DEALING WITH NON-TRANSITIVITY IN TWO-PLAYER ZERO-SUM GAMES

Dr. Yaodong Yang
King's College London
www.yangyaodong.com
08/2021

Multi-Agent Reinforcement Learning in Games

Great advantages have been made in 2019!

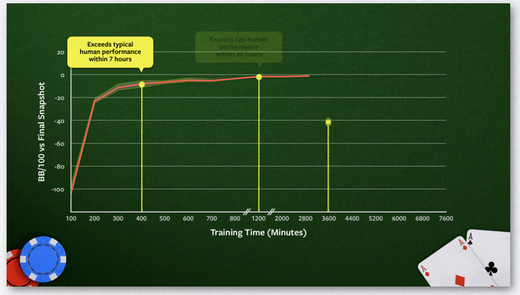
Jan 2016 | Dec 2017 | July 2018 | Jan 2019 | Apr 2019 | July 2019 | Sep 2019

AlphaGO Series

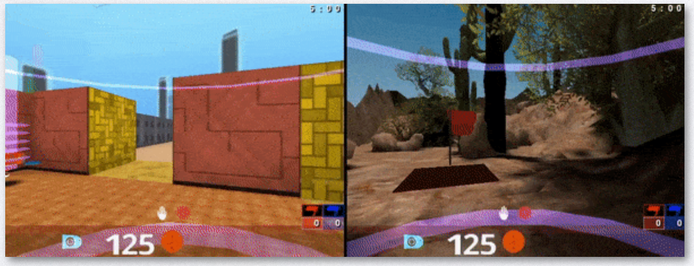
technique of single-agent decision-making is mature

AlphaStar (DeepMind)

Pluribus Poker (FAIR)

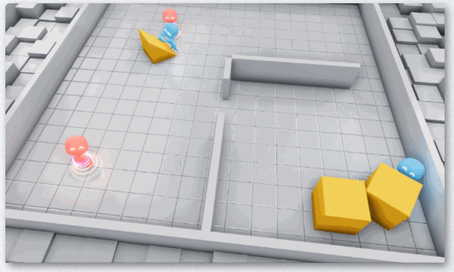


Capture-the-flag (DeepMind)



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 $(R^1, \ldots, R^N)$

Black-box multi-agent game engine

input

Our algorithm:

output

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

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

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

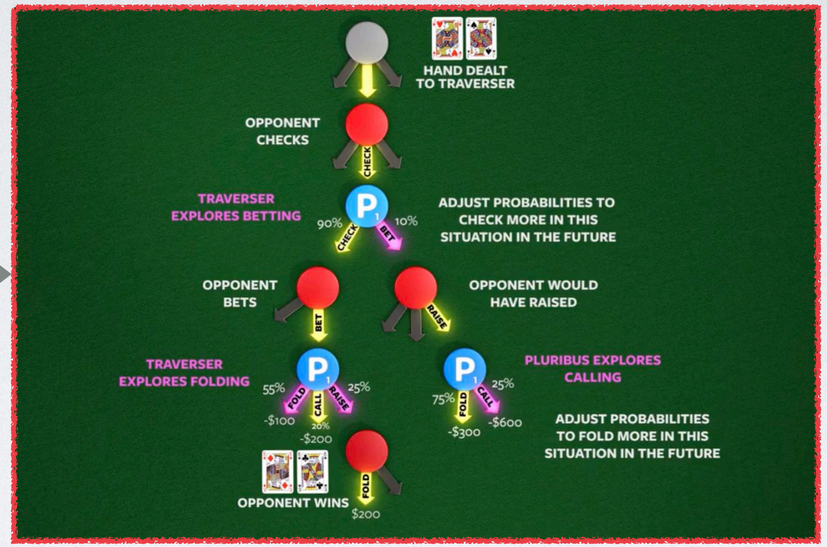
Input: a joint strategy $(\pi^1, \ldots, \pi^N)$

Not All Games Can be Solved by Tree Search

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

Black-box multi-agent game engine

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



Regret based methods: Poker Type

Best response based methods: StarCraft type

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

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

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

Planning is not always feasible. StarCraft has 10^{26} choices per step (vs. the game tree size of chess 10^{50}, Texas holdem 10^{80}, GO 10^{170})

Enumerating all policies' actions at each state and then playing a randomise best response is infeasible (i.e. RPS can not apply)

Solution: design a game of game — meta-game, the problem problem, auto-curricula.

Naive Self-play Will Not Work

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

PPO + PBT + Self-play = Panacea ?

self-plays

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: $\mathbf{v}_{T+1}$

$$\left(\pi^1, \pi^2\right) \rightarrow \left(\pi^1, \pi^{2,*} = \textbf{Br}(\pi^1)\right) \rightarrow \left(\pi^{1,*} = \textbf{Br}(\pi^{2,*}), \pi^{2,*}\right)$$

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

Naive Self-play Will Not Work

- It is because of Non-Transitivity

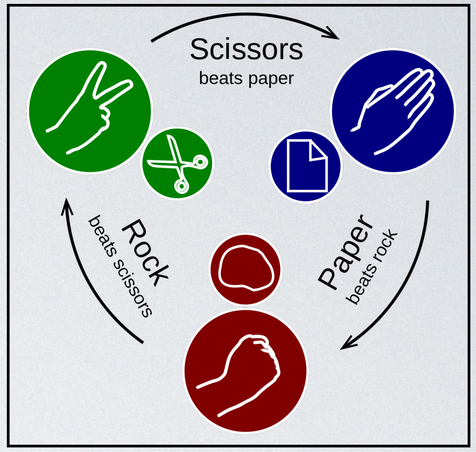
$$\int_W \phi(\mathbf{v}, \mathbf{w}) \cdot d\mathbf{w} = 0, \quad \forall \mathbf{v} \in W$$

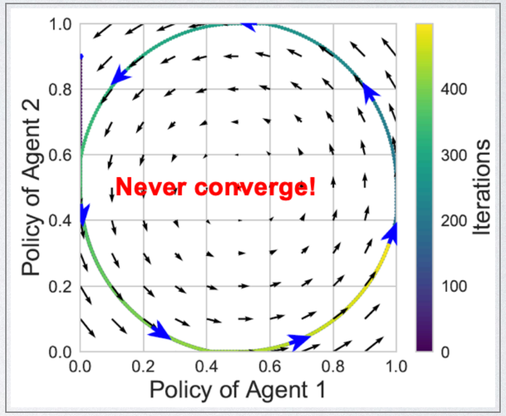
  - Rock-Paper-Scissor game:

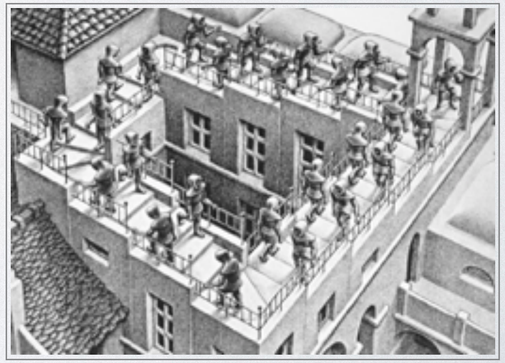
$$\begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$

  - Disc game:

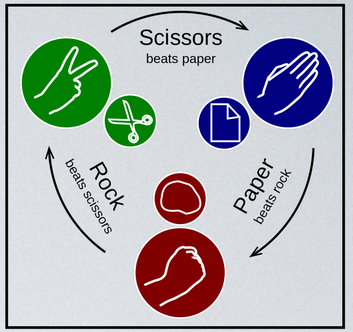
$$\phi(\mathbf{v}, \mathbf{w}) = \mathbf{v}^\top \cdot \begin{pmatrix} 0, & -1 \\ 1, & 0 \end{pmatrix} \cdot \mathbf{w} = v_1 w_2 - v_2 w_1$$







What is Non-Transitivity ?



- Every FFG can be decomposed into two parts

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

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

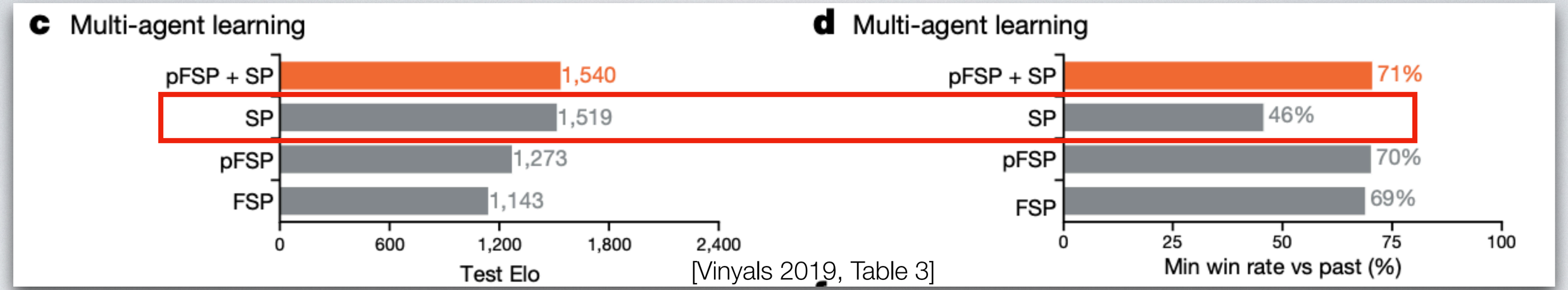
$$v_t \text{ beats } v_{t-1}, \quad v_{t+1} \text{ beats } v_t \nrightarrow v_{t+1} \text{ beats } v_{t-1}$$

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

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

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:

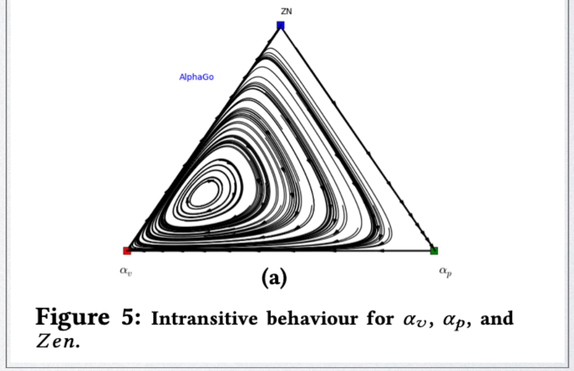


Figure 5: Intransitive behaviour for α_v, α_p, and *Zen*.

[Silver 2016, table 9]

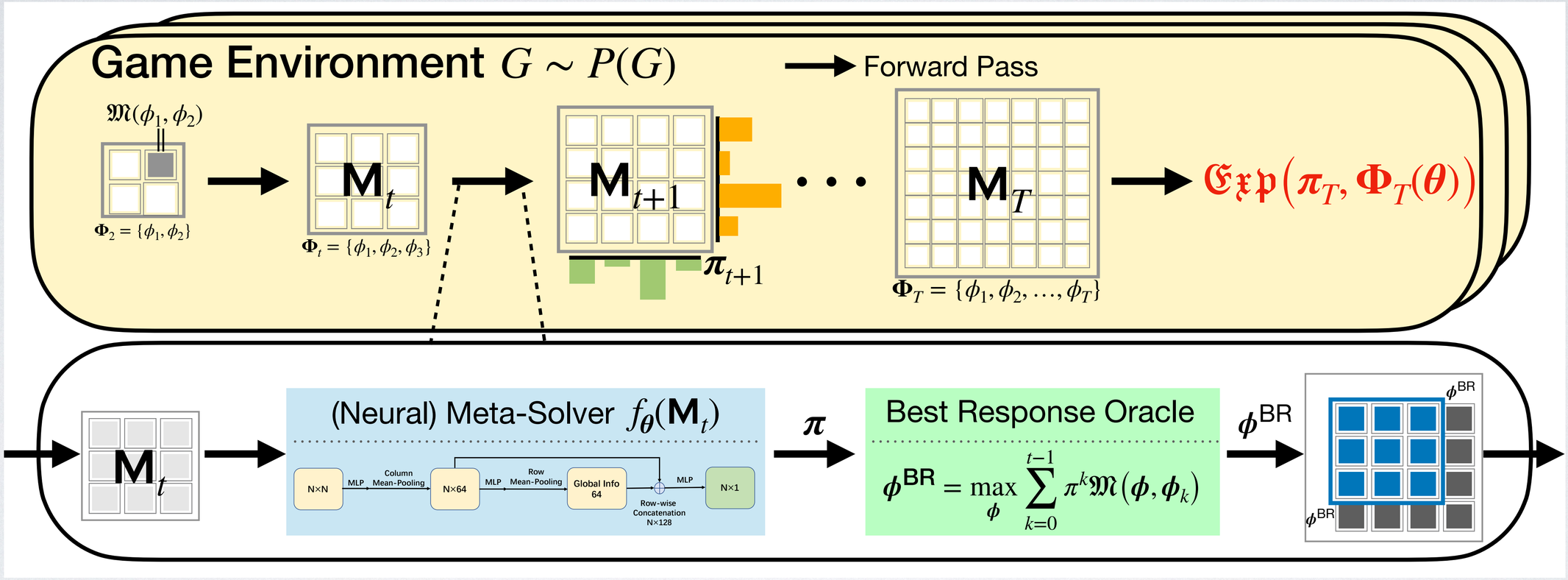
Policy Space Response Oracle: To Build Populations of RL Agents

- 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 $U : \Pi \to \mathbb{R}^n$ 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: $\sigma_{-i} = (0,...,0,0,1)$
  - self-play: $\sigma_{-i} = (0,...,0,1,0)$
  - 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 Π
Compute exp. utilities U^{Π} for each joint $\pi \in \Pi$
Initialize meta-strategies $\sigma_i = \textsc{Uniform}(\Pi_i)$
while *epoch e in* $\{1, 2, \cdots\}$ **do**
  for *player* $i \in [[n]]$ **do**
    for *many episodes* **do**
      select opponent policies: Sample $\pi_{-i} \sim \sigma_{-i}$
      compute the best response: Train oracle π_i' over $\rho \sim (\pi_i', \pi_{-i})$
    augment strategy pool: $\Pi_i = \Pi_i \cup \{\pi_i'\}$
    expand the payoff matrix: Compute missing entries in U^{Π} from Π
    solve the new meta game: Compute a meta-strategy σ from U^{Π}
Output current solution strategy σ_i for player i

PSRO Incorporate Many Variants



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

iterated best response
fictitious play
double oracle/PSRO
PSRO-Nash
α-PSRO
JPSRO

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).

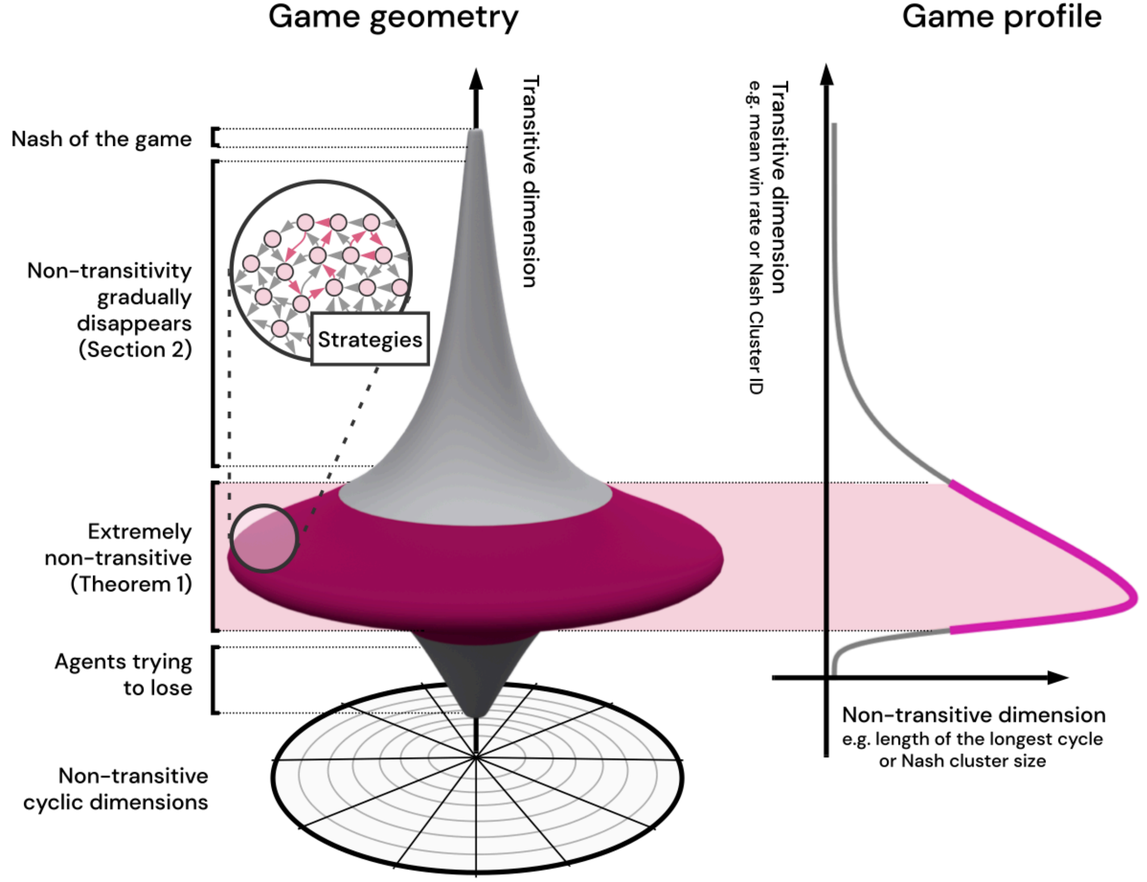
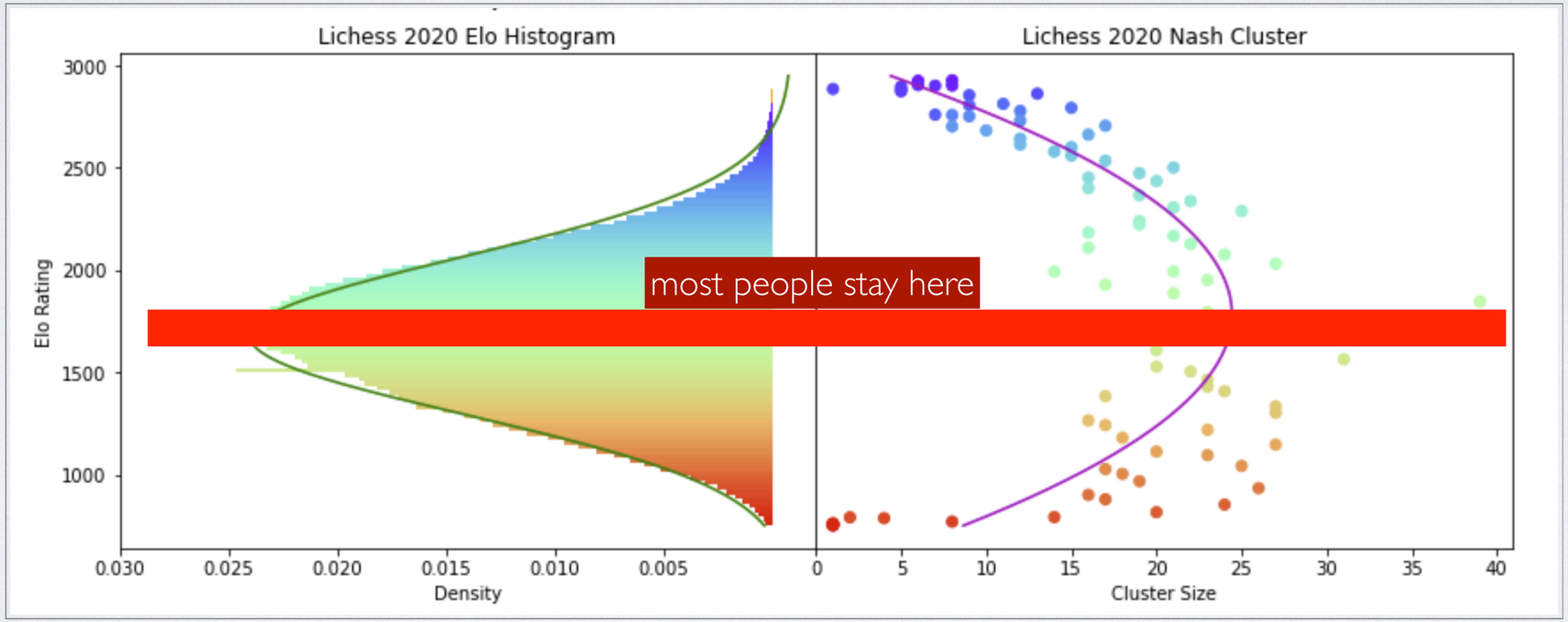


Figure 1: High-level visualisation of the geometry of Games of Skill. It shows a strong transitive dimension, that is accompanied by the highly cyclic dimensions, which gradually diminishes as skill grows towards the Nash Equilibrium (upward), and diminishes as skill evolves towards the worst possible 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

- Real-world data set from human players on Chess
  - previous results are based on AI, now we study 1000 human players from Lichess
  - Chess presents the same spinning top pattern, which verifies the hypothesis

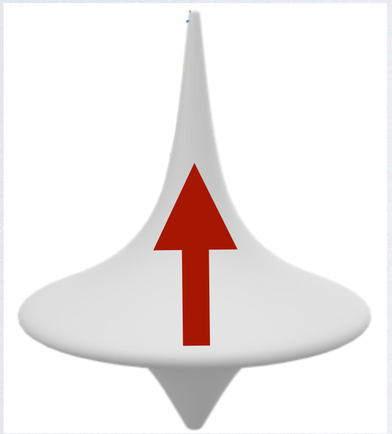


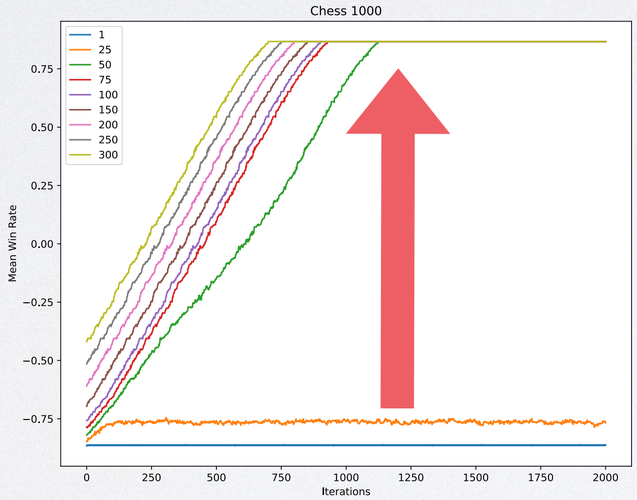
[Ricky Sanjaya]

Why Modelling Diversity is Critical ?

- Diversity matters because the more diverse the strategy pool, the less un-exploitable. Promoting diversity can help you walk out of the in-transitive region faster.

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

- In real-world applications, you want policies to be diverse enough, covering different skill levels. This is a realistic need from autonomous driving and gaming AI applications.

Recent Advance: Diverse PSRO

1. Diversity should include both response diversity (in terms of reward), and behavioural diversity (in terms of policy occupancy measure)

2. We want both the outcomes and the policies that lead to those outcomes to be diverse.

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

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

Recent Advance: Diverse PSRO

1. behavioural diversity: assuming existing population of policy mixed by Nash distribution is $\pi_E = (\pi_i, \pi_{E_{-i}})$, 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:

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

2. in practice, 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 large prediction error (not covered by the existing occupancy measure).

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

Recent Advance: Diverse PSRO

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

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

2. the above equation has no close form, but we can optimise a lower bound

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

3. chicken-egg problem: how can we know the payoff $\mathbf{a}_{M+1}$ before we train the policy ?

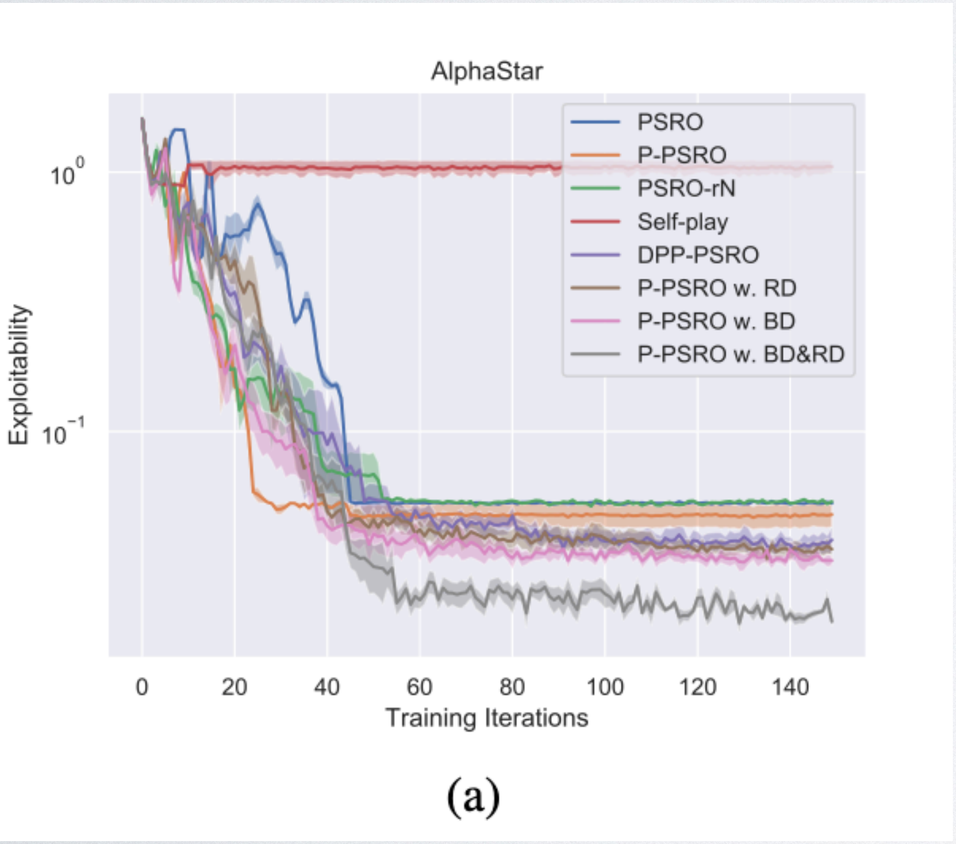
$$\frac{\partial F\left(\pi_i'(\theta)\right)}{\partial \theta} = \left(\frac{\partial \phi_i\left(\pi_i'(\theta), \pi_{-i}^1\right)}{\partial \theta}, \ldots, \frac{\partial \phi_i\left(\pi_i'(\theta), \pi_{-i}^M\right)}{\partial \theta} \right) \frac{\partial F}{\partial \mathbf{a}_{M+1}}$$

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

Recent Advance: Diverse PSRO

1. considering both diversity terms in the PSRO process

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



(a)

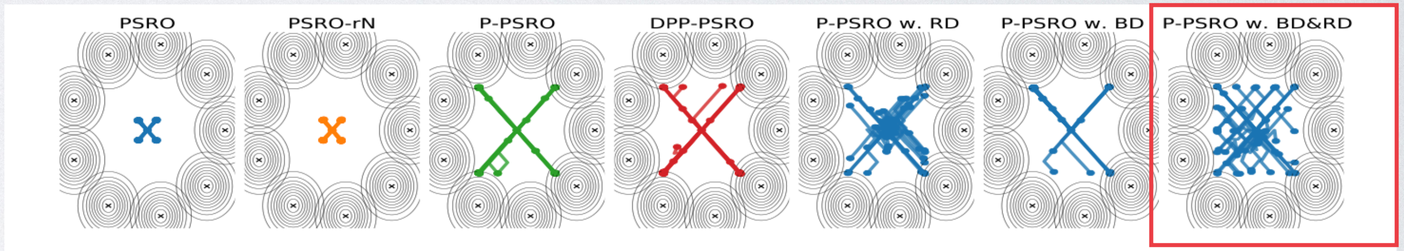


Figure 2: Exploration trajectories during training process on *Non-Transitive Mixture Games*.

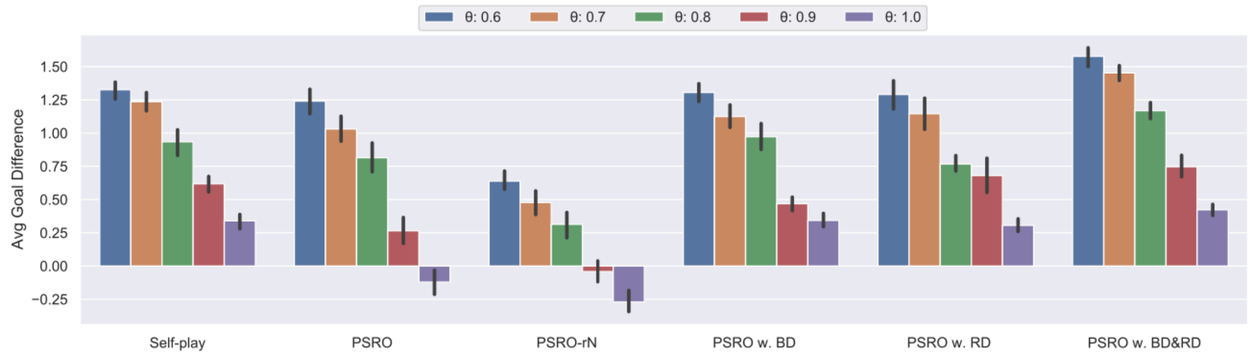


Figure 3: The average goal difference between all the methods and the built-in bots with various difficulty levels θ ($\theta \in [0, 1]$ and larger θ means harder bot) on *Google Research Football*.

Diverse Behaviours Learned on Google Football

https://sites.google.com/view/diverse-psro/

make offside

push and run

Recent Advance: Auto-PSRO

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

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

Ziyu Wan[2], Bo Liu[3], Stephen McAleer[4], Ying Wen[2], Jun Wang[1]

1. Learning to learn: to discover multi-agent algorithms ("who to beat" and "how to beat them") from data.

2. Maybe game theoretical knowledge (transitivity/non-transitivity/Nash) are not necessarily needed, the solution algorithm can be learned purely from data.

3. The idea is to learn how to build an auto-curricula based on the type of game provided to the meta-learning algorithm, rather than what the auto-curricula should be (e.g. PSRO/DO).

4. Why it will work better than DO/PSRO: because RL oracle can only approximate best response, and using Nash, though theoretically guaranteed, may not be the best option for a solver.

5. On single-agent RL, the discovered RL methods are proved to outperform TD learning designed by humans.

Discovering Reinforcement Learning Algorithms

Junhyuk Oh Matteo Hessel Wojciech M. Czarnecki Zhongwen Xu

Hado van Hasselt Satinder Singh David Silver

DeepMind

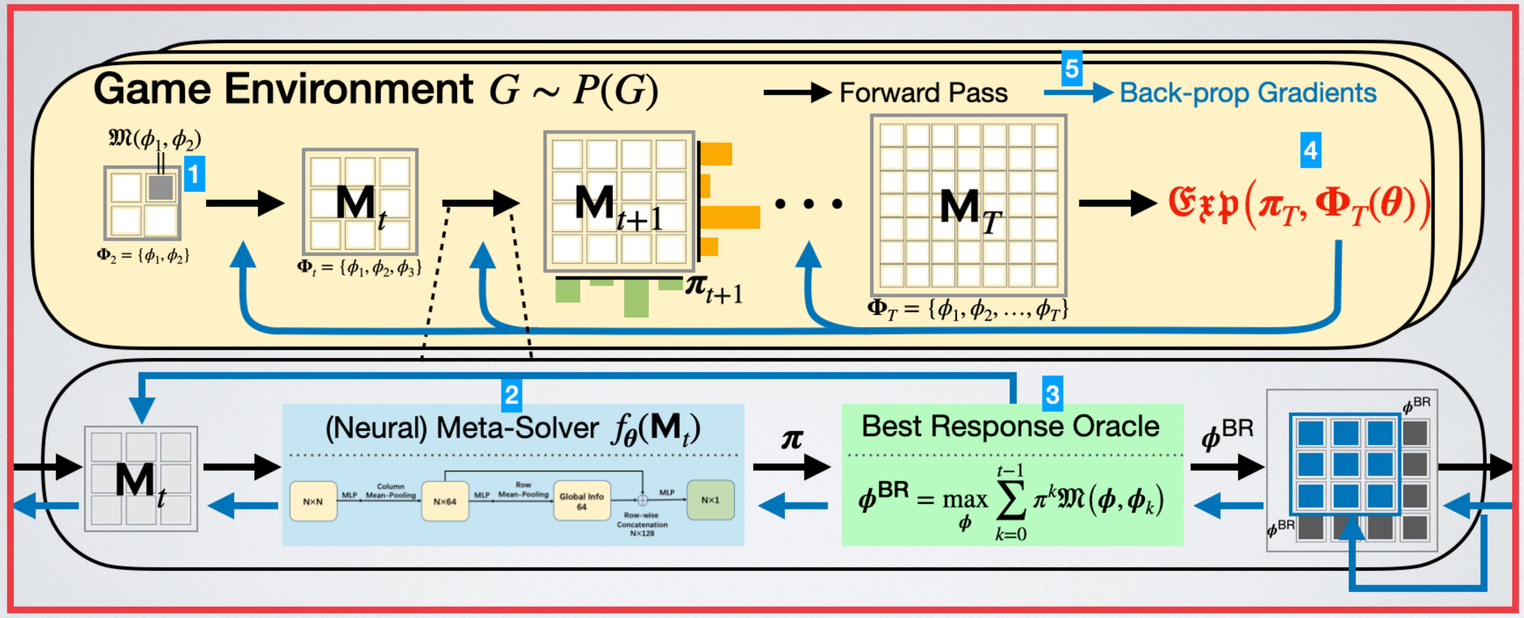
Meta-Gradient Reinforcement Learning with an Objective Discovered Online

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

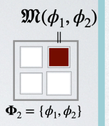
Recent Advance: Auto-PSRO Framework

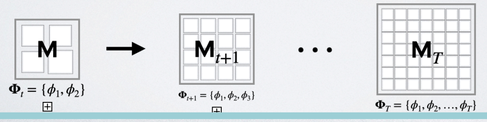


Let's differentiate the whole process using back propagation!

1 The Meta-Game

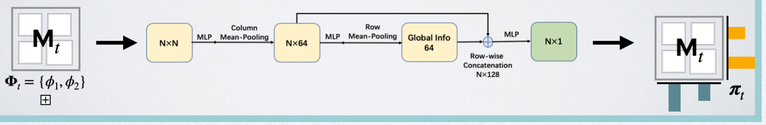
- Main component of population-based methods - **The meta-game**
  - An agent is a mapping $\phi : S \times A \to [0,1]$
  - The payoff for agent *i* vs. agent *j* is defined as $\mathfrak{M}(\phi_i, \phi_j)$
  - Payoff matrix between *agents* in a population amenable to GT analysis
  - The goal of these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





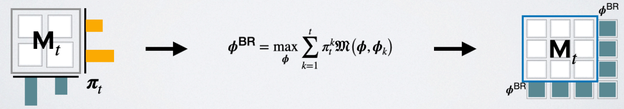
2 The Meta-Solver

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



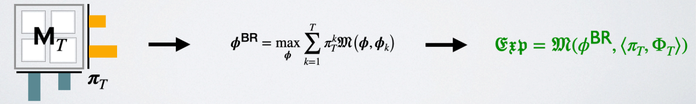
3 The Best-Response Oracle

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



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



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:

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

  - What does the gradient boil down to then?

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

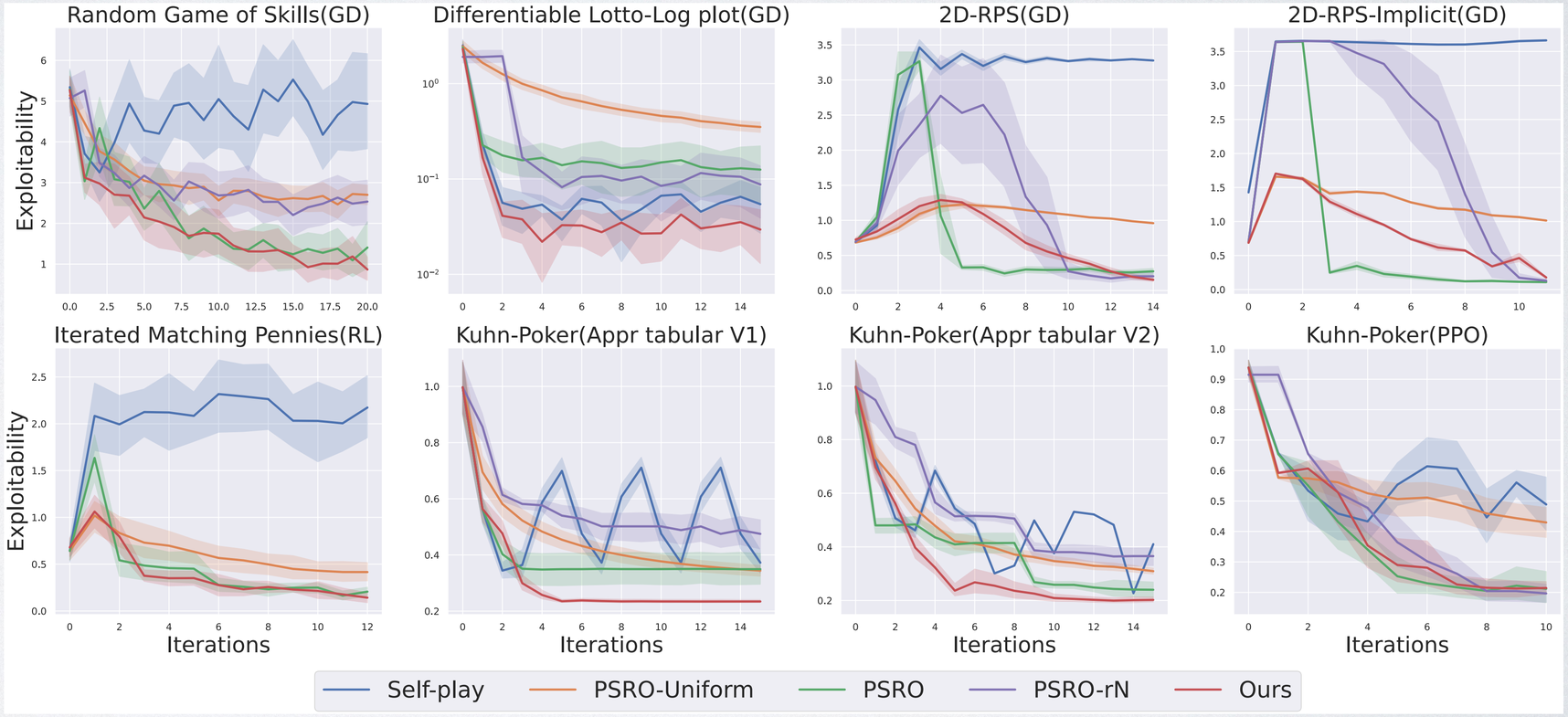
Gradient of most interest decomposes to

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

Recent Advance: Auto-PSRO Result

- **1st question**: 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

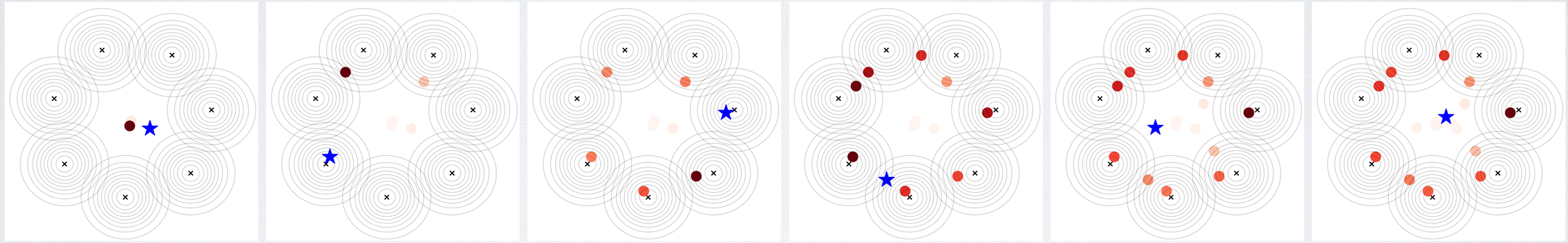
- Performance *at least* as good as baseline measures
- Outperforms PSRO in multiple settings



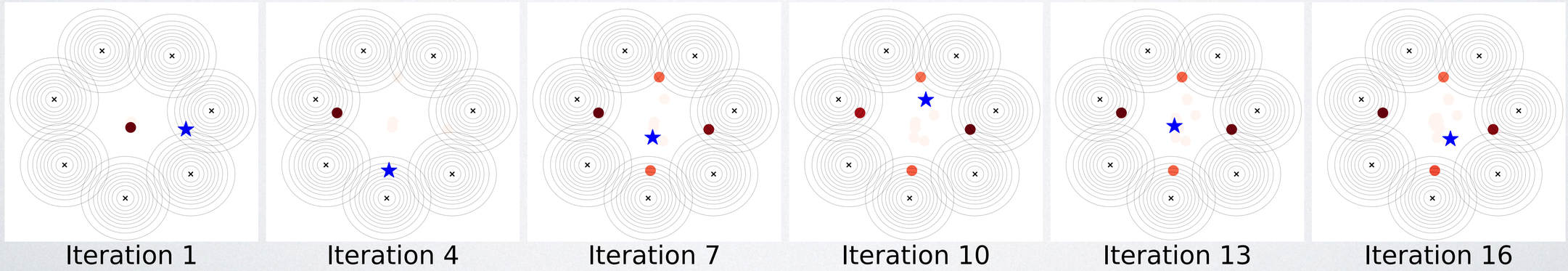
Recent Advance: Auto-PSRO Result

- **2nd question**: What is the learned multi-agent algorithm ?
  - Compare agents found and their respective densities in the meta-distribution

Ours



PSRO



Recent Advance: Auto-PSRO Result

- **3rd question**: Can the learned solver generalise over different games?
  - the most promising and striking aspect of LMAC - Train on small games and generalise to large game, e.g., train on Kukn Poker and test on Leduc Poker

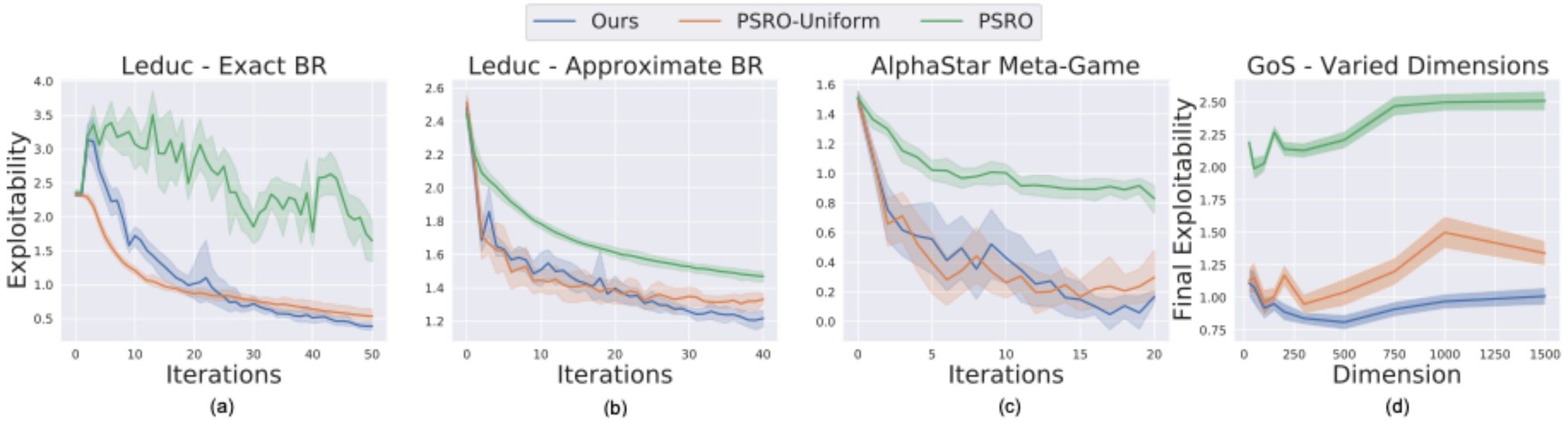


Figure 5: (a) Exploitability when trained on Kuhn Poker with an exact tabular BR oracle using ES-LMAC and tested on Leduc Poker. (b) Same as (a) with approximate tabular BR V2 (c) Exploitability when trained on GoS with a GD oracle and tested on the AlphaStar meta-game from [8] (d) Final exploitability when trained on 200 Dimension GoS and tested on a variety of dimension size GoS.

Additional Resources:

- If you want to know more details about PSRO and its variations, please refer to
  - Talk: https://www.bilibili.com/video/av969218959/
  - Slides: https://rlchina.org/lectures/lecture11.pdf
- A self-contained MARL survey from game theoretical perspective:
  - https://arxiv.org/abs/2011.00583
- If you want to get hands on to solving some two-player zero-sum games, e.g., Poker/Chess
  - https://arxiv.org/pdf/2103.00187.pdf
  - https://github.com/aicenter/openspiel_reproductions

Thanks!

Any Questions?