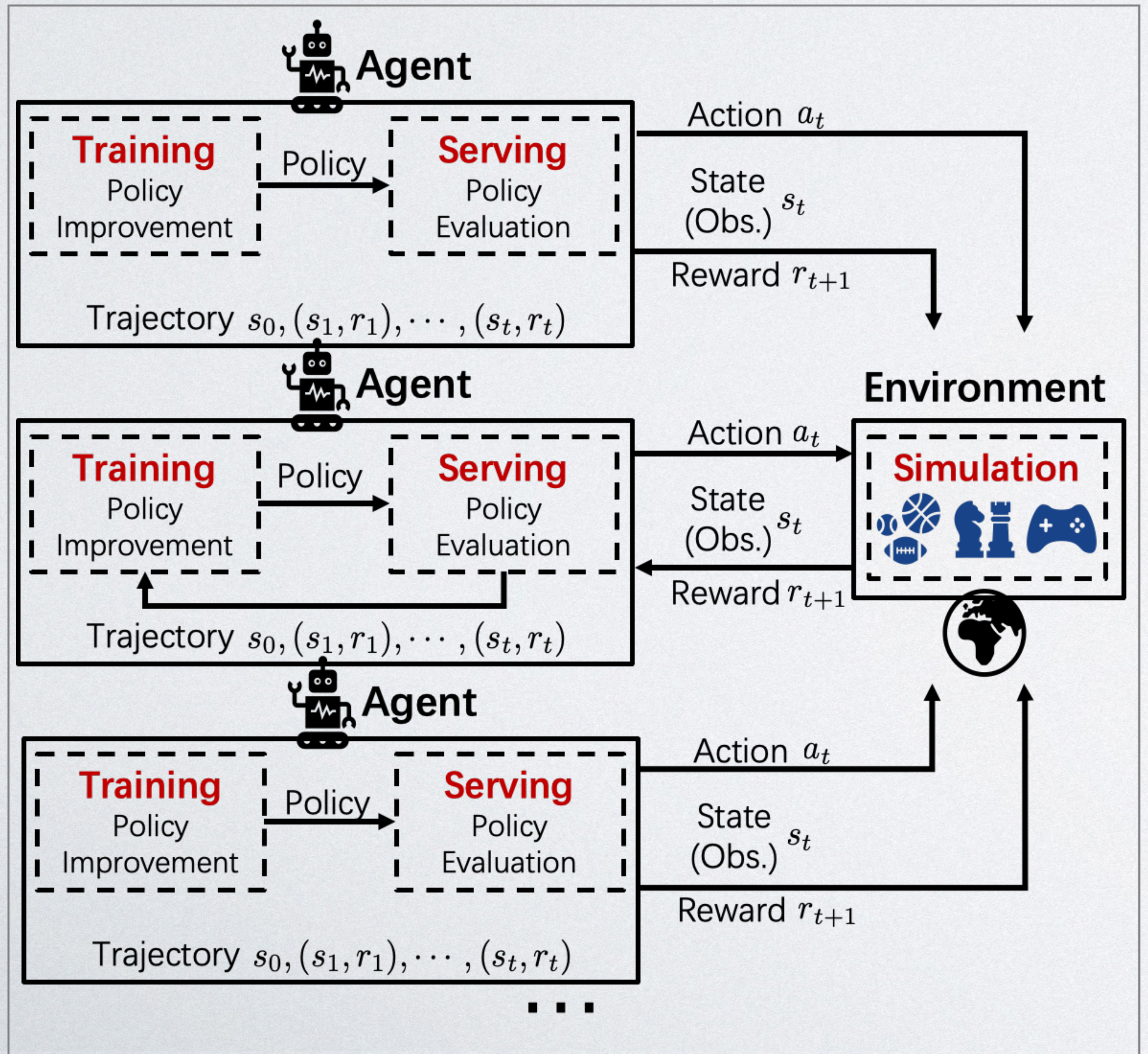
(3) Distributed Execution

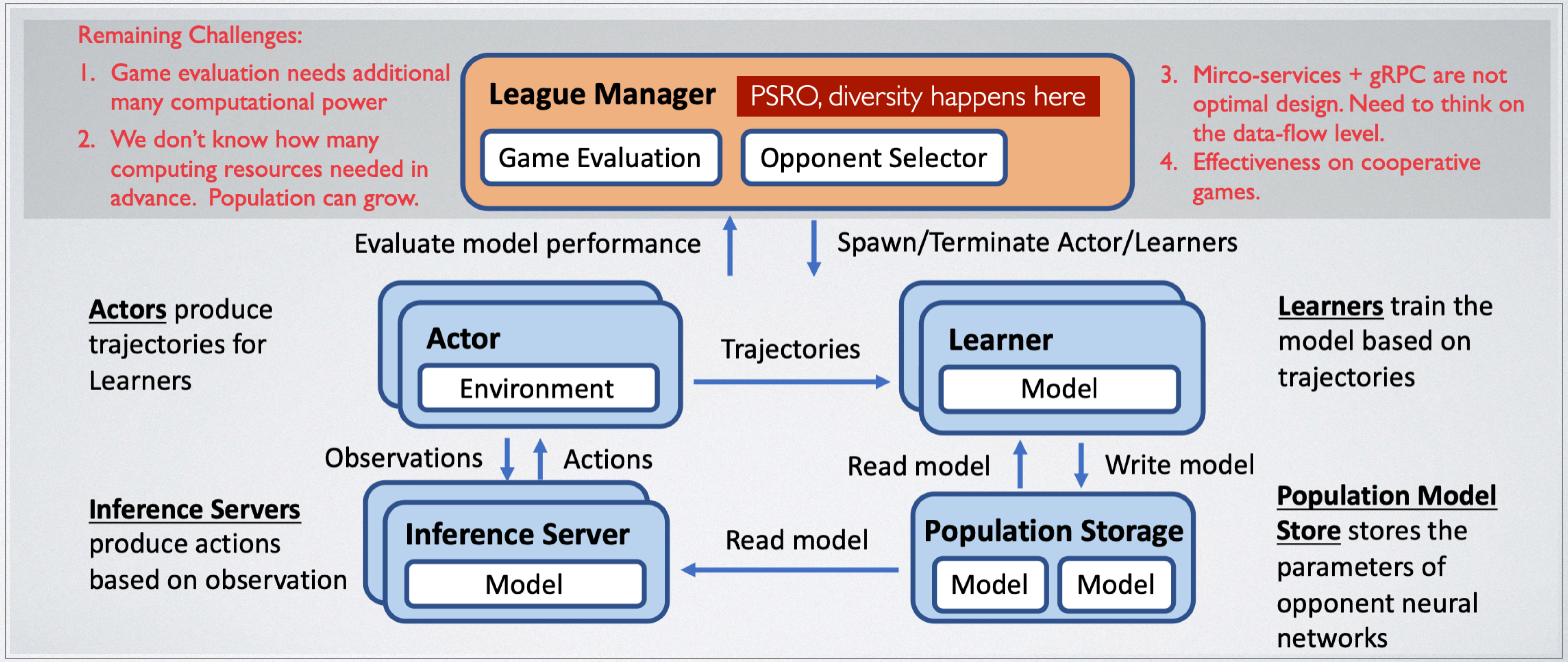
PB-MARL Poses New Requirements for AI Systems

- 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	✓	✓	✓





Remaining Challenges:

1. Game evaluation needs additional many computing power
2. We don't know how many computing resources needed in advance. Population can grow.
3. Mirco-services + gRPC are not optimal design. Need to think on the data-flow level.
4. Effectiveness on cooperative games.

Actors produce trajectories for Learners

Learners train the model based on trajectories

Inference Servers produce actions based on observation

Population Model Store stores the parameters of opponent neural networks

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

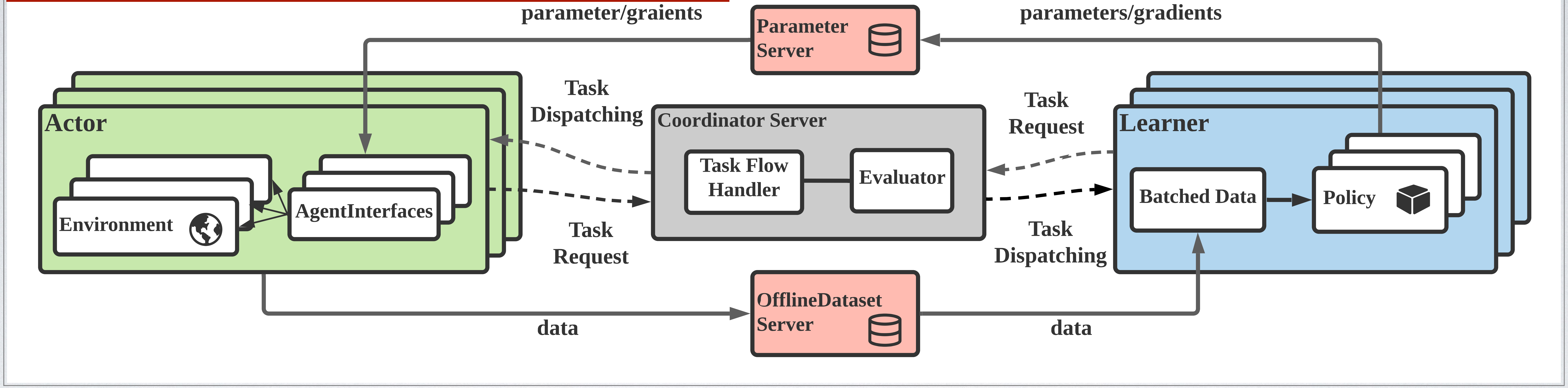
- **Large-scale rollout & training framework**
  - Support heterogenous tasks
  - High throughput distributed frameworks
  - Maximize utilization of different hardware
- **Support population-based training**
  - self-play / league mechanism
  - meta-game analysis that based on game/graph theory and representation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

MALib: Designed for PB-MARL

For Multi-Agent Reinforcement Learning

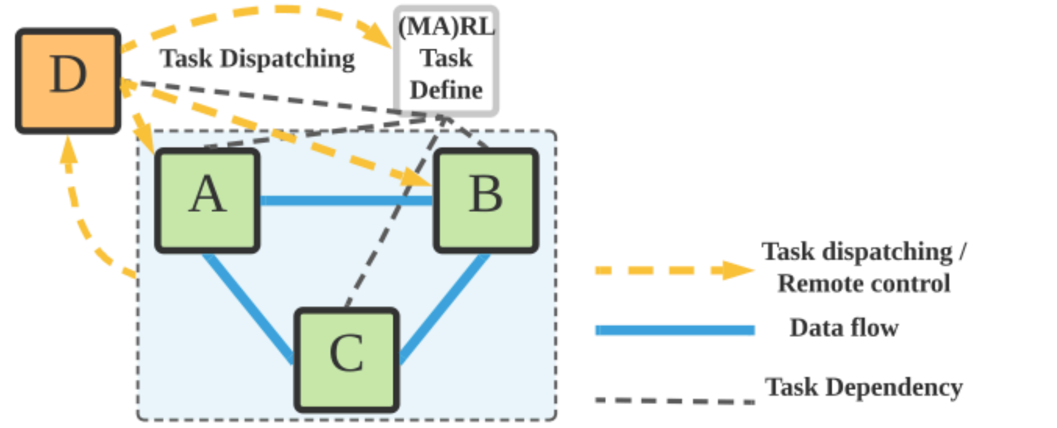
- MALib: https://malib.io

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



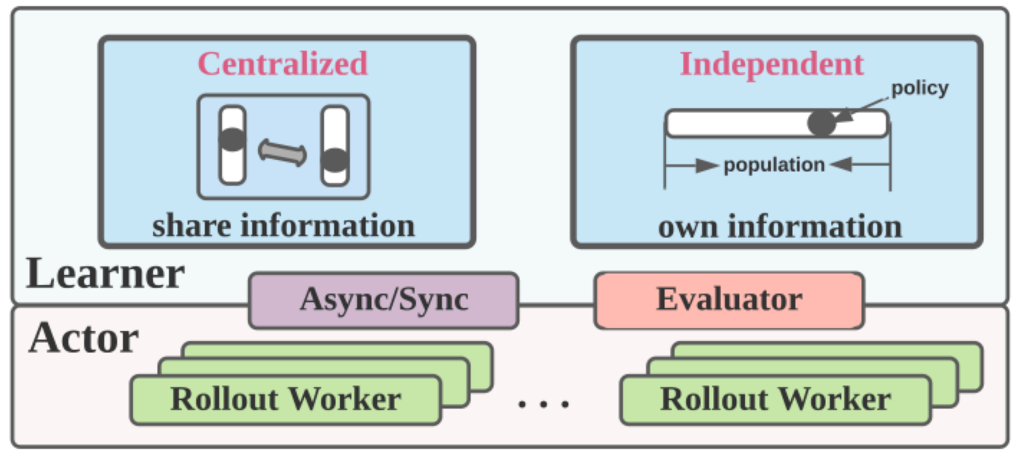
MALib: Centralized Task Dispatching Model (CTD)

- 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: …



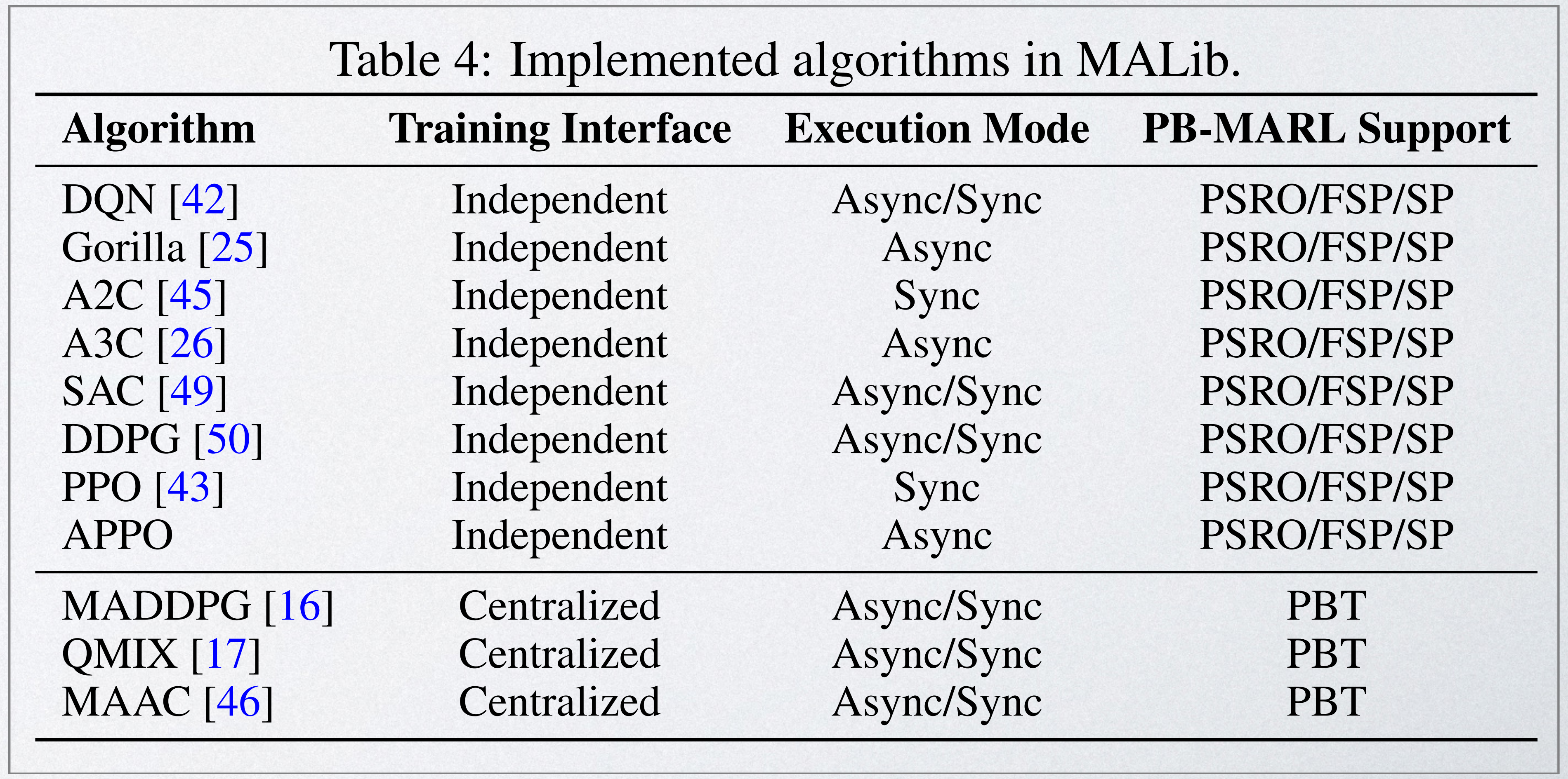
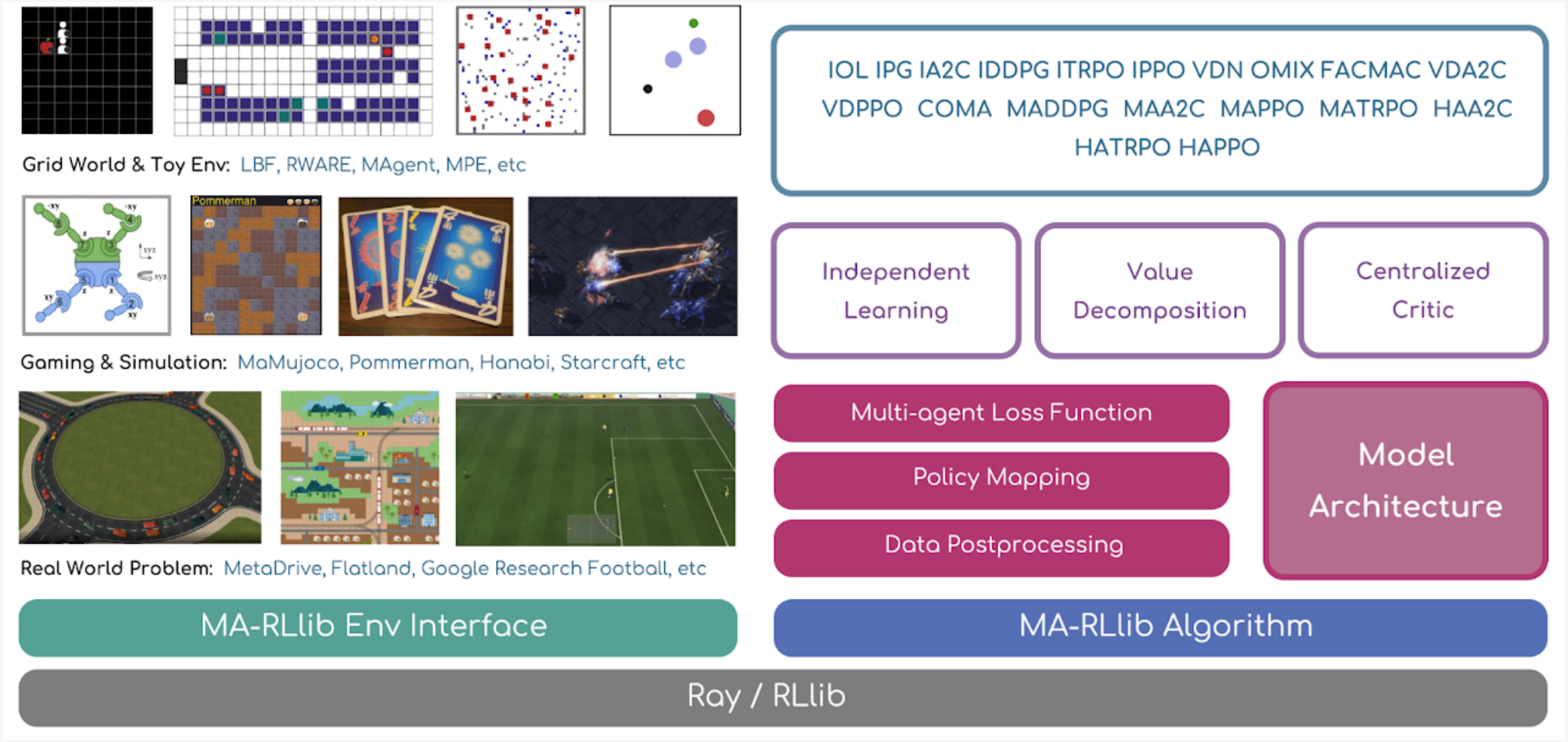


Table 4: Implemented algorithms in MALib.

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

MARLlib: the MARL Extension for RLlib

- MARLlib: https://github.com/Replicable-MARL/MARLlib



Grid World & Toy Env: LBF, RWARE, MAgent, MPE, etc

Gaming & Simulation: MaMujoco, Pommerman, Hanabi, Starcraft, etc

Real World Problem: MetaDrive, Flatland, Google Research Football, etc

IQL IPG IA2C IDDPG ITRPO IPPO VDN QMIX FACMAC VDA2C VDPPO COMA MADDPG MAA2C MAPPO MATRPO HAA2C HATRPO HAPPO

Independent Learning

Value Decomposition

Centralized Critic

Multi-agent Loss Function

Policy Mapping

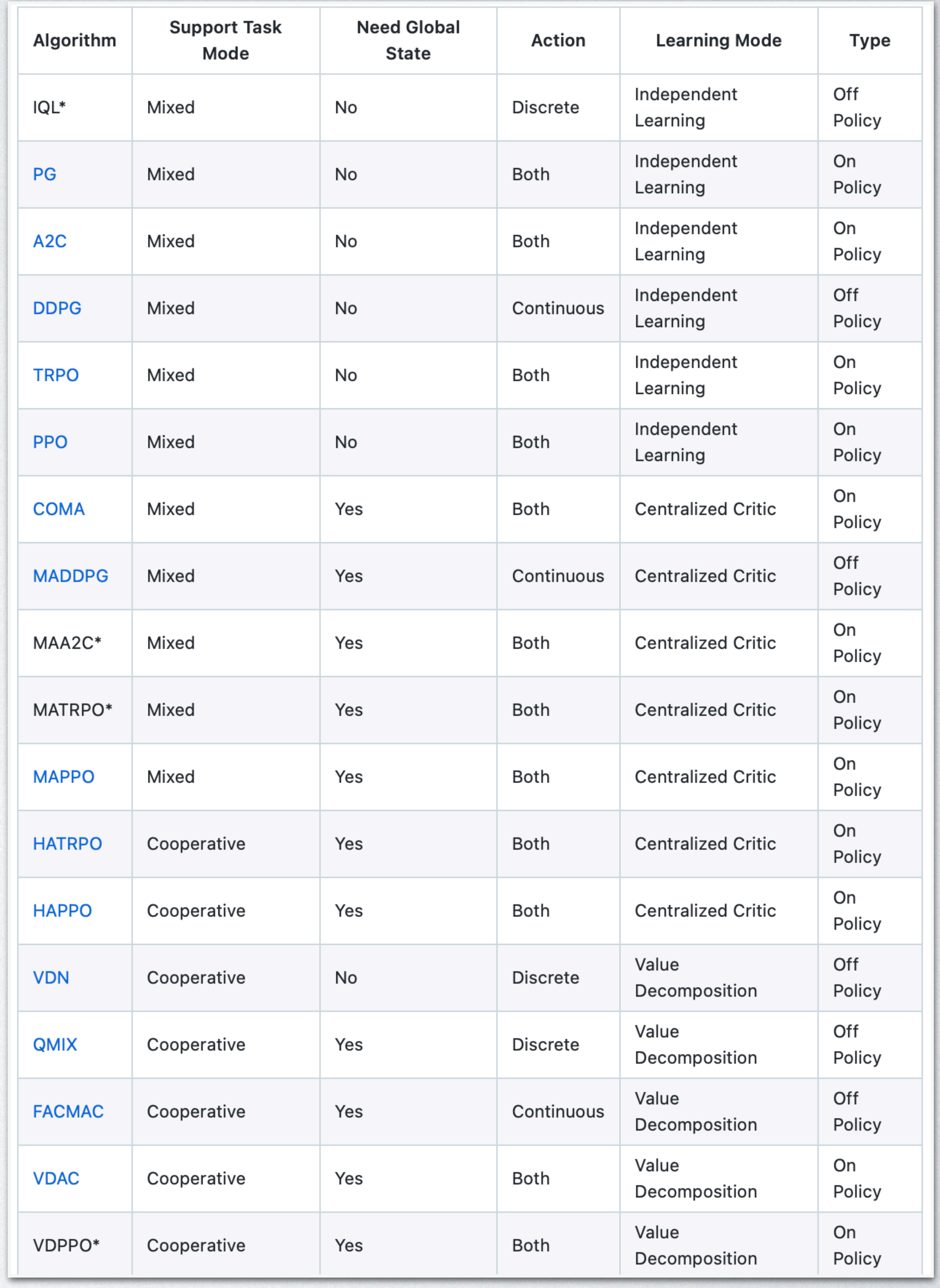
Data Postprocessing

Model Architecture

MA-RLlib Env Interface

MA-RLlib Algorithm

Ray / RLlib



Algorithm	Support Task Mode	Need Global State	Action	Learning Mode	Type
IQL*	Mixed	No	Discrete	Independent Learning	Off Policy
PG	Mixed	No	Both	Independent Learning	On Policy
A2C	Mixed	No	Both	Independent Learning	On Policy
DDPG	Mixed	No	Continuous	Independent Learning	Off Policy
TRPO	Mixed	No	Both	Independent Learning	On Policy
PPO	Mixed	No	Both	Independent Learning	On Policy
COMA	Mixed	Yes	Both	Centralized Critic	On Policy
MADDPG	Mixed	Yes	Continuous	Centralized Critic	Off Policy
MAA2C*	Mixed	Yes	Both	Centralized Critic	On Policy
MATRPO*	Mixed	Yes	Both	Centralized Critic	On Policy
MAPPO	Mixed	Yes	Both	Centralized Critic	On Policy
HATRPO	Cooperative	Yes	Both	Centralized Critic	On Policy
HAPPO	Cooperative	Yes	Both	Centralized Critic	On Policy
VDN	Cooperative	No	Discrete	Value Decomposition	Off Policy
QMIX	Cooperative	Yes	Discrete	Value Decomposition	Off Policy
FACMAC	Cooperative	Yes	Continuous	Value Decomposition	Off Policy
VDAC	Cooperative	Yes	Both	Value Decomposition	On Policy
VDPPO*	Cooperative	Yes	Both	Value Decomposition	On Policy

END