Reinforcement Learning China Summer School

RLChina 2020

Advances of Multi-agent Learning (in Gaming AI)

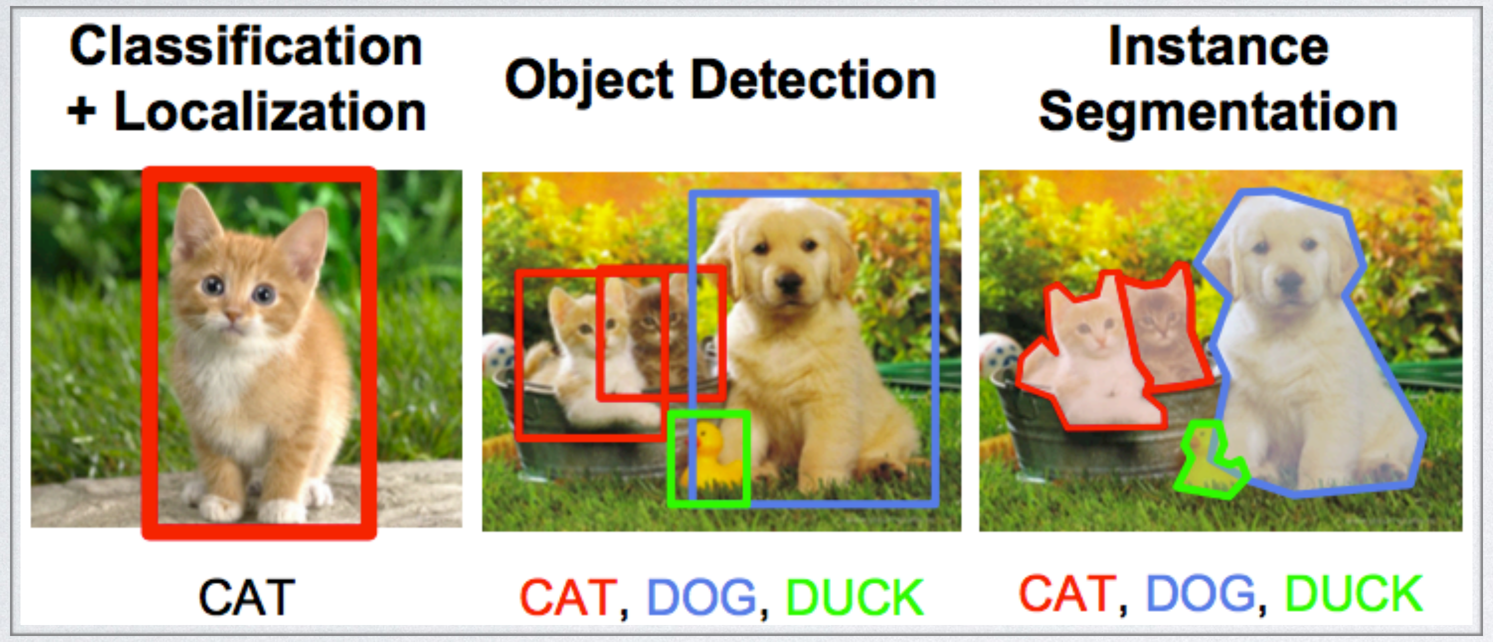
Yaodong Yang
www.yangyaodong.com
Huawei R&D UK
University College London

August 2020

Contents

Why Multi-agent Learning ?

- Reinforcement learning turns data/knowledge into closed-loop decision making.
- Multi-agent learning deal with interactions among the learning agents.

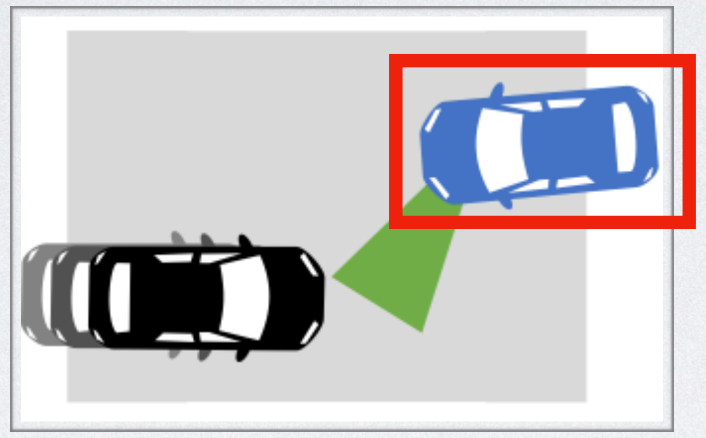


Data
↓
Knowledge

Data
↓
Knowledge
↓ interactions happen!
Decisions

Multi-agent Learning for Autonomous Driving

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



scenario

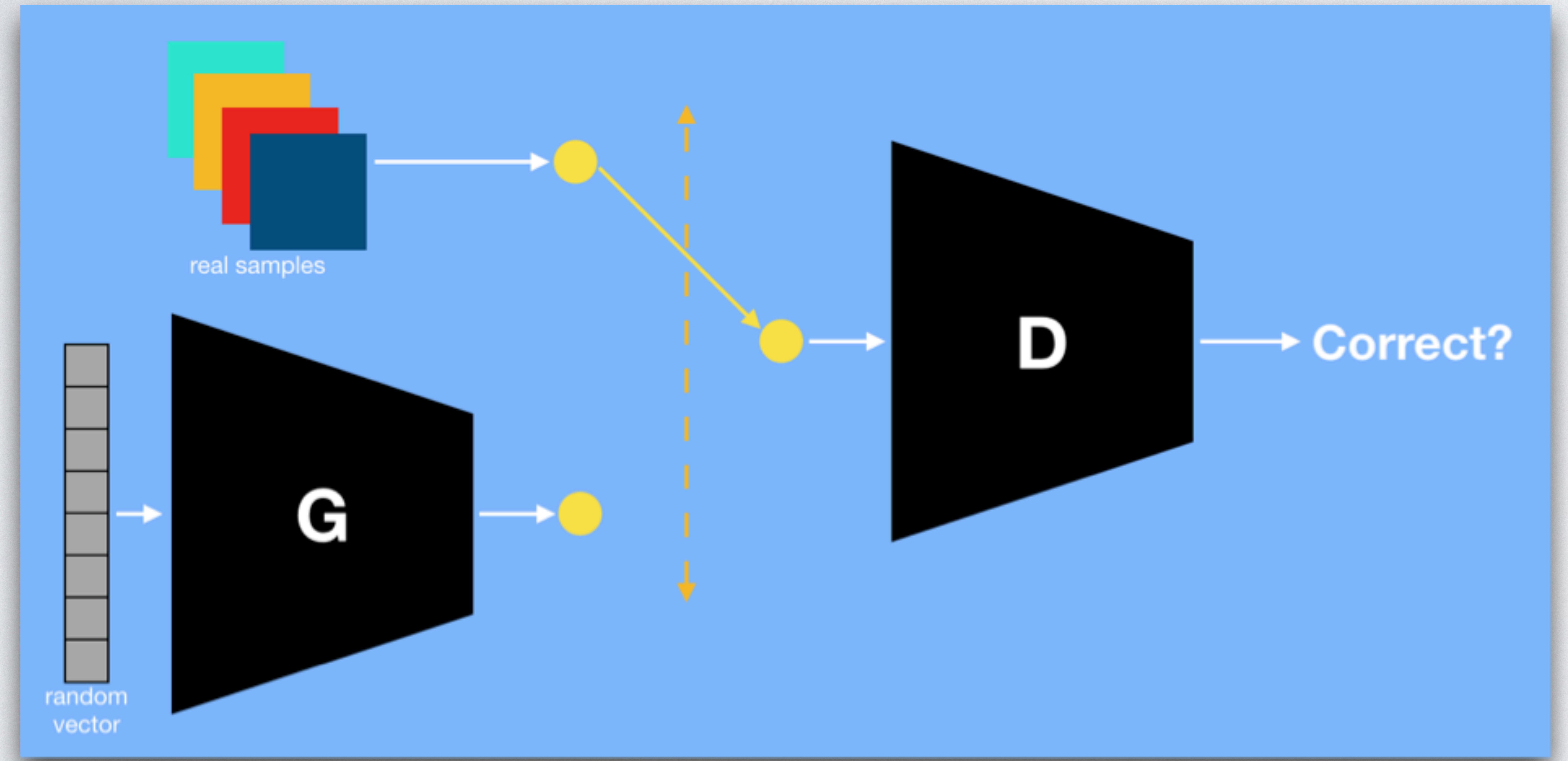
	Yield	Rush
Yield	(0, 0)	(1, 2)
Rush	(2, 1)	(0, 0)

normal-form game

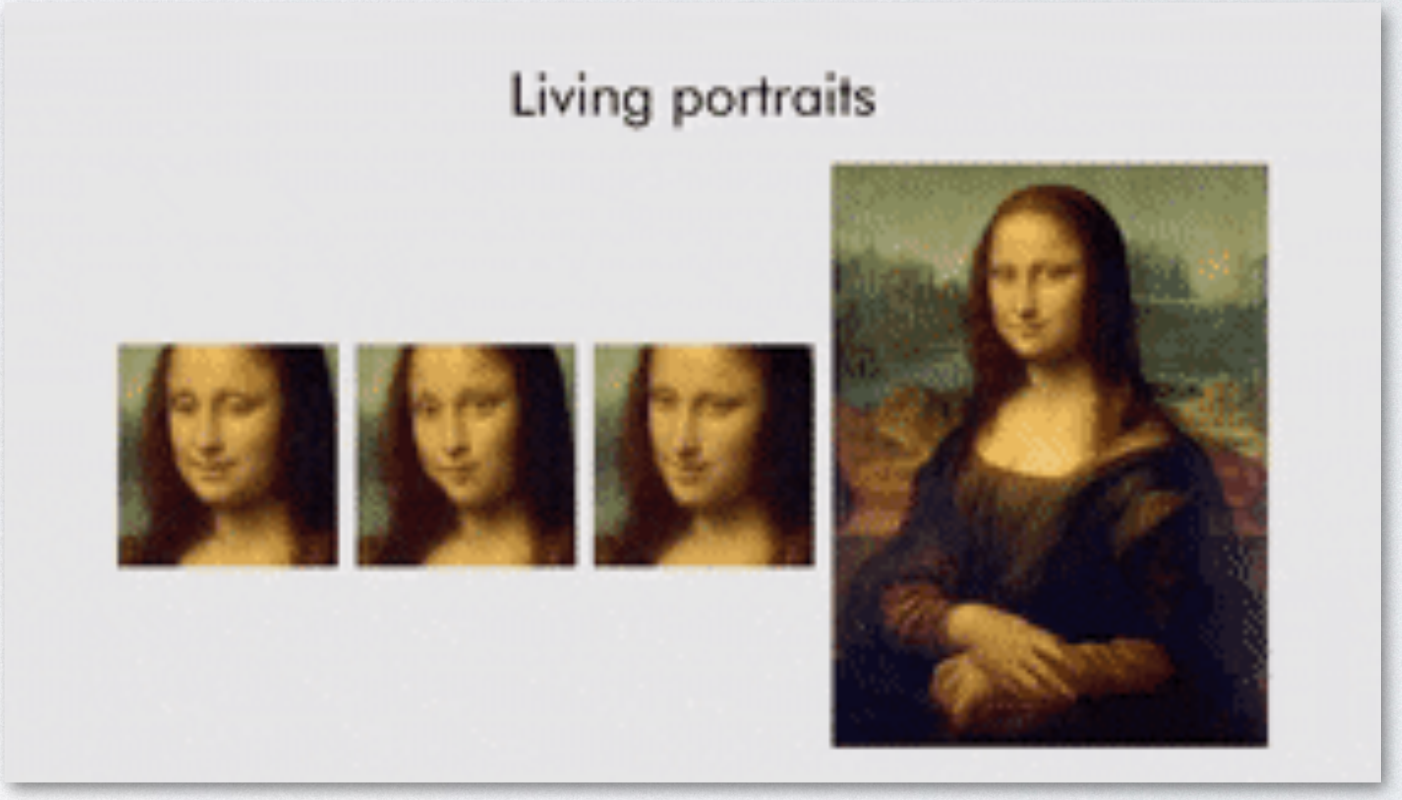
- When the drivers are rational, they will reach the outcome of a Nash Equilibrium. It is the outcome of interaction. Knowing it can predict future.
- Real-world decision making has cooperation & competition. For each agent, how to infer the belief of the other agents and make the optimal action is critical.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

Multi-agent Learning for Machine Learning

Two-player zero-sum game → Generative Adversarial Network



CycleGANs



StyleGAN

player 1 | player 2

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbf{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbf{E}_{z \sim p(z)} \log\left(1 - D_{\theta_d}\left(G_{\theta_g}(z)\right)\right) \right]$$

Problem Formulation: Singe-agent Reinforcement Learning

- Learn the optimal behaviour through trial-and-errors from the environment.
- Modelled by a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{P}_0, \gamma)$
  - $\mathcal{S}$ denotes the state space,
  - $\mathcal{A}$ is the action space,
  - $\mathcal{R} = \mathcal{R}(s, a)$ is the reward function,
  - $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the state transition function,
  - $\mathcal{P}_0$ is the distribution of the initial state, γ is a discount factor.
- The goal is to find the optimal policy π that maximises expected reward:
  - Discounted reward:

$$V_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbf{E}_{\pi,\mathcal{P}} \left\{ R_t \,|\, s_0 = s, \pi \right\}$$

  - Time-average reward:

$$V_\pi(s) = \lim_{T\to\infty} \sum_{t=0}^{T} \frac{1}{T} \mathbf{E}_{\pi,\mathcal{P}} \left\{ R_t \,|\, s_0 = s, \pi \right\}$$

Solution to Single-Agent RL

- Value-based method (**learn the Q-function** $Q(s,a) = r^j(s,a) + \gamma \mathbf{E}_{s' \sim p}[v_\pi(s')]$):

$$Q^{\text{new}}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{R_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \Bigg)}^{\text{temporal difference}}$$

$$\mathcal{H}Q(s,a) = \mathbf{E}_{s'}\left(R(s,a) + \gamma \max_b Q\left(s', b\right) \right)$$ is a contraction-mapping operator.

- Policy-based method (**learn the policy** $\pi_\theta(\cdot \mid s_t)$ **parameterised by** θ):

$$J(\theta) = \sum_{s \in S} d^\pi(s) V^\pi(s) = \sum_{s \in S} d^\pi(s) \sum_{a \in \mathcal{A}} \pi_\theta(a \mid s) Q^\pi(s,a), \quad d^\pi(s) = \lim_{t \to \infty} \mathcal{P}\left(s_t = s \mid s_0, \pi_\theta\right)$$

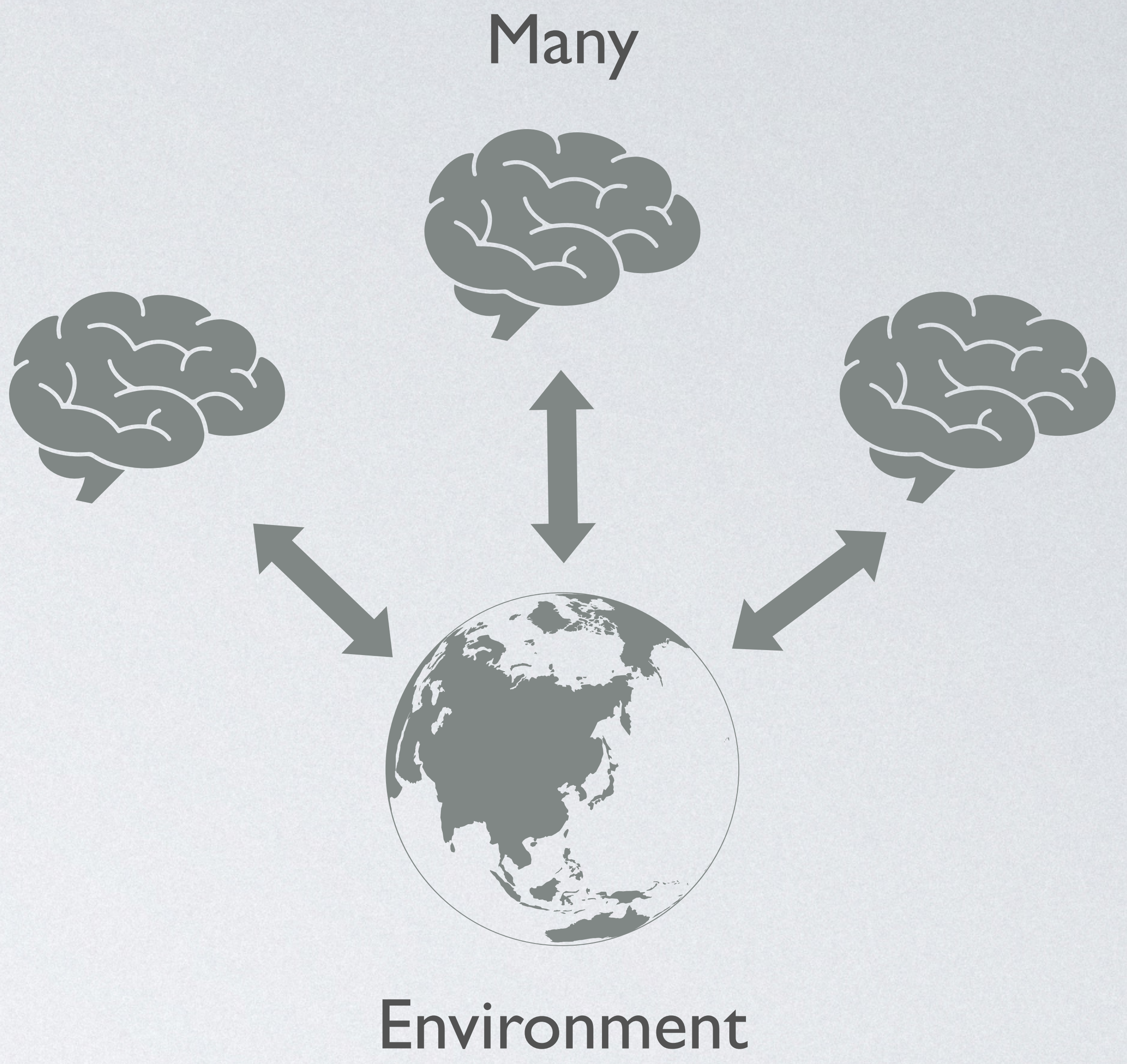
Occupancy measure on state induced by following π_θ in the MDP

$$\Delta\theta \propto \nabla_\theta J(\pi) = \mathbf{E}_{s,a}\left[\nabla_\theta \log \pi(s,a) \cdot Q^\pi(s,a)\right]$$

Push the parameters towards the direction where the reward is large

Problem Formulation: Multi-agent Reinforcement Learning

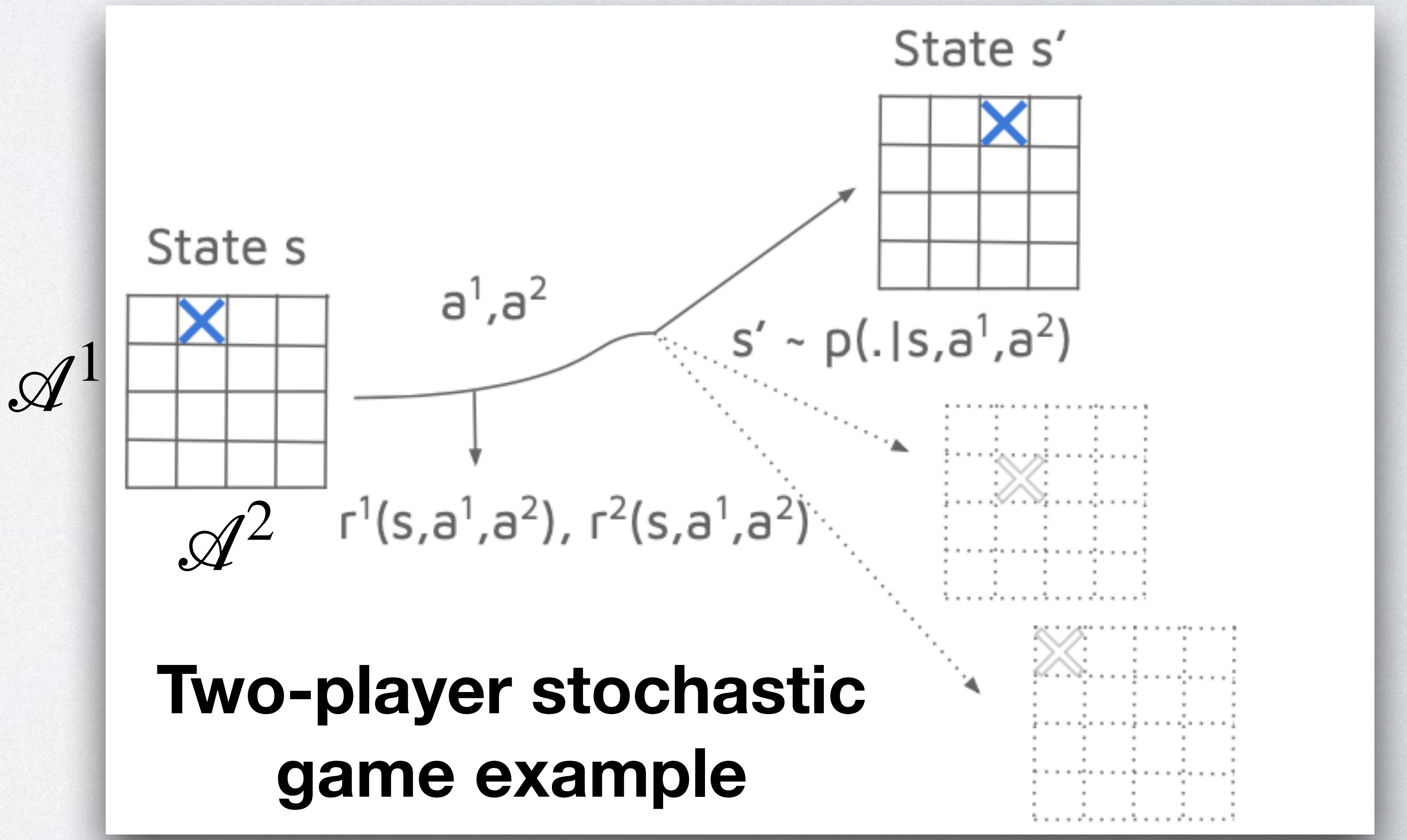
- Modelled by a Stochastic Game $(\mathscr{S}, \mathscr{A}^{\{1,\ldots,n\}}, \mathscr{R}^{\{1,\ldots,n\}}, \mathscr{T}, \mathscr{P}_0, \gamma)$
  - $\mathscr{S}$ denotes the state space,
  - $\mathscr{A}$ is the joint-action space $\mathscr{A}^1 \times \ldots \times \mathscr{A}^n$,
  - $\mathscr{R}^i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



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

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

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



Two-player stochastic game example

Solution to Multi-Agent RL

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

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

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

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

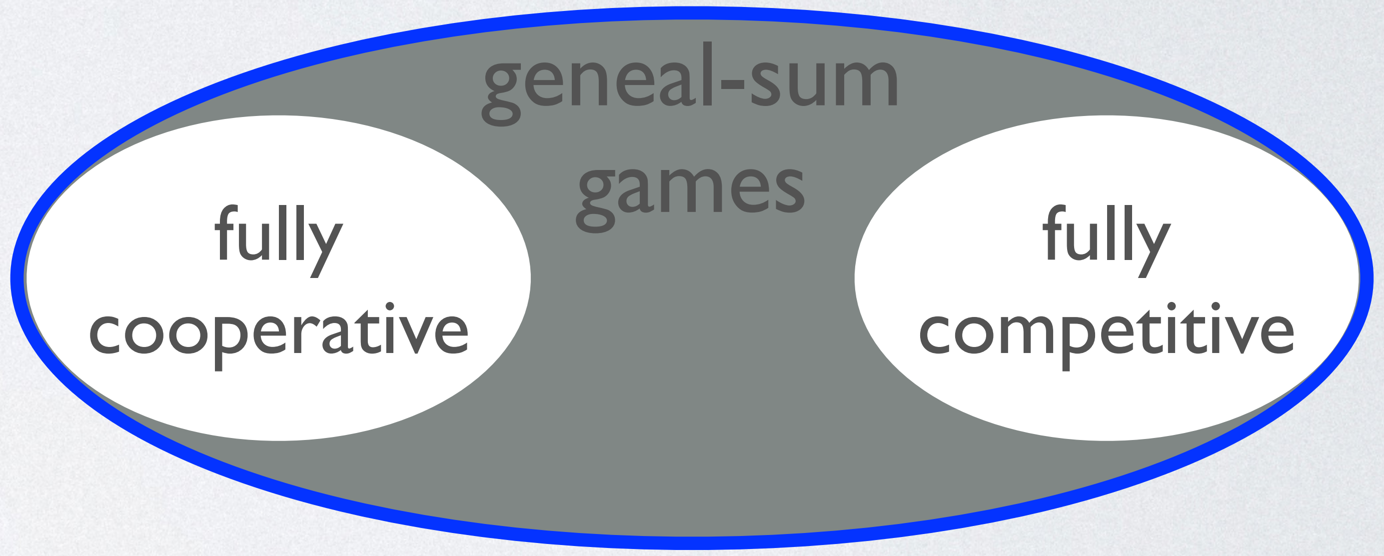
$$\textbf{eval}_i\left\{Q_{\cdot,t}(s_{t+1}, \cdot\,)\right\} = \max_{\boldsymbol{a}} Q_{i,t}(s_{t+1}, \boldsymbol{a})$$

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

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

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

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



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

The Sense of Optimality in a Multi-Agent System

Unlike single-agent RL, "optimality" has many definitions in a multi-agent system:
☐ *minimal regret*, ☐ *Stackelberg equilibrium*, ☐ *evolutionary stable strategy*, ☐ *correlated equilibrium*, ☐ *Pareto optimal*, ■ *Nash equilibrium, etc.*

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

Definition 2 (Nash Equilibrium)

For a stochastic game, a Nash equilibrium is a collection of policies, one for each player, π^i, such that,

$$\pi^i \in \mathrm{BR}^i(\pi^{-i}).$$

So, no player can do better by changing policies given that the other players continue to follow the equilibrium policy.

Solution to Multi-Agent RL

- Value-based method:

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

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

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

    1. Solve the Nash Equilibrium for the current stage game

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

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

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

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

Solution to Multi-Agent RL

- Policy-based method (objective $J(\boldsymbol{\theta}) = \mathbf{E}_{s\sim P,\boldsymbol{a}\sim\boldsymbol{\pi}}[\sum_{i=1}^{N} R_i(s,\boldsymbol{a})]$):
  - Stochastic policy gradient:

$$\nabla_{\theta_i} J(\theta_i) = \mathbf{E}_{s\sim\mathscr{P},\mathbf{a}\sim\boldsymbol{\pi}}\left[\nabla_{\theta_i}\log\pi_i(a_i|s_i)\, Q_i^{\boldsymbol{\pi}}(s,a_i,\boldsymbol{a_{-i}})\right]$$

  - Deterministic policy gradient:

$$\nabla_{\theta_i} J(\theta_i) = \mathbf{E}_{s,\mathbf{a}}\left[\nabla_{\theta_i}\pi_i(a_i|s_i)\,\nabla_{a_i} Q_i^{\boldsymbol{\pi}}(s,a_i,\boldsymbol{a_{-i}})\Big|_{a_i=\pi_i(s_i)}\right]$$

  - Centralised training with decentralised execution methods further learn critics in a centralised way.

$$\mathscr{L}(\phi_i) = \mathbf{E}_{s,\mathbf{a},r,s'}\left[\left(Q_{\phi_i}^{\boldsymbol{\pi}}(s,a_i,\boldsymbol{a_{-i}}) - y\right)^2\right], \quad y = R_i + \gamma Q_{\phi_i}^{\boldsymbol{\pi}'}(s,a_i',\boldsymbol{a'_{-i}})\Big|_{a_j'=\boldsymbol{\pi}_j'(s_j)}$$

  - Yet, PG methods have no theoretical guarantee in even linear-quadratic games [Mazumdar 2019].

Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☐ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☐ Motivation of studying games
  - ☐ When self-play does not work
  - ☐ The landscape of real-world games
  - ☐ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☐ Elo rating
  - ☐ Nash Equilibrium
  - ☐ Replicator dynamics
  - ☐ α-Rank & α^{α}-Rank
- **Policy Improvement in Meta-games**
  - ☐ Iterated self-plays
  - ☐ Fictitious play & generalised weaken fictitious play
  - ☐ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

Tractability of Multi-agent Learning

- 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]

  - 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.
    - It holds for two-player symmetrical game with finite time length.

Tractability of Multi-agent Learning

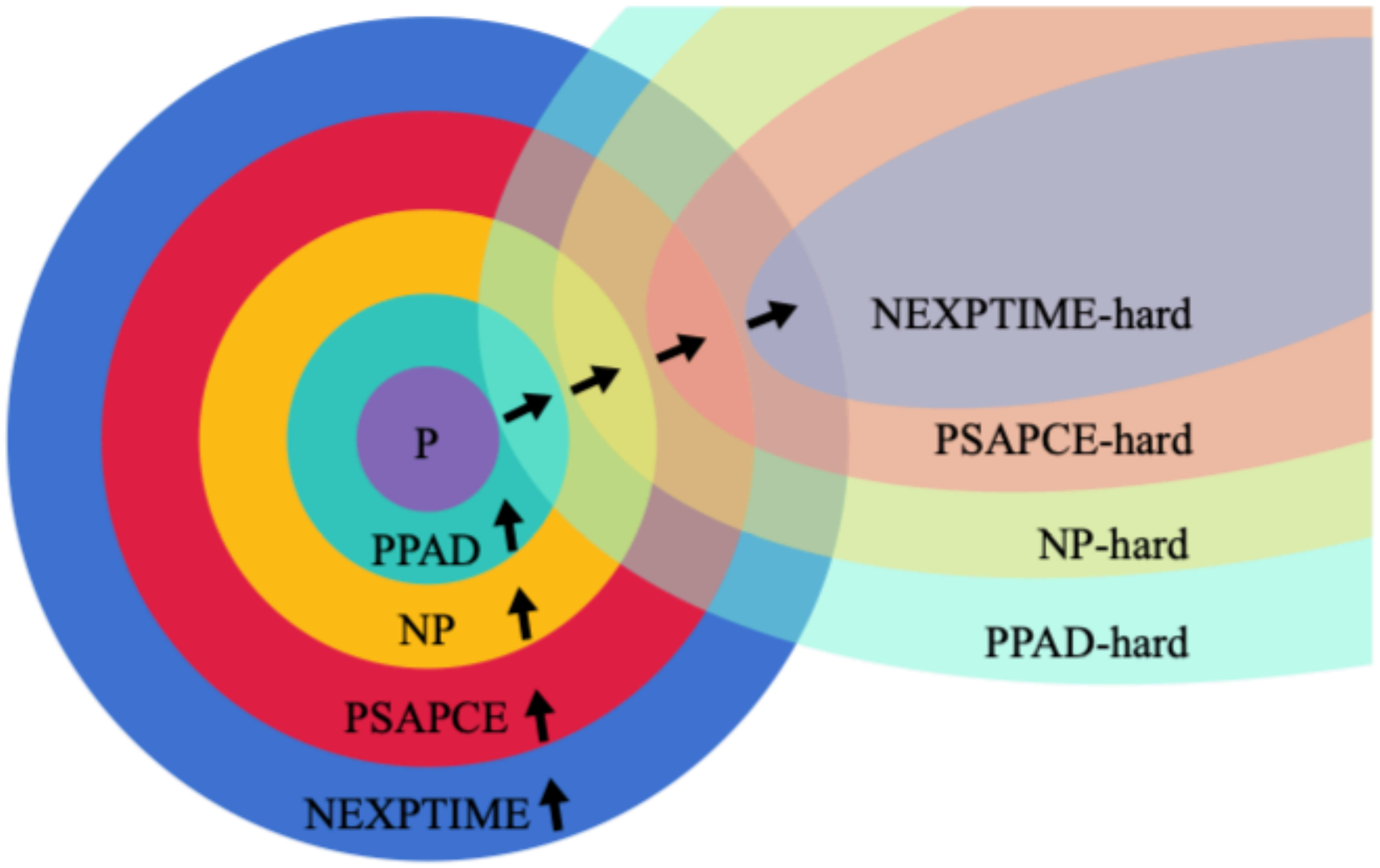
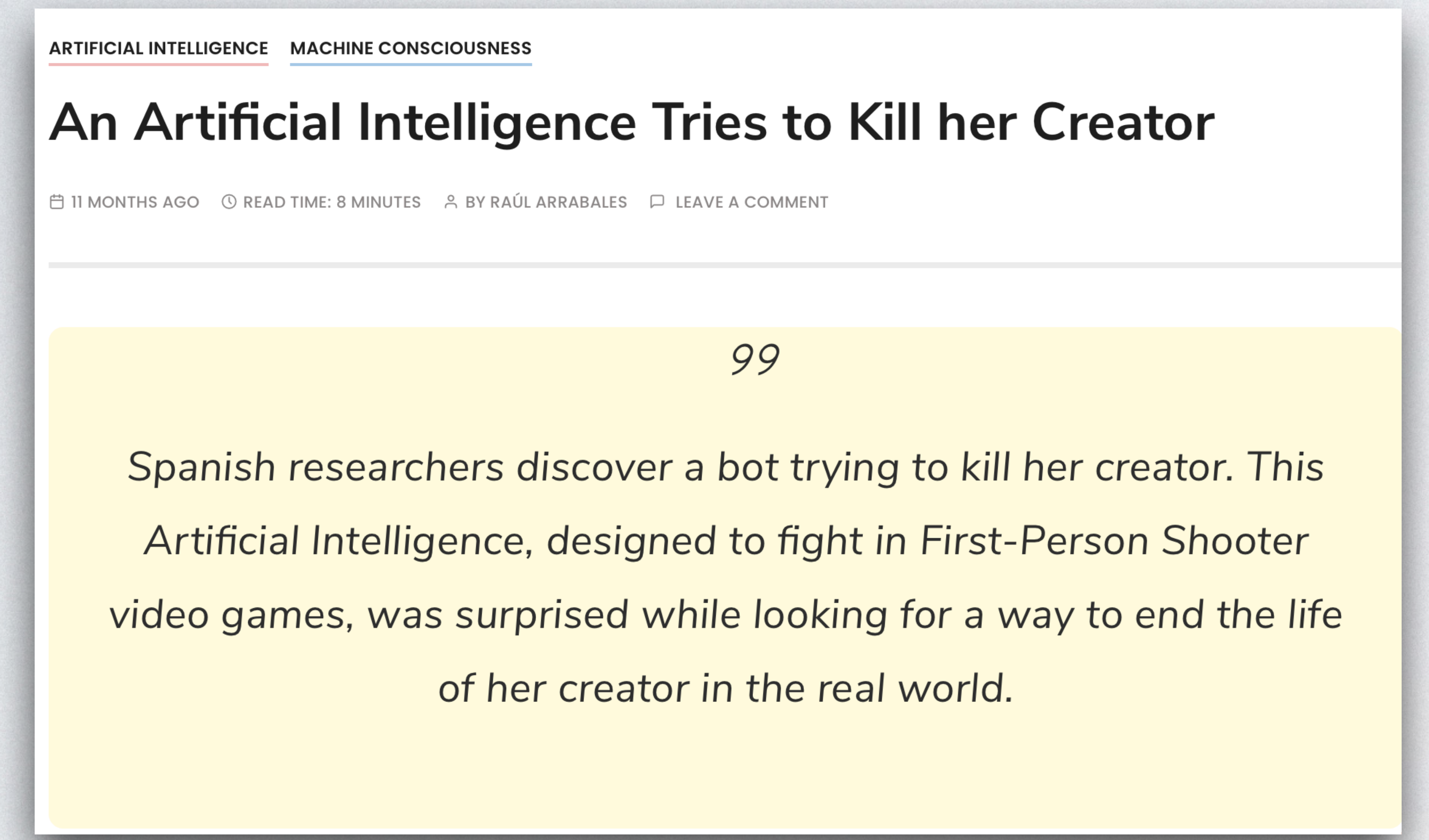


Figure 1.5: Landscape of different complexity classes. Relevant examples are: 1) solving NE in two-player zero-sum game is *P* (Neumann, 1928). 2) solving NE in two-player general-sum game is *PPAD*-hard (Daskalakis et al., 2009). solving NE in three-player zero-sum game is also *PPAD*-hard (Daskalakis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

As a result ….

what you Mum thinks


ARTIFICIAL INTELLIGENCE MACHINE CONSCIOUSNESS

An Artificial Intelligence Tries to Kill her Creator

11 MONTHS AGO · READ TIME: 8 MINUTES · BY RAÚL ARRABALES · LEAVE A COMMENT

99

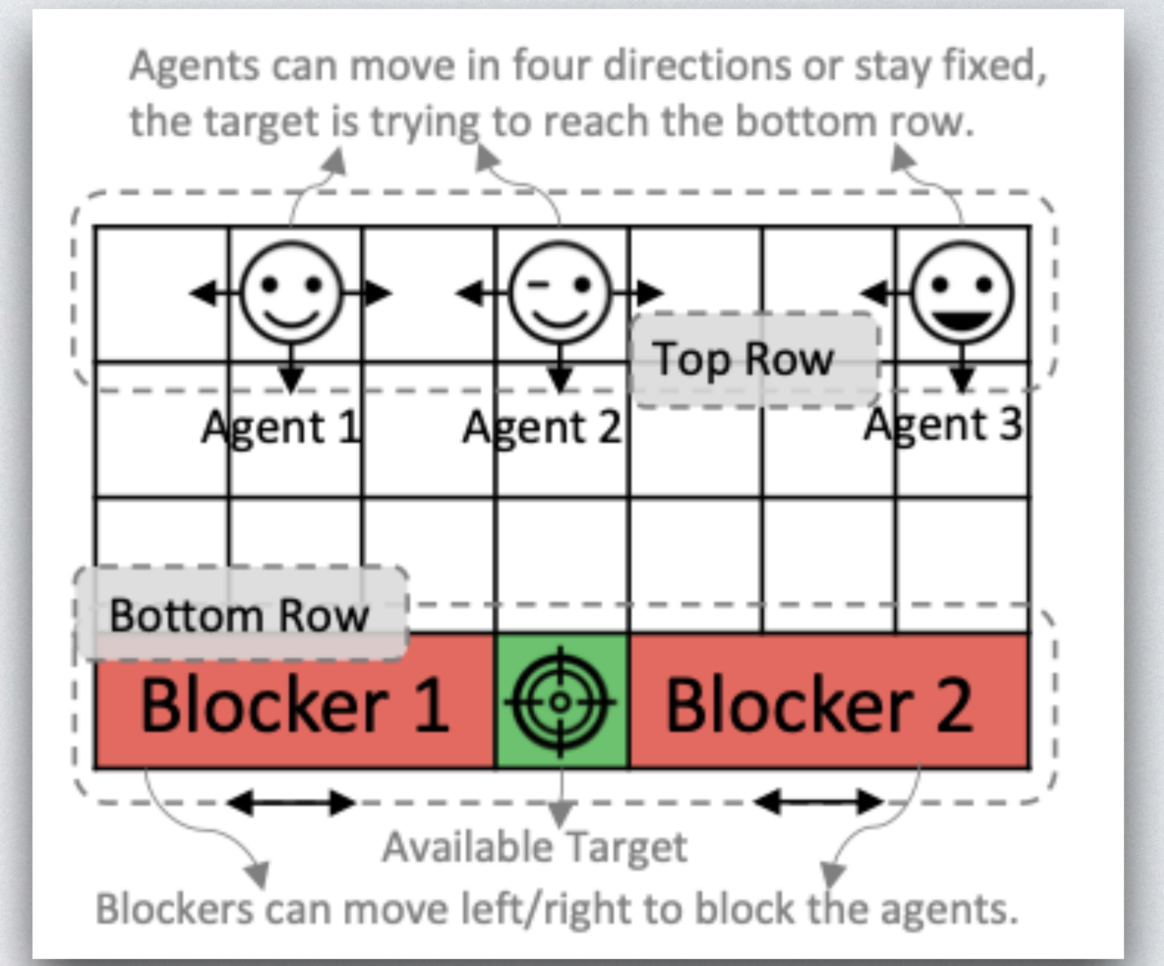
Spanish researchers discover a bot trying to kill her creator. This Artificial Intelligence, designed to fight in First-Person Shooter video games, was surprised while looking for a way to end the life of her creator in the real world.

Something undescribable :)

what you think you are doing

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

what you are actually doing



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

As a result ….

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Artificial Intelligence 171 (2007) 365–377

Artificial Intelligence

www.elsevier.com/locate/artint

If multi-agent learning is the answer, what is the question?

Yoav Shoham *, Rob Powers, Trond Grenager

Department of Computer Science, Stanford University, Stanford, CA 94305, USA

Received 8 November 2005; received in revised form 14 February 2006; accepted 16 February 2006

Available online 30 March 2007

"For the field to advance one cannot simply define arbitrary learning strategies, and analyse whether the resulting dynamics converge in certain cases to a Nash equilibrium or some other solution concept of the stage game. This in and of itself is not well motivated."

As a result ….

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Artificial Intelligence 171 (2007) 365–377

Artificial Intelligence

www.elsevier.com/locate/artint

If multi-agent learning is the answer, what is the question?

Yoav Shoham *, Rob Powers, Trond Grenager

Department of Computer Science, Stanford University, Stanford, CA 94305, USA

Received 8 November 2005; received in revised form 14 February 2006; accepted 16 February 2006

Available online 30 March 2007

"So, what is the question?" I believe is gaming AI, but at a meta-game level!

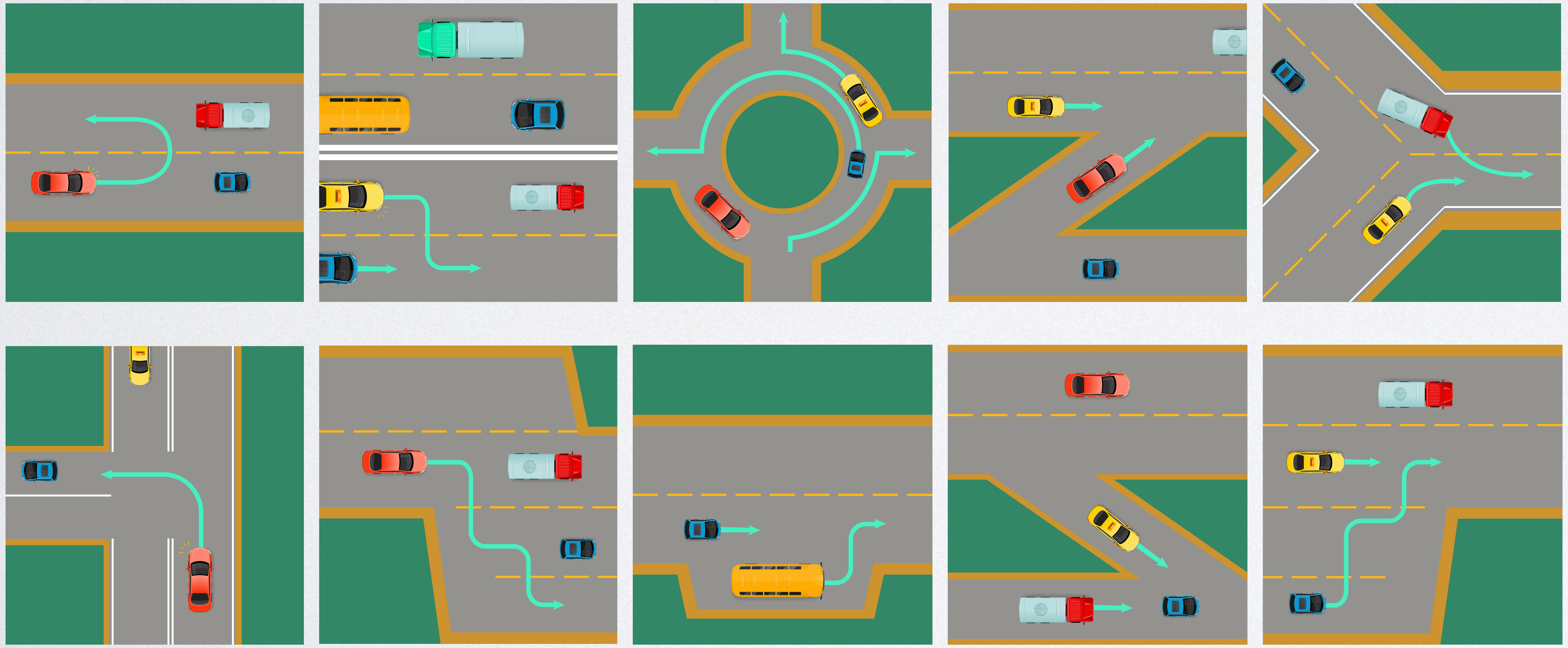
Contents

Why Focus on Gaming AI ?

- "Drosophila" to genetics is what "games" to AI research.
  - Games drives the research of AI frontiers.
  - Simple rules but with deep concepts.
  - Designing winning strategies are intriguing, thousands of years of history.
  - Microeconomic encapsulates real world business, e.g., energy system, auction system, Uber order-dispatching.
- Games is a multi-agent system with co-evolution learners.
  - Great place for landing multi-agent reinforcement learning techniques.
- Games are fun by itself, and gaming business is a cash cow for making profits.

Why Focus on Gaming AI ?

- Autonomous driving is a "game" at the behavioural selection level.
  - The Behaviour Selector subsystem is responsible for choosing the current driving behaviour, such as **lane changing/keeping, intersection handling, traffic light handling, etc.**



[SMARTS autonomous driving simulator, Huawei]

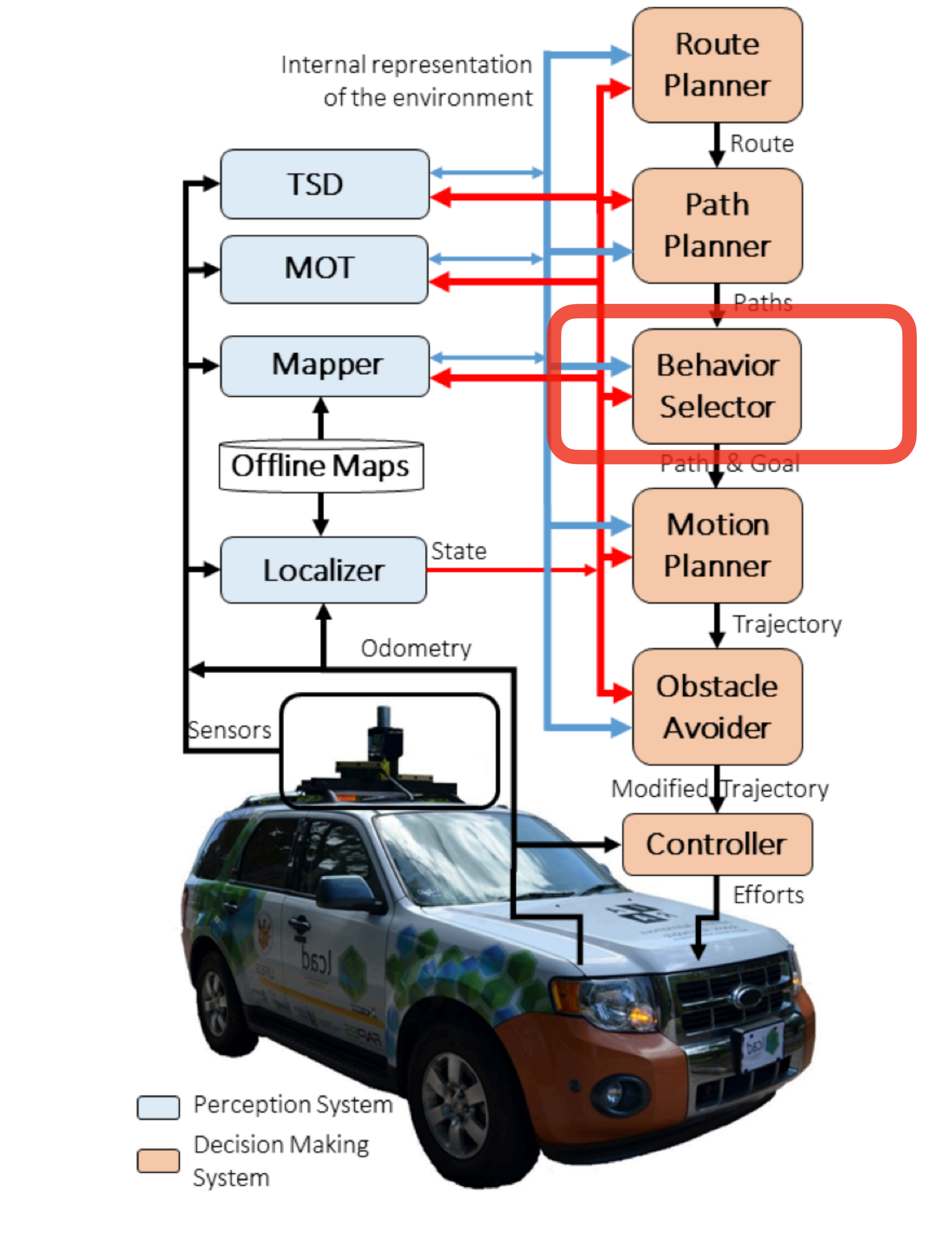


Figure 1: Overview of the typical hierarchical architecture of self-driving cars. TSD denotes Traffic Signalization Detection and MOT, Moving Objects Tracking.

[Badue et. al 2019]

Why Zero-sum Games in Particular ?

- **Many questions in machine learning itself are inherently zero-sum.**
  - Training GANs.
  - All kinds of Poker games, chess, GO, stock market, etc.
  - The idea of maximising the worst-case scenario, i.e., robustness.

- **Two-player Zero-sum games in tabular case has solution.**
  - There are many ways to solve a two-player zero-sum games, e.g., LP, minimising regret.
  - In many-player case, there exists standard evaluation algorithms, e.g., NashConv / exploitability.
- **There are still a lot of very hard open-questions in the zero-sum games.**
  - For example, how to find a saddle point in non-convex non-concave setting. This in turn can help better understand the tools we are developing in the deep learning era.

Multi-agent Learning for Gaming AI

Great advantages have been made in 2019!

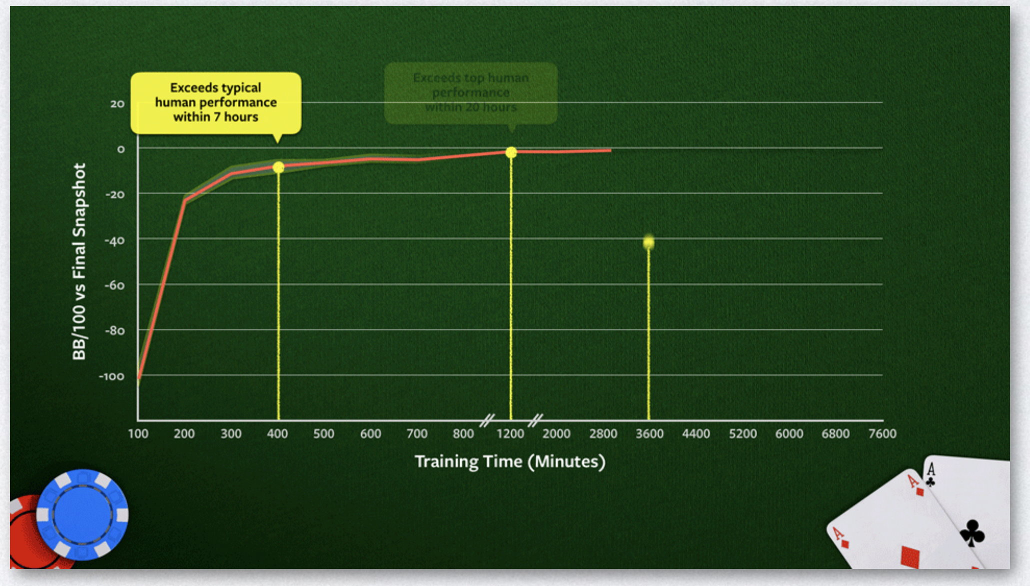
Jan 2016 — Dec 2017 — July 2018 — Jan 2019 — Apr 2019 — July 2019 — Sep 2019

AlphaGO Series

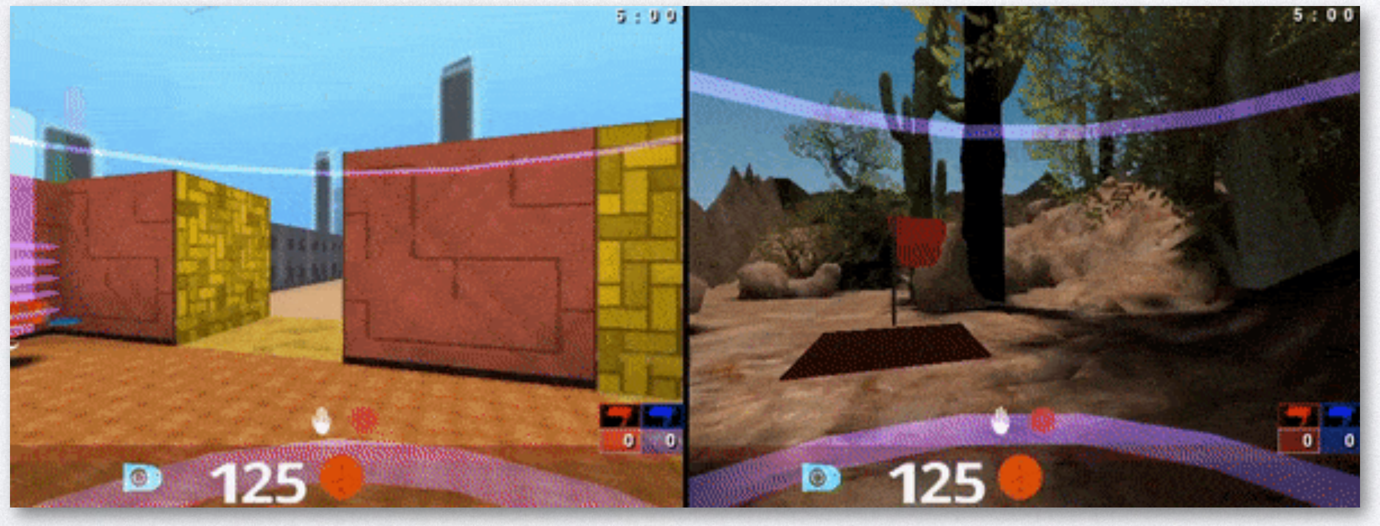
technique of single-agent decision-making is mature

AlphaStar (DeepMind)

Pluribus Poker (FAIR)

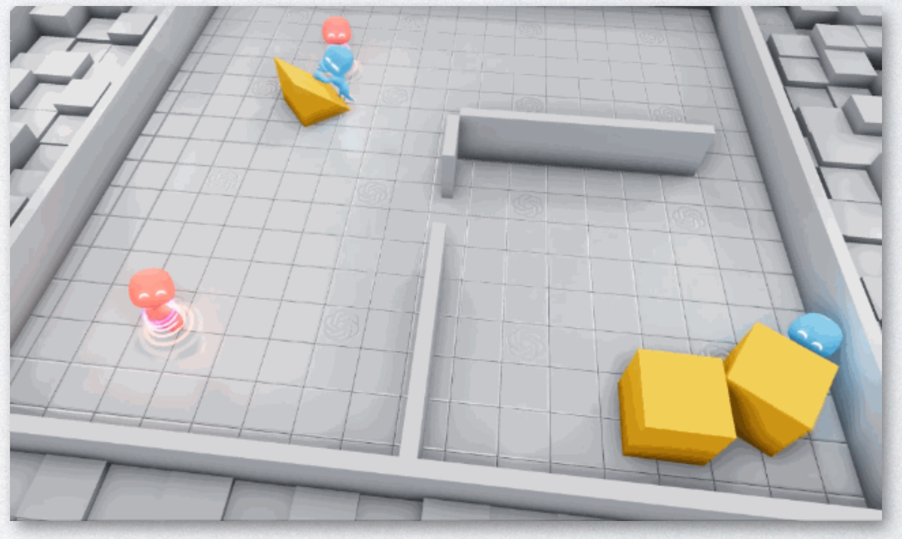


Capture-the-flag (DeepMind)



Dota2 (OpenAI)

Hide and Seek (OpenAI)



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

Our Goal: to find some good policies that can solve the game

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

Black-box multi-agent game engine

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

input

Our algorithm:

Multi-agent policy evaluation

Multi-agent policy improvement

output

"good" strategy $(\pi^{1,*}, \ldots, \pi^{N,*})$

Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☐ When self-play does not work
  - ☐ The landscape of real-world games
  - ☐ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☐ Elo rating
  - ☐ Nash Equilibrium
  - ☐ Replicator dynamics
  - ☐ α-Rank & α^{α}-Rank
- **Policy Improvement in Meta-games**
  - ☐ Iterated self-plays
  - ☐ Fictitious play & generalised weaken fictitious play
  - ☐ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

A Naive Self-play Approach to Our Goal

- Let's do the alchemy for multi-agent learning.
  - Define the "good" to be winning ratio/maximising reward.
  - **Select one learning algorithm**: PPO/TRPO, MADDPG/QMIX.
  - **Select one hyper-parameter tuning model**:, e.g., PBT [Jaderberg 2017].
  - **Start to self-play**: iteratively do best response.
- Master equation of designing gaming AIs for any types of games.

self-plays

PPO + PBT + Self-play = Nothing unhackable

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

A Naive Self-play Approach to Our Goal

$\phi :$

- 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

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

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

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

    - **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.

A Naive Self-play Approach to Our Goal

- Let's formulate the self-play process.

PPO + PBT + Self-play = Nothing unhackable

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

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

Or, even worse

Behavorial cloning on existing players' data + PPO = Nothing unhackable

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

Algorithm 1 Optimization (against a fixed opponent)

input: opponent $\mathbf{w}$; agent $\mathbf{v}_1$
fix objective $\phi_{\mathbf{w}}(\bullet)$
for $t = 1, \ldots, T$ **do**
  $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \phi_{\mathbf{w}}(\bullet)\right)$
end for
output: $\mathbf{v}_{T+1}$

Recall $v' := \text{Br}(w) = \text{Oracle}\left(v, \phi_w(\cdot)\right)$ **s.t.** $\phi_{\mathbf{w}}(\mathbf{v}') > \phi_{\mathbf{w}}(\mathbf{v}) + \epsilon$

The Naive Approach of Self-play Will Not Work

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

PPO + PBT + Self-play = Nothing unhackable

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

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

The Naive Approach of Self-play Will Not Work

- See some counter-examples
  - Rock-Paper-Scissor game:

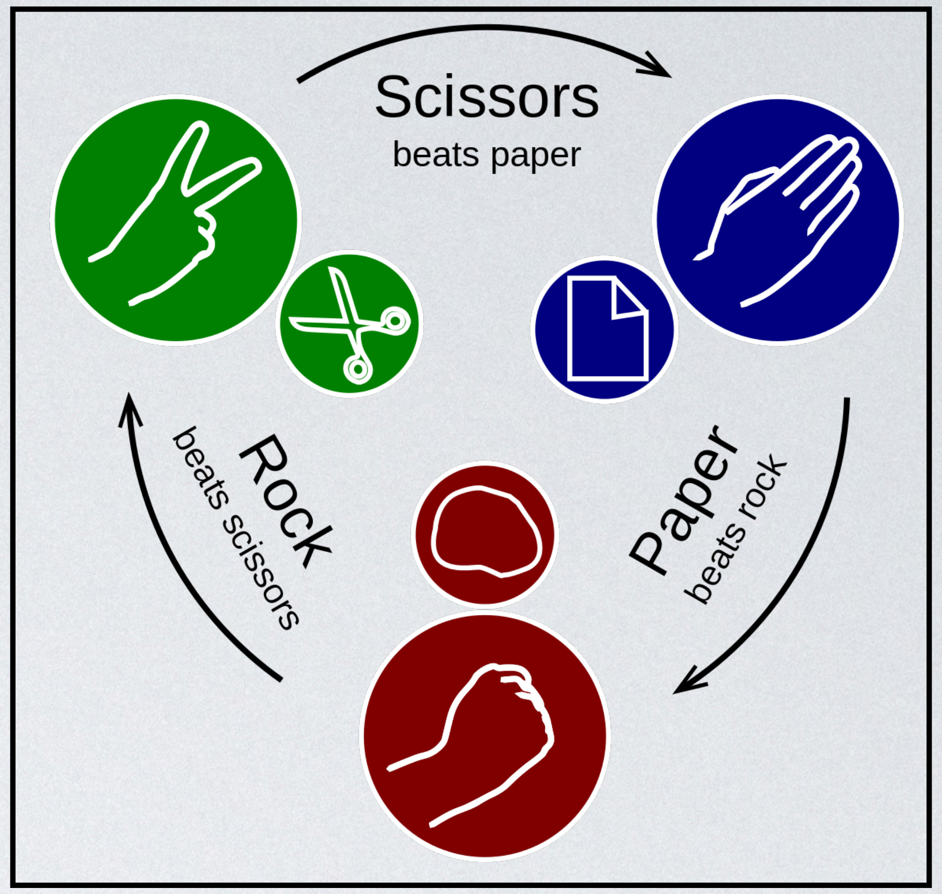
$$\begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$

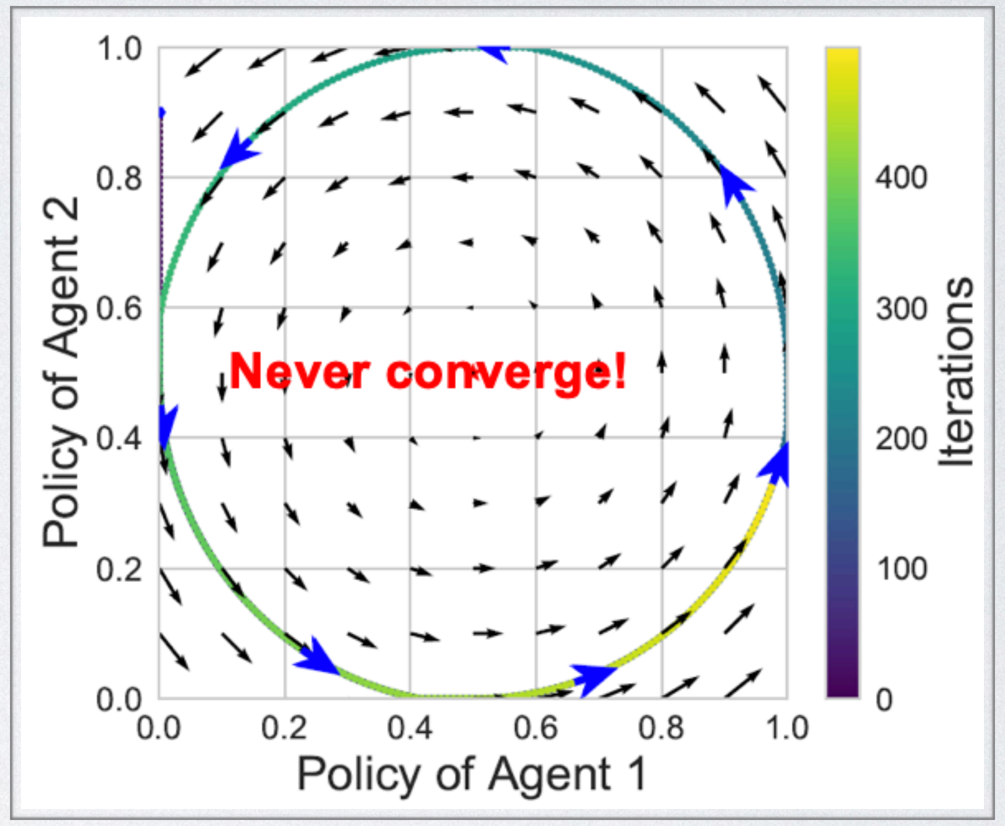
  - Disc game:

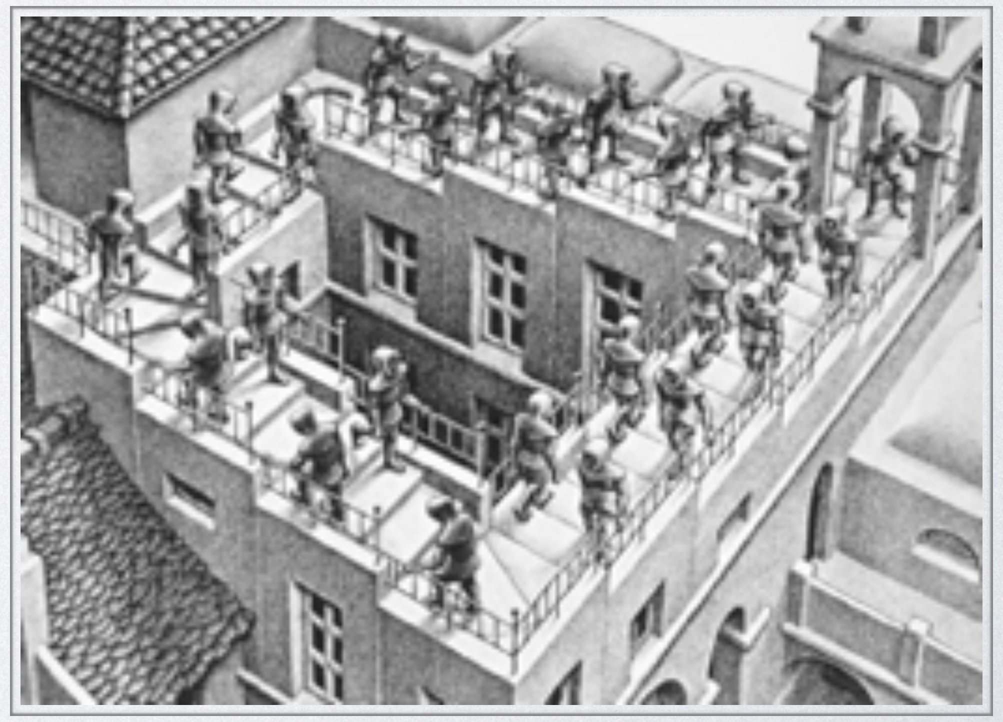
$$\phi(\mathbf{v}, \mathbf{w}) = \mathbf{v}^\top \cdot \begin{pmatrix} 0, & -1 \\ 1, & 0 \end{pmatrix} \cdot \mathbf{w} = v_1 w_2 - v_2 w_1$$

  - or any games that meets the Conservation law

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







Theoretically, Self-play Does Not Work

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

$$\text{FFG} = \text{Transitive game} \oplus \text{In-transitive/Cyclic game}$$

- 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].
- If we define **gradient, divergence, and curl** operators to be

$$\text{grad}(f)(\mathbf{v}, \mathbf{w}) := f(\mathbf{v}) - f(\mathbf{w})$$

$$\text{div}(\phi)(\mathbf{v}) := \int_W \phi(\mathbf{v}, \mathbf{w}) \cdot d\mathbf{w}$$

$$\text{curl}(\phi)(\mathbf{u}, \mathbf{v}, \mathbf{w}) := \phi(\mathbf{u}, \mathbf{v}) + \phi(\mathbf{v}, \mathbf{w}) - \phi(\mathbf{u}, \mathbf{w})$$

Note: these are different operators from basic calculus

- We can write any games ϕ as summation of two **orthogonal** components

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

Transitive game **Cyclic game**

Theoretically, Self-play Does Not Work

- Every FFG can be decomposed into two parts

$$\text{FFG} = \text{Transitive game} \oplus \text{In-transitive/Cyclic game}$$

  - **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, Chess, Battle of Arena.
    - This explains why you don't want to play with rookies, when $f(\boldsymbol{v}_t) \gg f(\boldsymbol{w})$,

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

Theoretically, Self-play Does Not Work

- Every FFG can be decomposed into two parts

FFG = Transitive game ⊕ In-transitive/Cyclic game

- **Cyclic Game**: the rules of winning are not-transitive across different 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 will lead to looping forever.

Physical Meaning of Decomposition in Normal-form Games

- Any normal-form games can be decomposed into two parts [Candogan 2010]:

Normal-form Game = Potential Game $\oplus$ Hamonic Game

- **Transitive (Potential game)**: the single-agent component in the multi-agent learning.

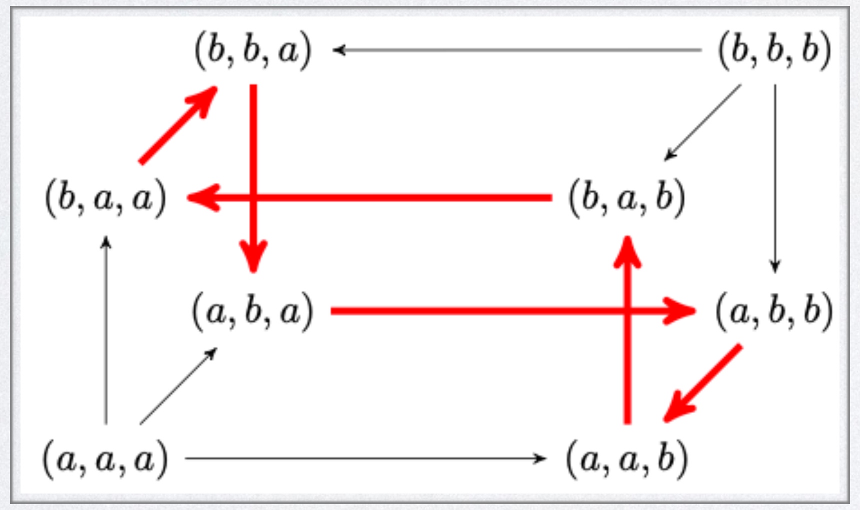
$$\mathbf{E}_{\pi_i,\pi_{-i}}\left[R_i\left(s, a_s^i, a_s^{-i}\right)\right] - \mathbf{E}_{\pi'_i,\pi_{-i}}\left[R_i\left(s, a_s'^{i}, a_s^{-i}\right)\right] = \mathbf{E}_{\pi_i,\pi_{-i}}\left[\mathscr{P}\left(s, a_s^i, a_s^{-i}\right)\right] - \mathbf{E}_{\pi'_i,\pi_{-i}}\left[\mathscr{P}\left(s, a_s'^{i}, a_s^{-i}\right)\right]$$

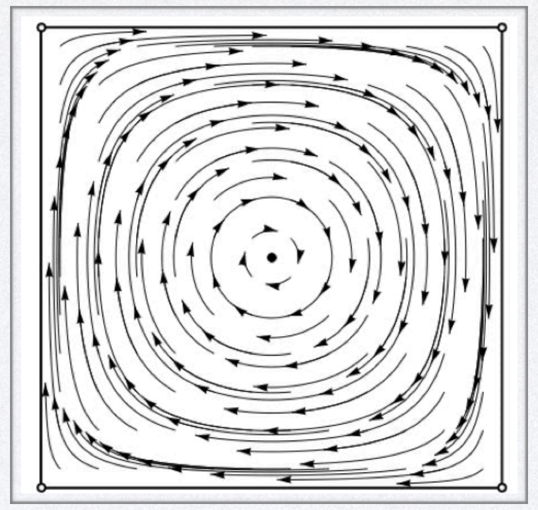
(0, 0)	(1, 2)
(2, 1)	(0, 0)

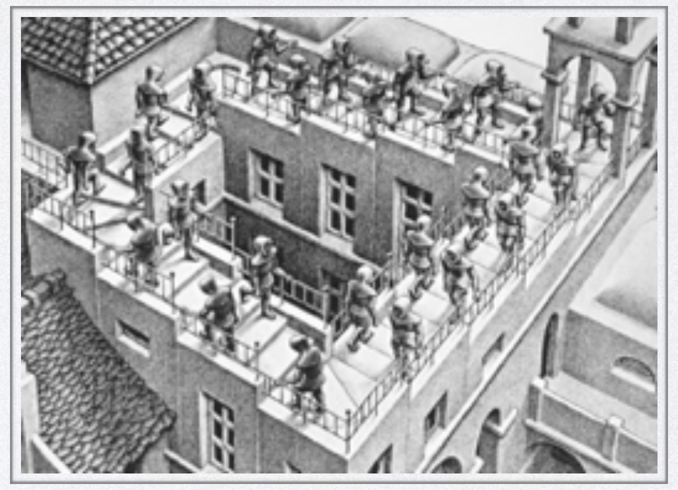
$\Leftrightarrow$

0	2
2	1

- **Cyclic (Harmonic game)**: the origin of limited cycles, uniformly random strategy is always a Nash.







- **Example of decomposition**:

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

(a) Generalized RPS Game

=

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

(c) Potential Component

+

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

(d) Harmonic Component

+

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

(b) Nonstrategic Component

Visualisation of Transitive and In-transitive Games

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

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

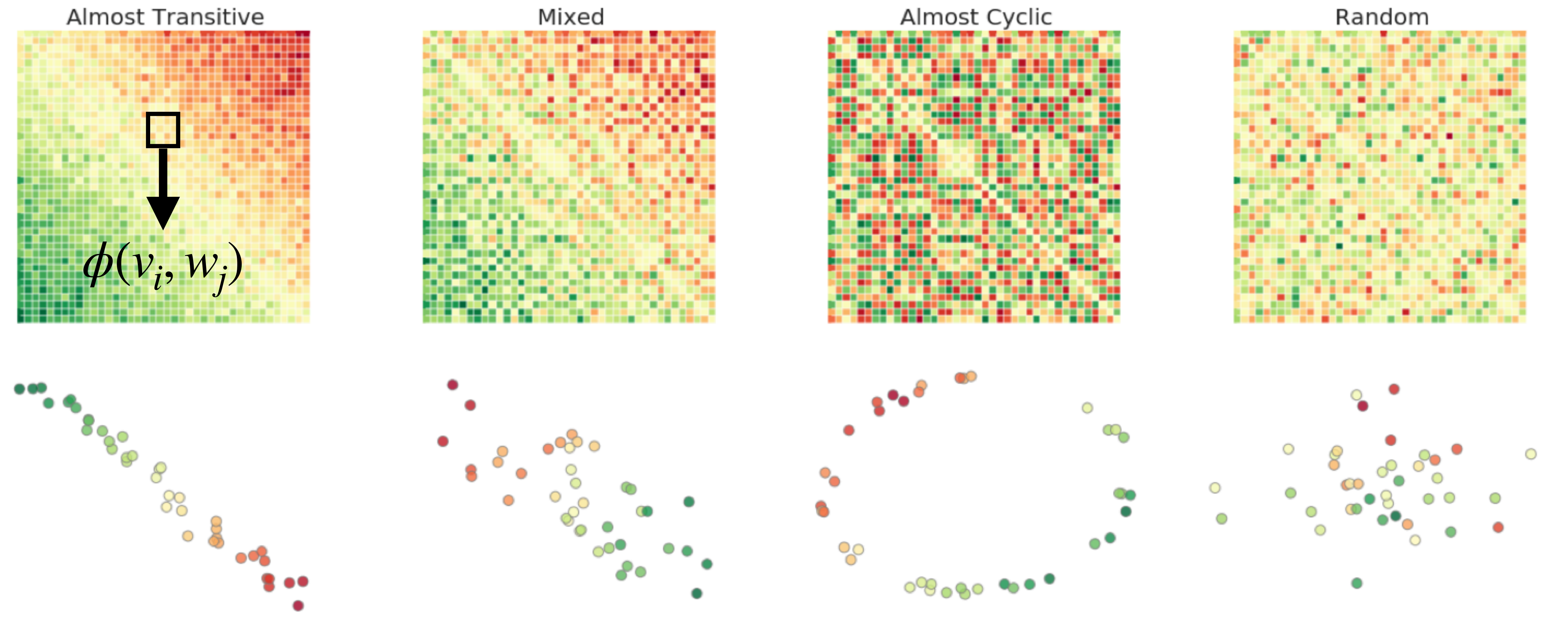
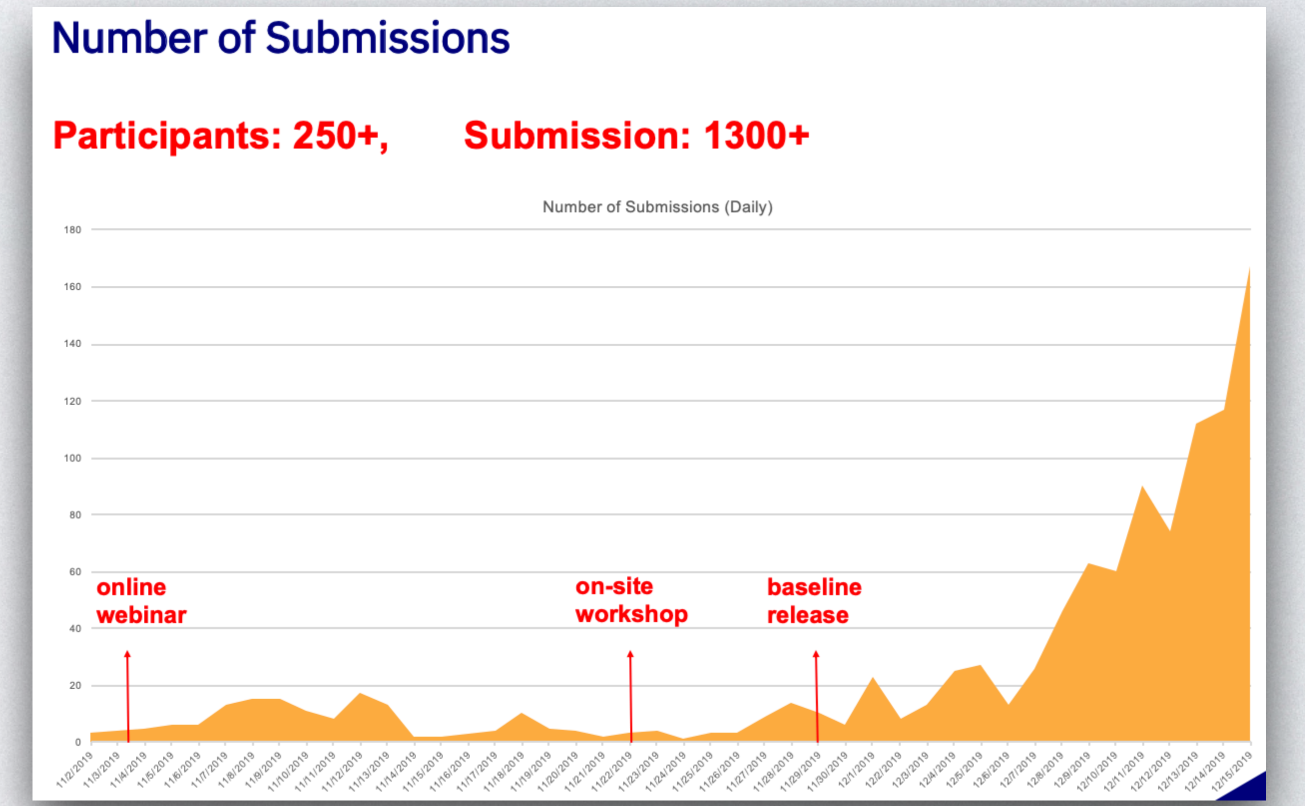


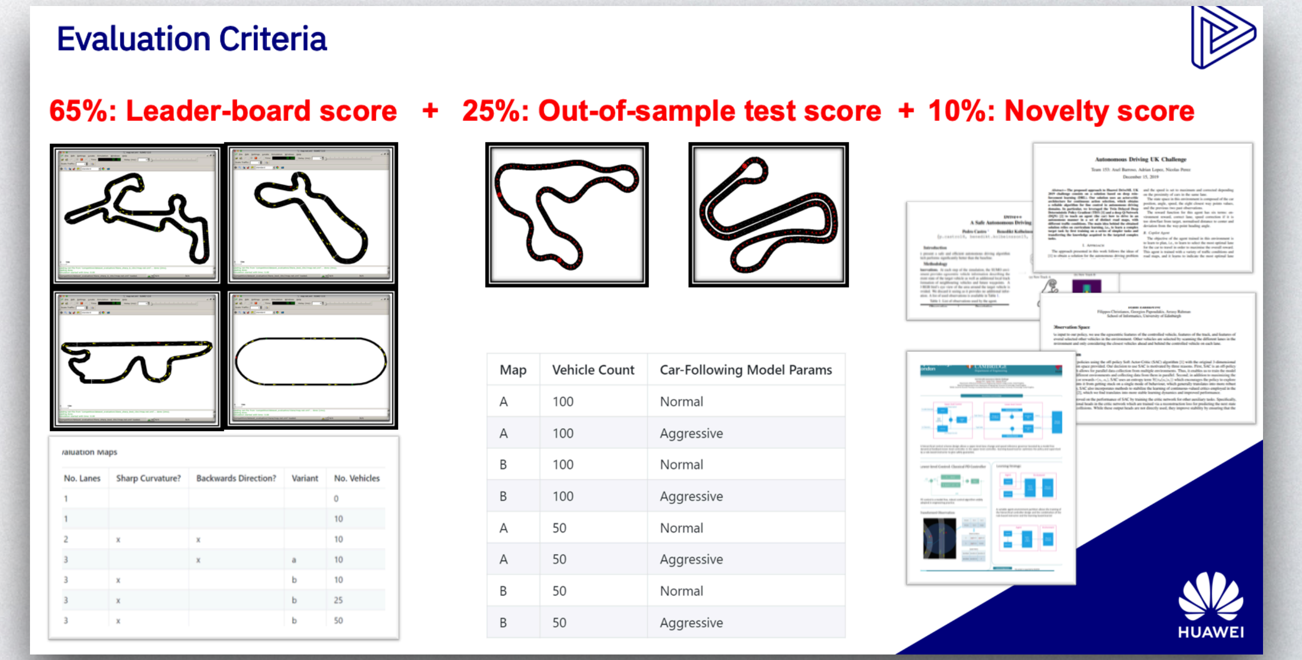
Figure 1. Low-dim gamescapes of various basic game structures. **Top row:** Evaluation matrices of populations of 40 agents each; colors vary from red to green as ϕ ranges over $[-1, 1]$. **Bottom row:** 2-dim embedding obtained by using first 2 dimensions of Schur decomposition of the payoff matrix; 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]

Empirically, Self-play Did Not Work Either!

If we put the top-3 winner models together into one map, the top player will no longer perform the best.

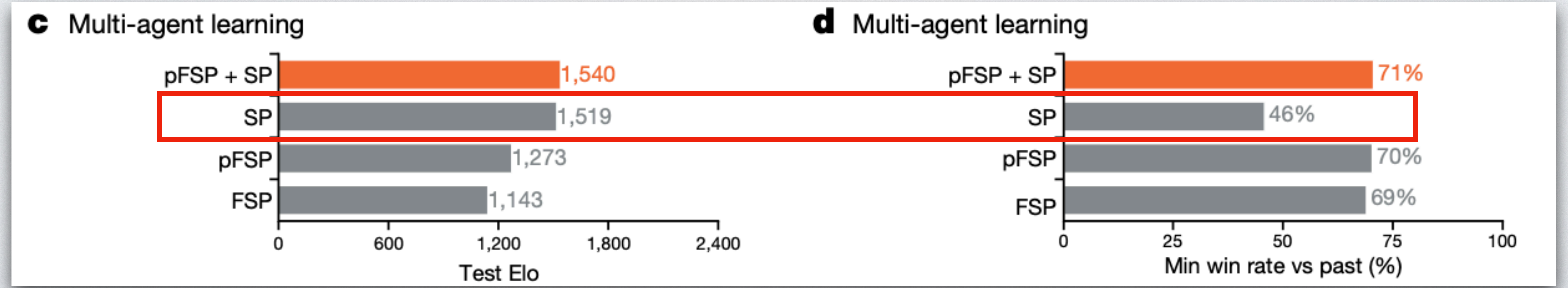




Empirically, Self-play Did Not Work Either!

Example on training AlphaStar:

- self-play can give you agents that are strong in terms of Elo, however, if one makes it compete against its previous strategies, it still loses.
- This shows that naive self-play will not work in real-world games simply because the cyclic dynamics, or, in other words, the agent will forget what has learned.



[Vinyals 2019, Table 3]

The Lesson: Understanding Game Structures are Critical !

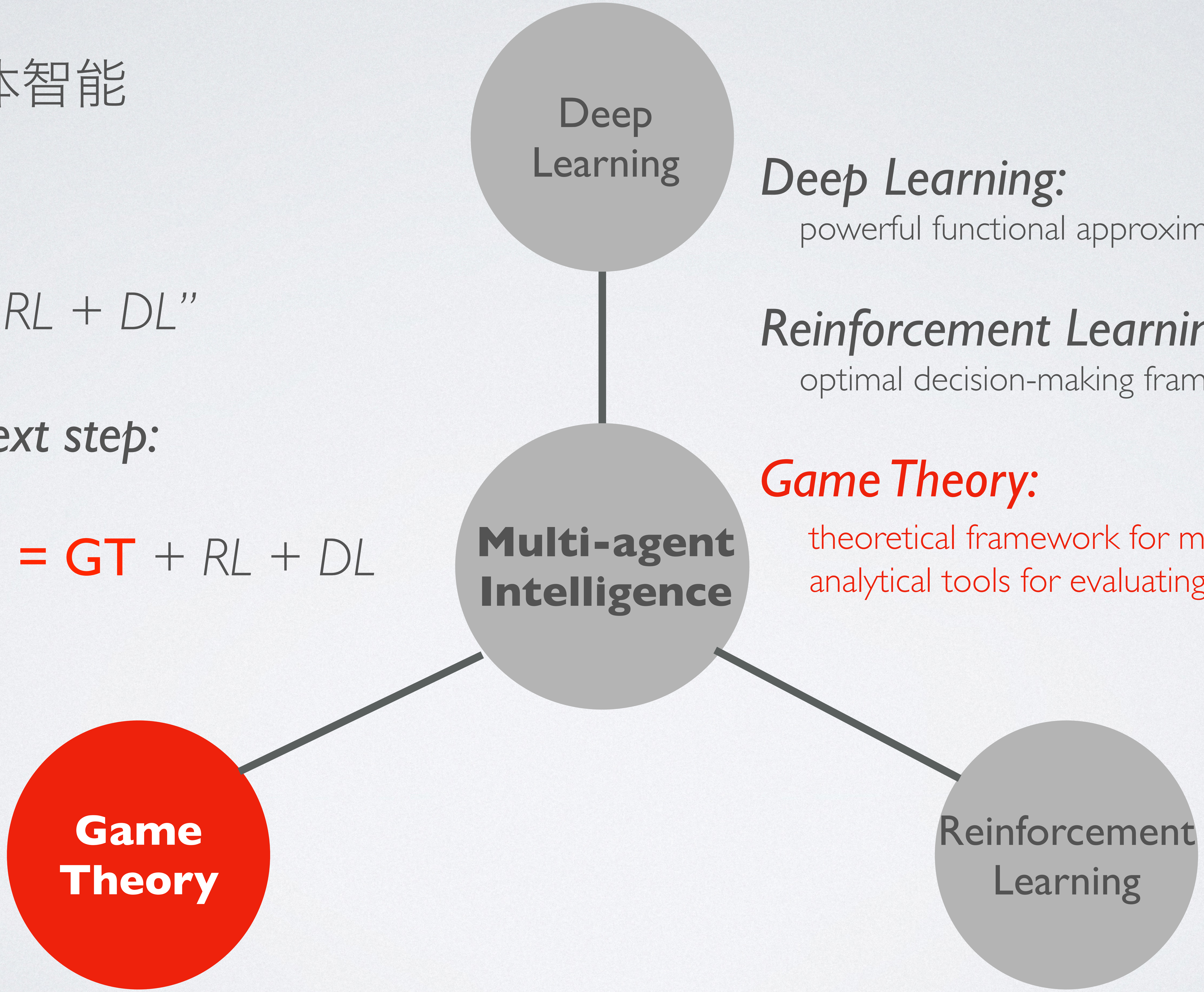
通用智能和群体智能

David Silver:

"AI = RL + DL"

I believe, in the next step:

Multi-agent AI = GT + *RL* + *DL*



Deep Learning:
powerful functional approximator

Reinforcement Learning:
optimal decision-making framework

Game Theory:
theoretical framework for modelling multi-agent system
analytical tools for evaluating agents' policies

Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☑ When self-play does not work
  - ☐ The landscape of real-world games
  - ☐ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☐ Elo rating
  - ☐ Nash Equilibrium
  - ☐ Replicator dynamics
  - ☐ α-Rank & α^α-Rank
- **Policy Improvement in Meta-games**
  - ☐ Iterated self-plays
  - ☐ Fictitious play & generalised weaken fictitious play
  - ☐ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

Recall Our Goal

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

Black-box multi-agent game engine

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

input

Our algorithm:

Multi-agent policy evaluation

Multi-agent policy improvement

output

"good" strategy $(\pi^{1,*}, \ldots, \pi^{N,*})$

Real World Games Look Like Spinning Tops.

- 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).

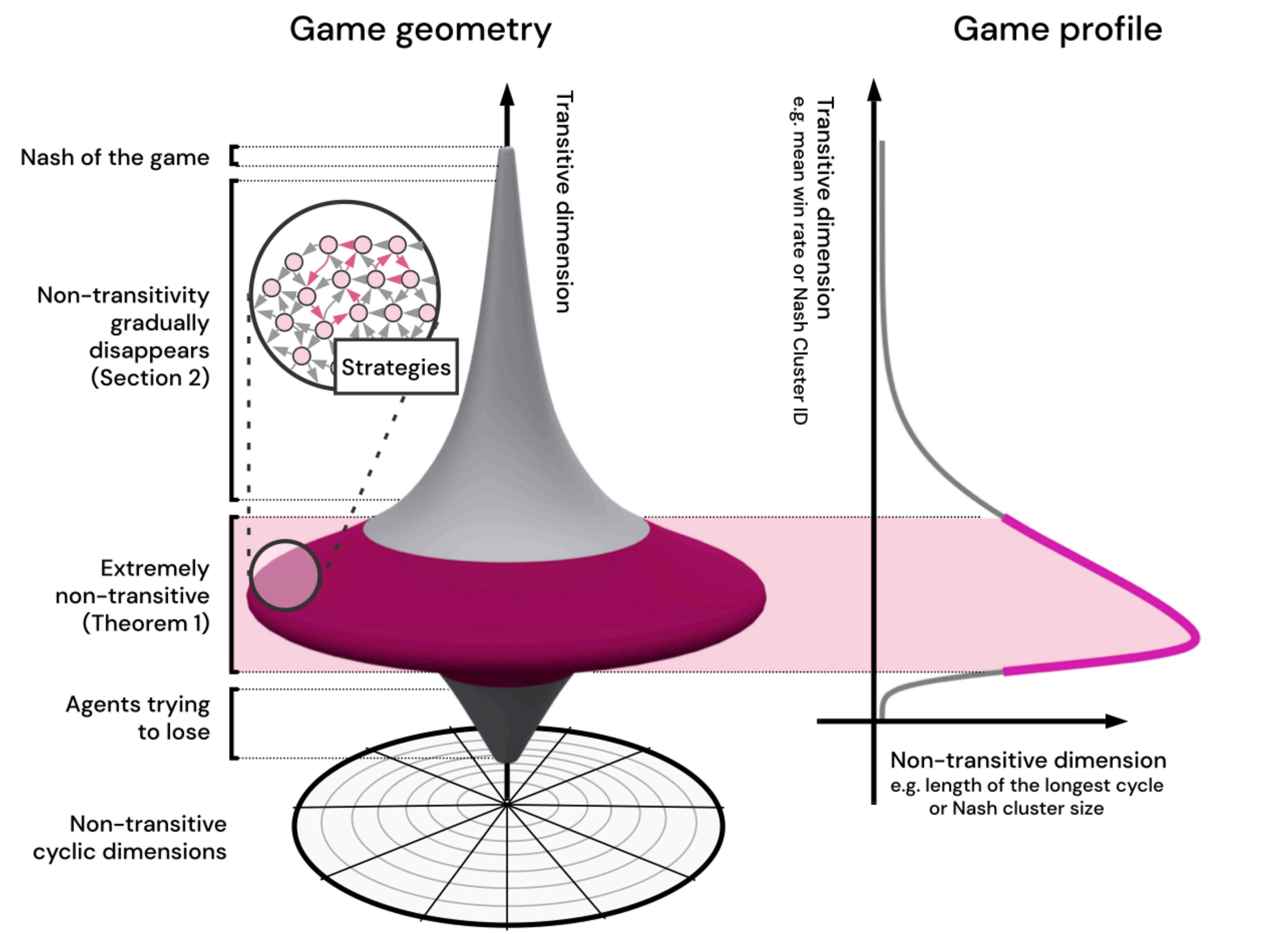
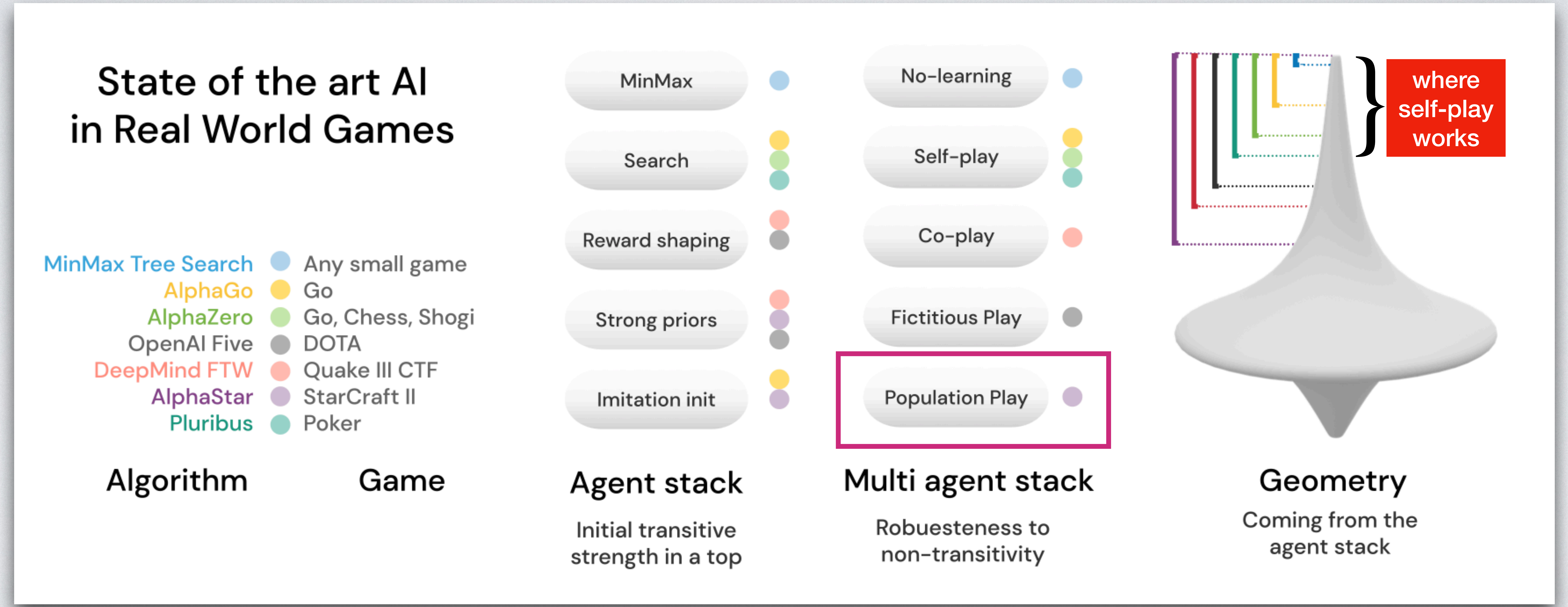


Figure 1: High-level visualisation of the geometry of Games of Skill. It shows a strong transitive dimension, that is accompanied by the highly cyclic dimensions, which gradually diminishes as skill grows towards the Nash Equilibrium (upward), and diminishes as skill evolves towards the worst possible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

Understanding the game structure helps develop solutions

We should have a clear idea of why we use a method rather than hacking by trail and error from the beginning. Never use “reinforcement learning” to design reinforcement learning algorithms!

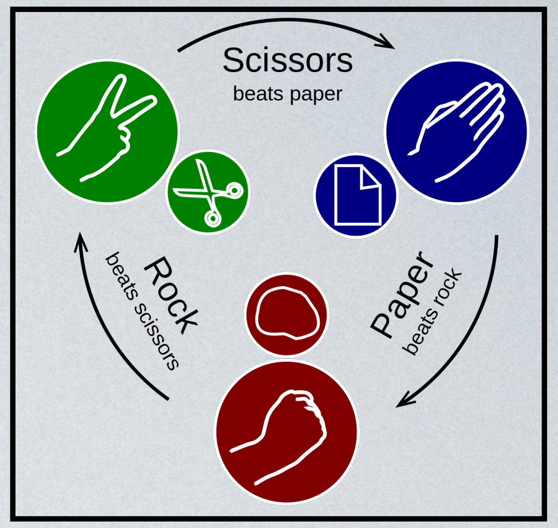


[Czarnecki 2020]

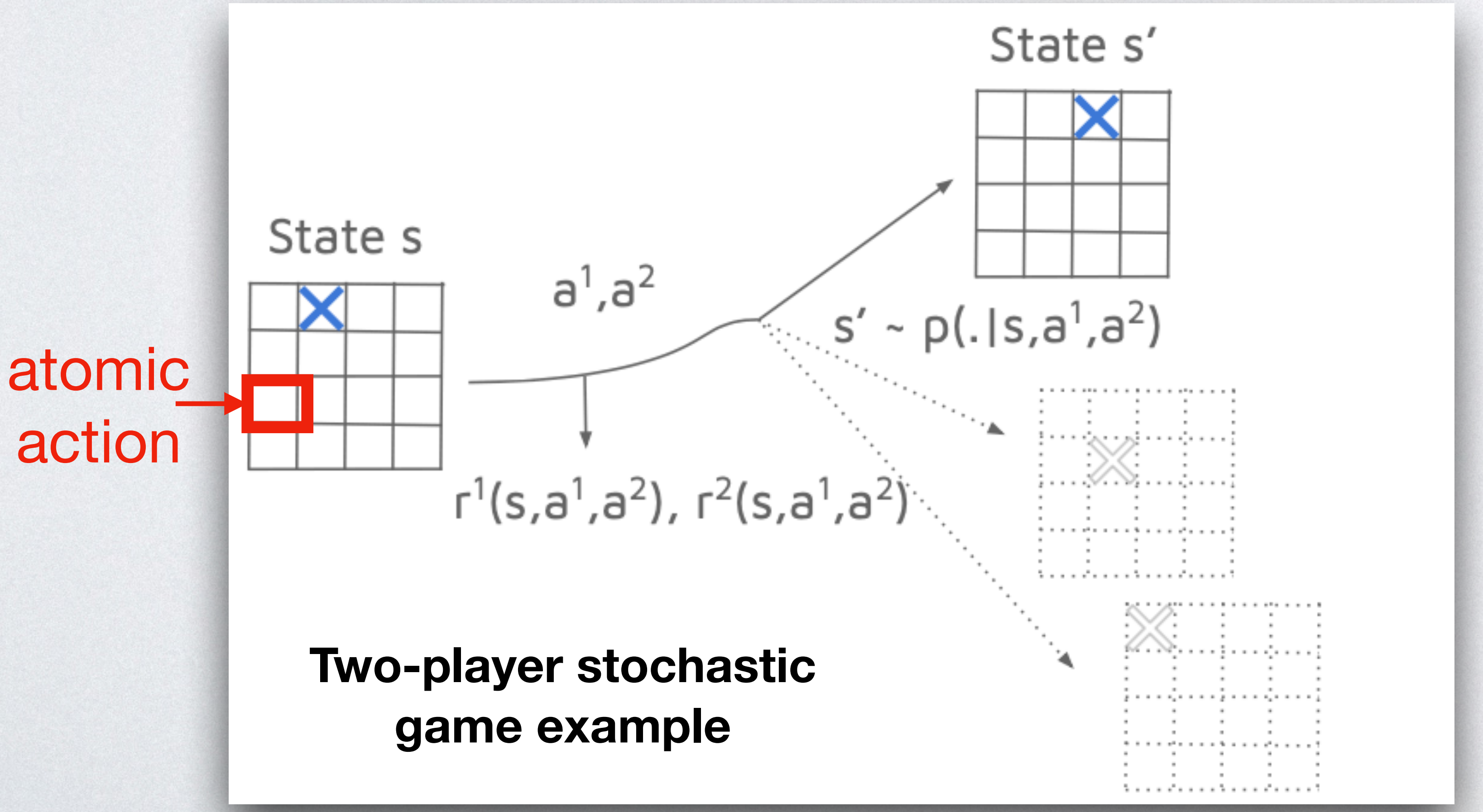
Contents

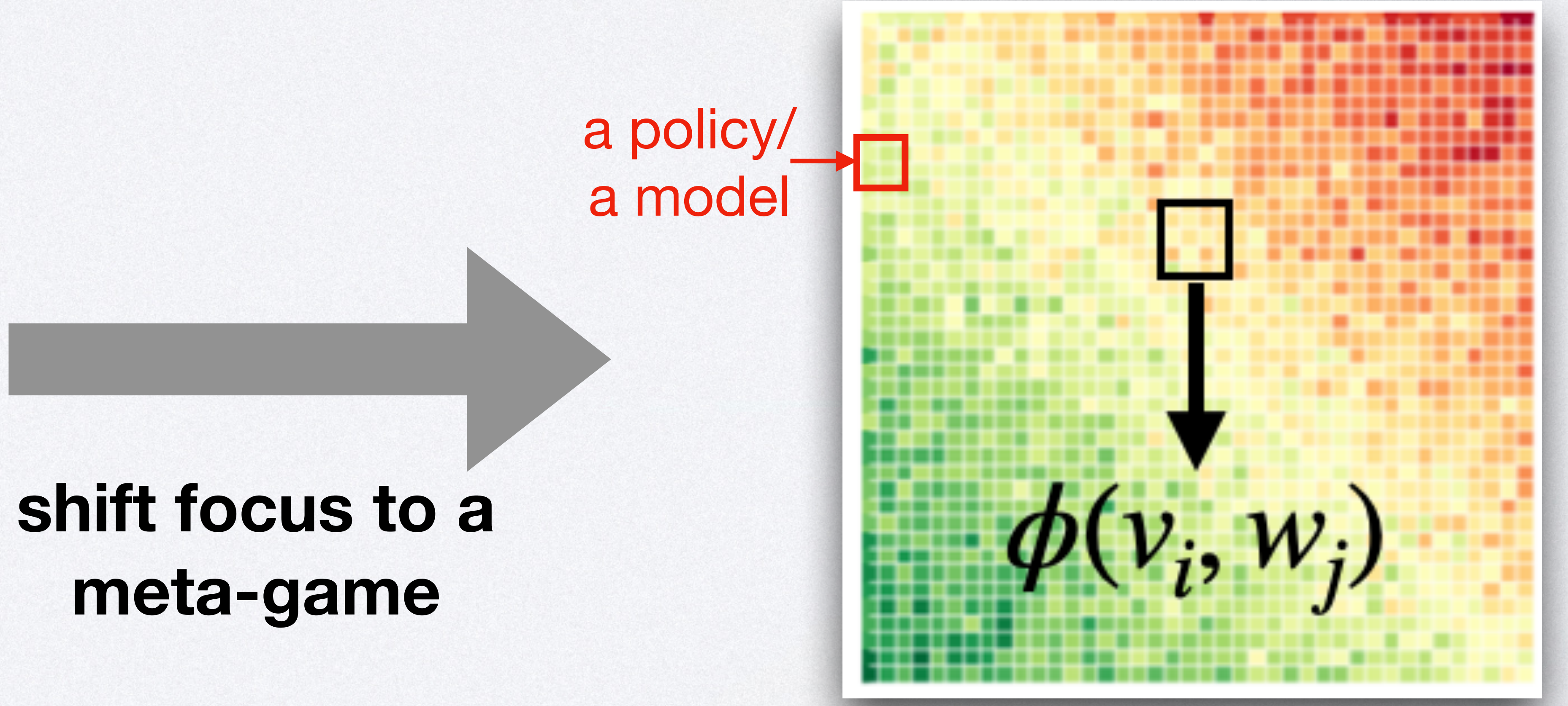
- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☑ When self-play does not work
  - ☑ The landscape of real-world games
  - ☐ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☐ Elo rating
  - ☐ Nash Equilibrium
  - ☐ Replicator dynamics
  - ☐ α-Rank & α^{α}-Rank
- **Policy Improvement in Meta-games**
  - ☐ Iterated self-plays
  - ☐ Fictitious play & generalised weaken fictitious play
  - ☐ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

The Necessity of Studying Meta-games.



- An important intuition of solving games is to train many policies, a population of them. In RPS, if we have a population of three players, each of them plays R/P/S, and we randomise over which player to pick, then no one will ever be able to exploit us.
- On the other hand, enumerating every possible atomic state-action pairs is impossible for real-world games. We have to model on the higher-level policy level, e.g., aggressive/passive styles of policies, rather than state-action level.
- Understanding meta-games can help design both new games, and, new game solvers.
- It is called a **meta-game**, or, **empirical game**, or, **the problem problem**, or, **autocurricula**.





Terminology on Meta-Games.

- In the meta-game analysis, we assume a player can have many copies of itself, each of the copy can play different strategies.
- The "policy" in meta games mean how many copies of that player in the population play that particular type of policy, namely, a policy of policy.

Reinforcement Learning	Game Theory	Meta-game Analysis
environment	game	game
agent	player	population
action	action	type
policy	strategy	distribution over types
reward	payoff	fitness

How Does Meta-games Look Like

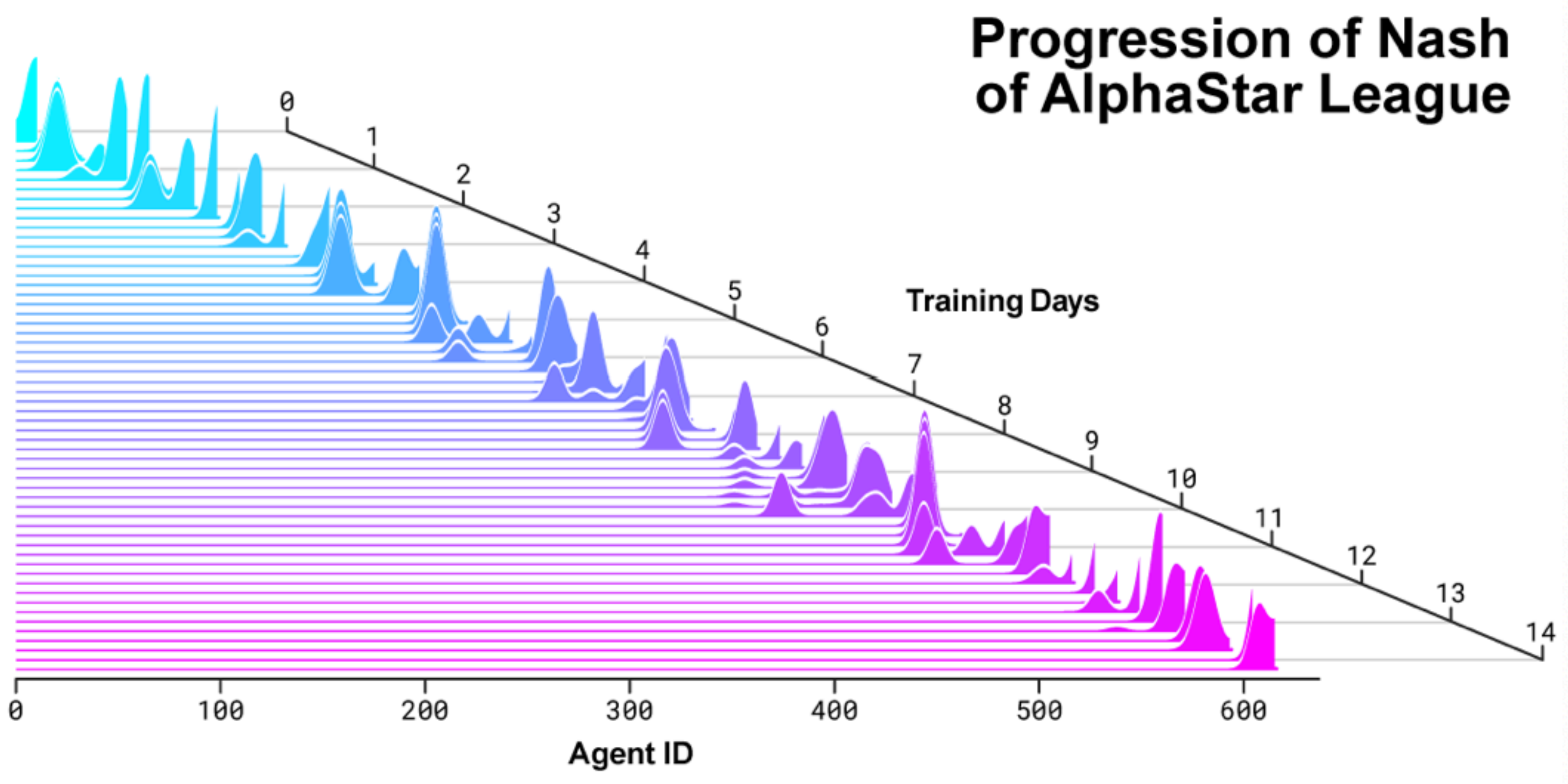
More examples of meta-games on AlphaGO and AlphaStar.

a policy/ a model → α_{rvp}

Extended Data Table 9 | Cross-table of win rates in per cent between programs

	α_{rvp}	α_{vp}	α_{rp}	α_{rv}	α_{r}	α_{v}	α_{p}
α_{rvp}	-	1 [0; 5]	5 [4; 7]	0 [0; 4]	0 [0; 8]	0 [0; 19]	0 [0; 19]
α_{vp}	99 [95; 100]	-	61 [52; 69]	35 [25; 48]	6 [1; 27]	0 [0; 22]	1 [0; 6]
α_{rp}	95 [93; 96]	39 [31; 48]	-	13 [7; 23]	0 [0; 9]	0 [0; 22]	4 [1; 21]
α_{rv}	100 [96; 100]	65 [52; 75]	87 [77; 93]	-	0 [0; 18]	29 [8; 64]	48 [33; 65]
α_{r}	100 [92; 100]	94 [73; 99]	100 [91; 100]	100 [82; 100]	-	78 [45; 94]	78 [71; 84]
α_{v}	100 [81; 100]	100 [78; 100]	100 [78; 100]	71 [36; 92]	22 [6; 55]	-	30 [16; 48]
α_{p}	100 [81; 100]	99 [94; 100]	96 [79; 99]	52 [35; 67]	22 [16; 29]	70 [52; 84]	-
CS	100 [97; 100]	74 [66; 81]	98 [94; 99]	80 [70; 87]	5 [3; 7]	36 [16; 61]	8 [5; 14]
ZN	99 [93; 100]	84 [67; 93]	98 [93; 99]	92 [67; 99]	6 [2; 19]	40 [12; 77]	100 [65; 100]

[Silver 2016, table 9]



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.

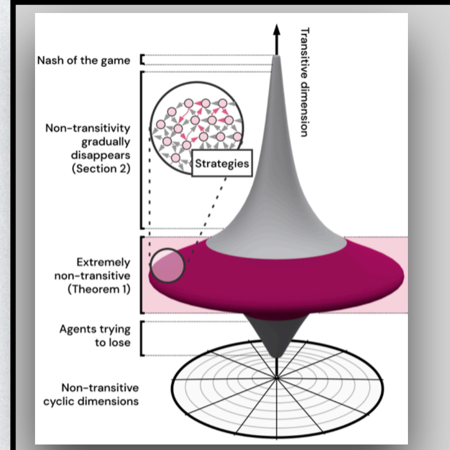
[AlphaStar blog]

The Target of Studying Meta-games.

- In the meta-game analysis, we can ask two critically important questions:

1. *How can we evaluate the population of policies in a meta-game, especially games with limited cycles?*

2. *How can we develop new policies based on the existing population of policies?*



Our algorithm:

1. Multi-agent policy evaluation

2. Multi-agent policy improvement

Relationships between Meta-games and Underlying games

- [Tuyls 2018] proved that a Nash for meta-game is an approximate Nash for the underlying game.
  - Define the Nash for the N-player K-strategy meta-game to be $\mathbf{x} = (x^1, \ldots, x^N)$, $\sum_{j=1}^{K} x_j^i = 1 \ \forall i \in N$.

$$E_{\pi \sim \mathbf{x}} \left[\hat{r}^i(\pi) \right] = \max_{\pi^i} E_{\pi^{-i} \sim x^{-i}} \left[\hat{r}^i \left(\pi^i, \pi^{-i} \right) \right], \forall i \in N$$

  - If we define the reward of the underlying game to be $r^i(\pi^i, \pi^{-i})$, $r^i = \mathbf{E}[\hat{r}^i]$, and $\epsilon = \sup_{\boldsymbol{\pi}, i} | \hat{r}^i(\boldsymbol{\pi}) - r^i(\boldsymbol{\pi}) |$

Distance to the Nash of the underlying game

$$\max_{\pi} E_{\pi^{-i} \sim x^{-i}} \left[r^i \left(\pi^i, \pi^{-i} \right) \right] - E_{\pi \sim x} \left[r^i(\pi) \right]$$

$$\leq \underbrace{\max_{\pi^i} E_{\pi^{-i} \sim x^{-i}} \left[\hat{r}^i \left(\pi^i, \pi^{-i} \right) \right] - E_{\pi \sim x} \left[\hat{r}^i(\pi) \right]}_{=0 \text{ since } x \text{ is a Nash equilibrium for } \hat{r}^i} + \underbrace{\max_{\pi^i} E_{\pi^{-i} \sim x^{-i}} \left[r^i \left(\pi^i, \pi^{-i} \right) - \hat{r}^i \left(\pi^i, \pi^{-i} \right) \right]}_{\leq \epsilon} \underbrace{- E_{\pi \sim x} \left[r^i(\pi) - \hat{r}^i(\pi) \right]}_{\leq \epsilon}$$

$$\leq 2\epsilon$$

- One can further use Hoeffding equation to have a finite-sample bound on how many samples n are needed in order to control ϵ with high probability $1 - \delta$.

$$P\left(\sup_{\boldsymbol{\pi}, i} \left| r^i(\boldsymbol{\pi}) - \hat{r}^i(\boldsymbol{\pi}) \right| < \epsilon \right) \geq \left(1 - 2e^{\left(-2\epsilon^2 n\right)} \right)^{K^{N+1}}$$

Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☑ When self-play does not work
  - ☑ The landscape of real-world games
  - ☑ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☐ Elo rating
  - ☐ Nash Equilibrium
  - ☐ Replicator dynamics
  - ☐ α-Rank & α^α-Rank
- **Policy Improvement in Meta-games**
  - ☐ Iterated self-plays
  - ☐ Fictitious play & generalised weaken fictitious play
  - ☐ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

Policy Evaluation on Meta Games via Elo Ratings

- Elo create a rating $(r_1, \ldots r_N)$ by averaging the historical performance. Assuming the true probability of agent i beating agent j is p_{ij}, Elo approximates it by $\hat{p}_{ij} = \text{softmax}(r_i - r_j)$ through minimising the cross entropy by

$$\ell_{\text{Elo}}\left(p_{ij}, \hat{p}_{ij}\right) = -p_{ij} \log \hat{p}_{ij} - \left(1 - p_{ij}\right) \log \left(1 - \hat{p}_{ij}\right)$$

- Suppose the t-th match pits i against j, and binary outcome is $S_{i,j}^t$, then the rating updates

$$r_i^{t+1} \leftarrow r_i^t - \eta \cdot \nabla_{r_i} \ell_{\text{Elo}}\left(S_{ij}^t, \hat{p}_{ij}^t\right) = r_i^t + \eta \cdot \left(S_{ij}^t - \hat{p}_{ij}^t\right)$$

- With enough race data, Elo ratings will converge to $p_{ij} = \bar{p}_{ij} = \sum_n \frac{S_{ij}^n}{N_{ij}}$, historical average.
- Elo cannot deal with in-transitive games, since $\text{curl}(\text{logit}\mathbf{P}) = 0$.
- In RPS, p_{ij} is (1/2, 1/2, 1/2), thus no predictive power about the game.
- Elo can be biased by weak players that intend to lose (刷分水军/演员) [Balduzzi 2018].

Policy Evaluation on Meta Games via Nash Equilibrium

- Treat meta game as a normal-form game, and compute Nash equilibrium by LP.
- In two-player zero-sum discrete case, it can be solved in polynomial time. The matrix $\mathbf{A}_{\mathfrak{P}}$ is anti-symmetrical, i.e., $\mathbf{A}_{\mathfrak{P}} = -\mathbf{A}_{\mathfrak{P}}^{\top}$.

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

- The minimax theorem is a natural outcome of the duality theorem in LP.

Prime problem

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

/

Dual problem

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

⟺

Minimax theorem

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

Policy Evaluation on Meta Games via Nash Equilibrium

- Cons of Nash equilibrium:
  - Only tractable in two-player zero-sum tabular case. Multi-player general-sum is PPAD-hard.
  - It is a fixed point due to the Brouwer fix-point theorem.
  - What Nash can tell, including its generalisation such as correlated or coarse correlate equilibrium, is the time-averaged behaviour; it tells us little about the "dynamical" behaviour of the actual system.
  - But some dynamics will not only converge to Nash, but they also cycle. Or, they do not end up with Nash at all. The following theorem can summarise.

Poincaré–Bendixson Theorem:

Given a differentiable real dynamical system defined on an open subset of the plane, every non-empty compact ω-limit set of an orbit, which contains only finitely many fixed points, is either

- ***a fixed point***
- ***a periodic orbit***
- ***a connected set composed of a finite number of fixed points together with homoclinic and heteroclinic orbits connecting these.***

Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☑ When self-play does not work
  - ☑ The landscape of real-world games
  - ☑ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☑ Elo rating
  - ☑ Nash Equilibrium
  - ☐ Replicator dynamics
  - ☐ α-Rank & α^{α}-Rank
- **Policy Improvement in Meta-games**
  - ☐ Iterated self-plays
  - ☐ Fictitious play & generalised weaken fictitious play
  - ☐ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

Policy Evaluation on Meta Games via Replicator Dynamics

- Replicator dynamics is a framework of dynamical system that describes the time dependencies of the players' behaviours.
  - Think of an infinitely-sized population of agents, let x_k be the proportion of agents in the population who play the k^{th} strategy among $K-$many possible strategies. In a two-player (i.e. two populations) game, let $(\mathbf{A}, \mathbf{B})$ be the payoff matrix, RD describes the continuous-time evolution of (x_k, y_k).
  - RD only works in symmetrical game $\mathbf{A} = \mathbf{B}^\top$ or anti-symmetrical game $\mathbf{A} = -\mathbf{B}^\top$.

payoff for the k^{th} strategy

$$\frac{dx_k}{dt} = x_k \left[(\mathbf{A}\mathbf{y})_k - \mathbf{x}^T\mathbf{A}\mathbf{y}\right], \quad \frac{dy_k}{dt} = y_k \left[\left(\mathbf{x}^T\mathbf{B}\right)_k - \mathbf{x}^T\mathbf{B}\mathbf{y}\right]$$

current proportion, replicating itself

current payoff against the opponent population

payoff matrix for the other population

Physical Meaning of Replicator Dynamics

- Replicator dynamics is deeply rooted with reinforcement learning.
  - In Cross Learning and finite action-set automata (RL back to the old times), with normalised reward, $0 \leq r \leq 1$, we have the learning rule of the probability of selecting the i-th action as:

$$\pi(i) \leftarrow \pi(i) + \begin{cases} r - \pi(i)r & \text{if } i = j \\ -\pi(i)r & \text{otherwise} \end{cases}$$

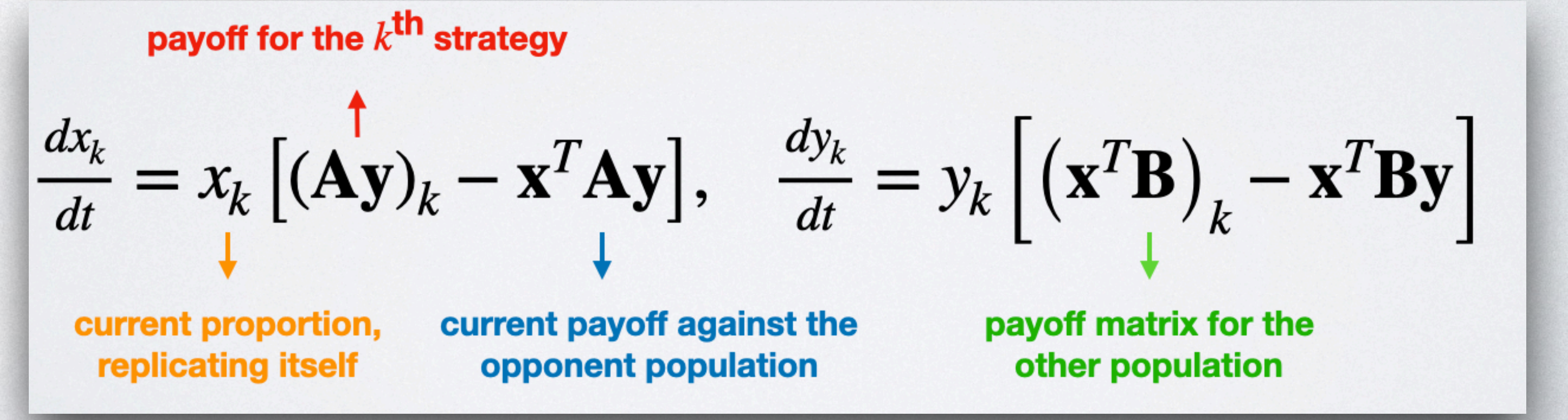
  - We can then write the expected change in policy i by:

$$E[\Delta\pi(i)] = \pi(i)\left[E_i[r] - \pi(i)E_i[r]\right] + \sum_{j \neq i} \pi(j)\left[-E_j[r]\pi(i)\right]$$

$$= \pi(i)\left[E_i[r] - \sum_j \pi(j)E_j[r]\right]$$

  - Assuming to take infinitesimal step $\lim \delta \rightarrow 0$ in $\pi_{t+\delta}(i) = \pi_t(i) + \delta\Delta\pi_t(i)$, we have

$$\dot{\pi}(i) = \pi(i)\left[\mathrm{E}_i[r] - \sum_j \pi(j)\mathrm{E}_j[r]\right]$$



Physical Meaning of Replicator Dynamics

- Replicator dynamics is deep rooted with reinforcement learning.
  - Q-learning can be derived equivalently as a variant of RD with exploration [Kianercy 2012].
  - In the stateless RL setting, one can write Q-learning update rule as

$$Q_i(t+1) = Q_i(t) + \alpha \left[r_i(t) - Q_i(t) \right]$$

Note, no $\max$ is needed here!

  - the continuous limit of the above update rule is

$$\dot{Q}_i(t) = \alpha \left[r_i(t) - Q_i(t) \right]$$

  - and naturally, the policy withe exploration is written as

$$x_i(t) = \frac{e^{Q_i(t)/T}}{\sum_k e^{Q_k(t)/T}}, i = 1,2,\cdots,n$$

  - differentiating the Boltzmann policy w.r.t to time, we can have

$$\frac{\dot{x}_i}{x_i} = [r_i - \sum_{k=1}^{n} x_k r_k] - T \sum_{k=1}^{n} x_k \ln \frac{x_i}{x_k}$$

  - plug in the reward functions

$$\dot{x}_i = x_i \left[(A\mathbf{y})_i - \mathbf{x} \cdot A\mathbf{y} + T_X \sum_j x_j \ln(x_j/x_i) \right]$$

$$\dot{y}_i = y_i \left[(B\mathbf{x})_i - \mathbf{y} \cdot B\mathbf{x} + T_Y \sum_j y_j \ln(y_j/y_i) \right]$$

New term on entropy

payoff for the k^{th} strategy

$$\frac{dx_k}{dt} = x_k \left[(\mathbf{A}\mathbf{y})_k - \mathbf{x}^T \mathbf{A}\mathbf{y} \right], \quad \frac{dy_k}{dt} = y_k \left[\left(\mathbf{x}^T \mathbf{B} \right)_k - \mathbf{x}^T \mathbf{B}\mathbf{y} \right]$$

current proportion, replicating itself

current payoff against the opponent population

payoff matrix for the other population

"*Perhaps a thing is simple if you can describe it fully in several different ways, without immediately knowing that you are describing the same thing*" —— R. Feynman

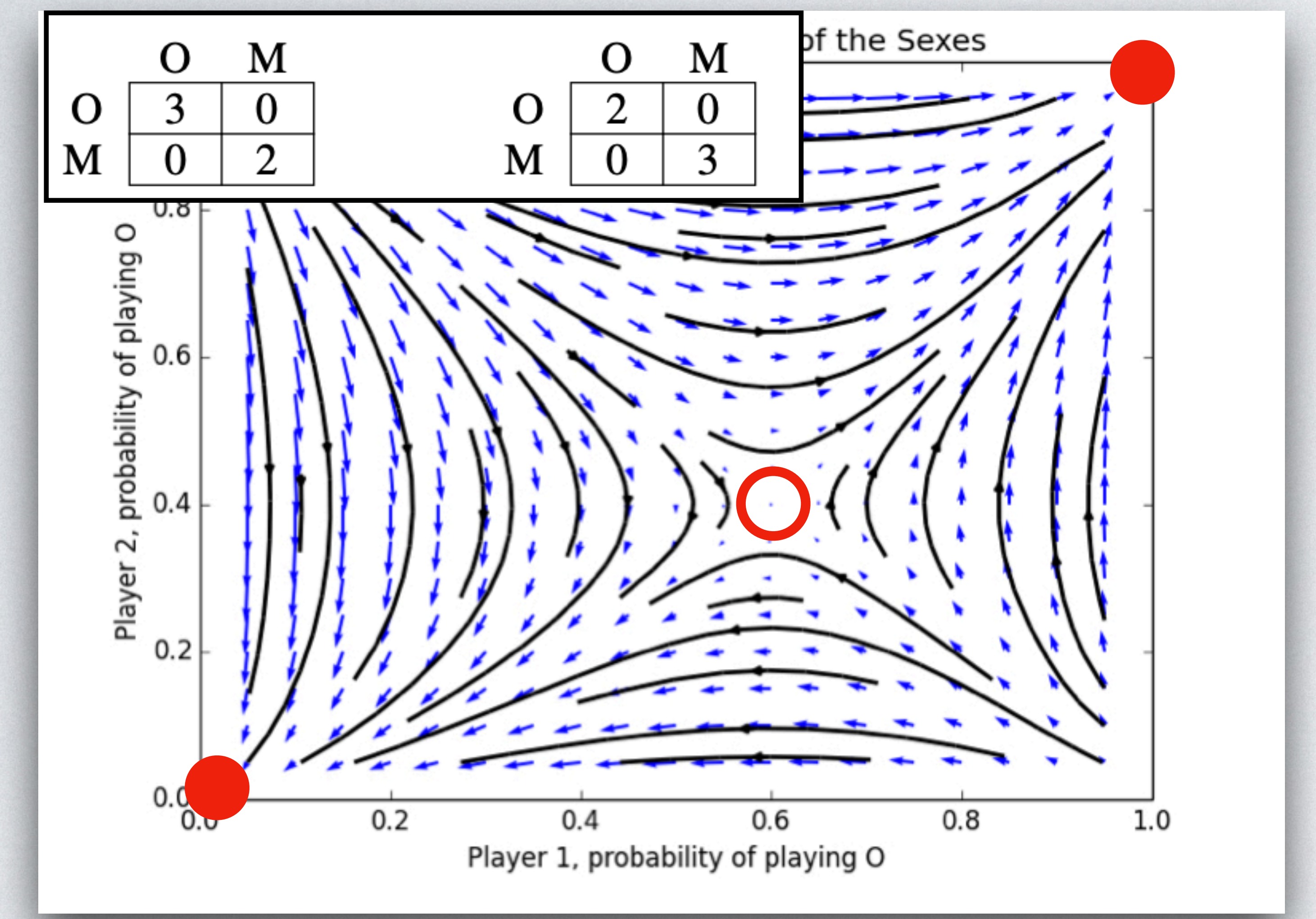
- Many RL algorithms are equivalent to the variants of replicator dynamics.
  - Besides Q-learning, policy gradient can also be written as RD [Hennes 2020].

Table 4: An overview of related empirical evaluations of learning dynamics. NFG: normal-form games; CNFG: continuous action normal-form games; SG: stochastic (Markov) games.

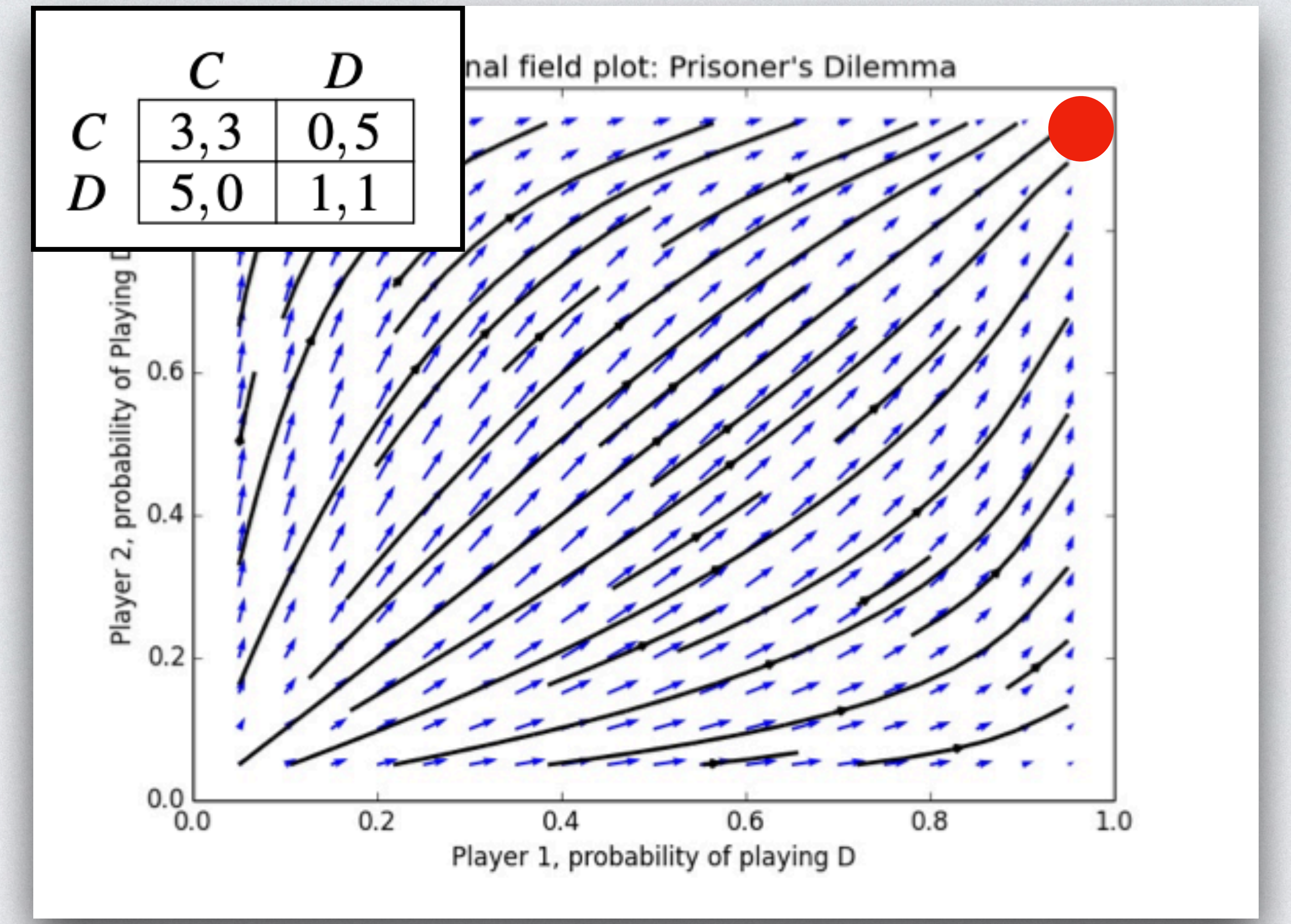
Type	Algorithm	Reference
NFG	Q-learning	Tuyls et al. (2003, 2006)
NFG	regret minimisation	Klos et al. (2010)
NFG	FAQ	Kaisers and Tuyls (2010, 2011)
NFG	lenient FAQ	Bloembergen et al. (2011) Kaisers (2012)
NFG	WoLF	Bowling and Veloso (2002)
NFG	IGA, IGA-WoLF, WPL	Abdallah and Lesser (2008)
CNFG	Q-learning	Galstyan (2013)
SG	networks of learning automata	Vrancx et al. (2008a) Hennes et al. (2009)
SG	RESQ-learning	Hennes et al. (2010)

[Bloembergen 2015]

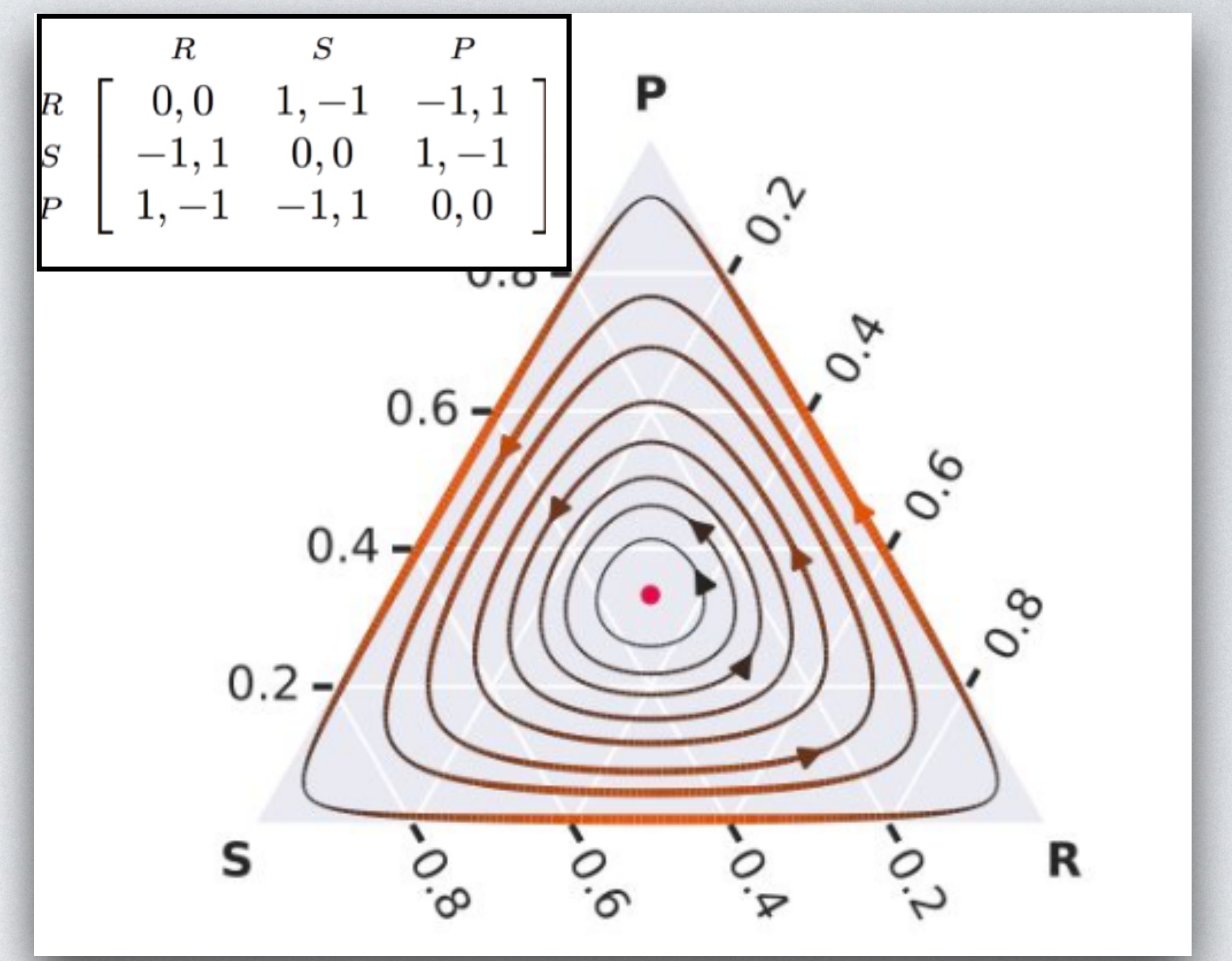
What does Replicator Dynamics suggest



Battle of sexes



Prison’s Dilemma



Rock-Paper-Scissor

Extended Data Table 9 | Cross-table of win rates in per cent between programs

	α_{rvp}	α_{vp}	α_{rp}	α_{rv}	α_r	α_v	α_p
α_{rvp}	-	1 [0; 5]	5 [4; 7]	0 [0; 4]	0 [0; 8]	0 [0; 19]	0 [0; 19]
α_{vp}	99 [95; 100]	-	61 [52; 69]	35 [25; 48]	6 [1; 27]	0 [0; 22]	1 [0; 6]
α_{rp}	95 [93; 96]	39 [31; 48]	-	13 [7; 23]	0 [0; 9]	0 [0; 22]	4 [1; 21]
α_{rv}	100 [96; 100]	65 [52; 75]	87 [77; 93]	-	0 [0; 18]	29 [8; 64]	48 [33; 65]
α_r	100 [92; 100]	94 [73; 99]	100 [91; 100]	100 [82; 100]	-	78 [45; 94]	78 [71; 84]
α_v	100 [81; 100]	100 [78; 100]	100 [78; 100]	71 [36; 92]	22 [6; 55]	-	30 [16; 48]
α_p	100 [81; 100]	99 [94; 100]	96 [79; 99]	52 [35; 67]	22 [16; 29]	70 [52; 84]	-
CS	100 [97; 100]	74 [66; 81]	98 [94; 99]	80 [70; 87]	5 [3; 7]	36 [16; 61]	8 [5; 14]
ZN	99 [93; 100]	84 [67; 93]	98 [93; 99]	92 [67; 99]	6 [2; 19]	40 [12; 77]	100 [65; 100]

AlphaGo meta game

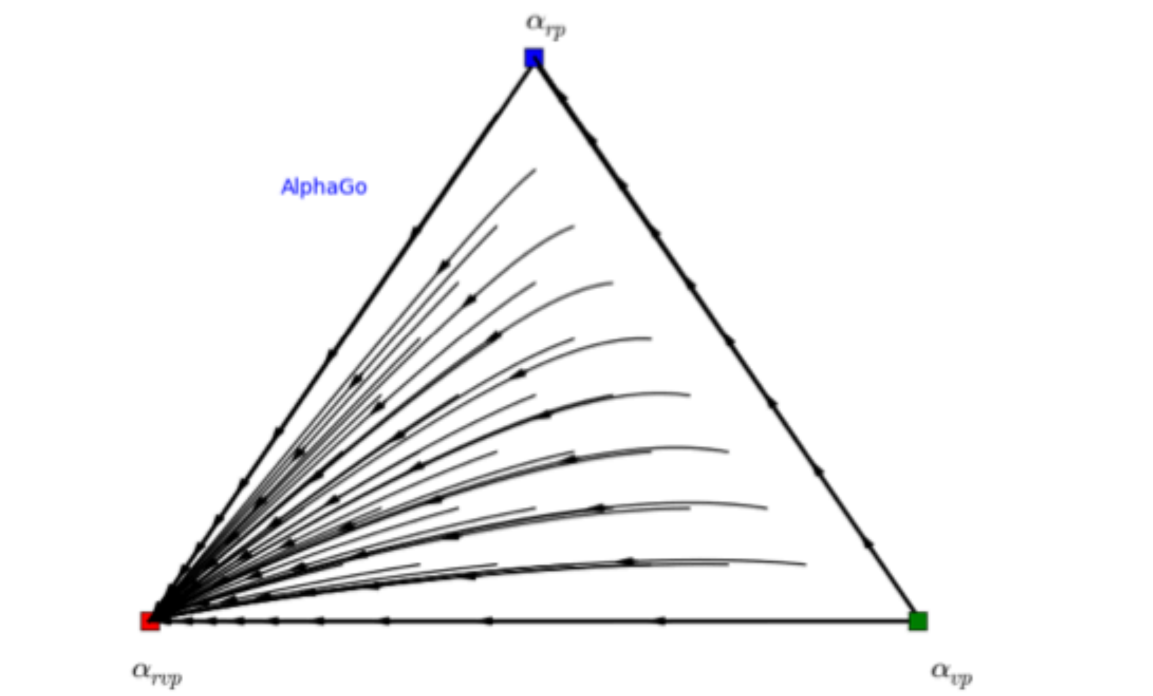
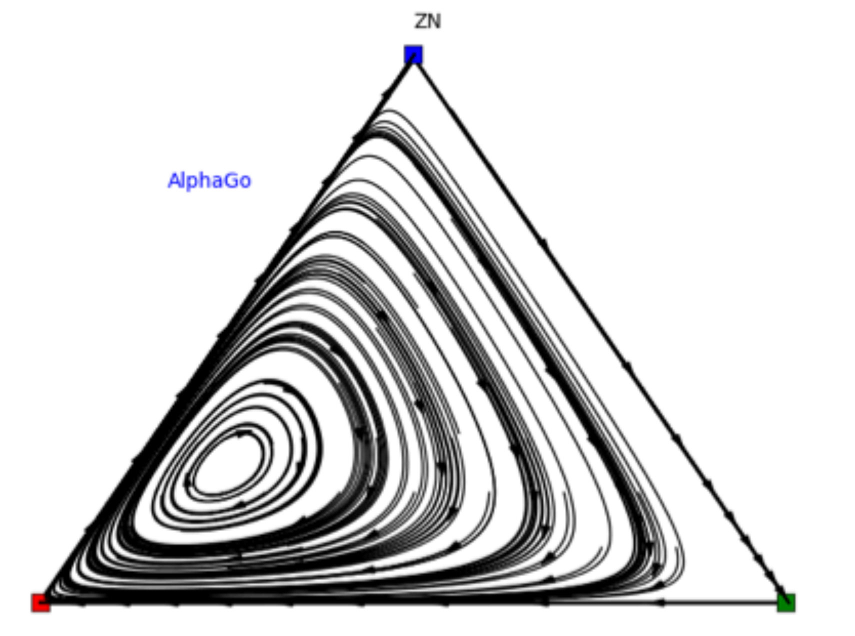


Figure 2: Trajectory plot for the 2-face consisting of strategies α_{rvp}, α_{vp}, α_{rp}

AlphaGo version comparison



(a)

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

AlphaGo version comparison

[Tuyls 2018]

Solution Concept of Replicator Dynamics

- The equilibrium points of replicator dynamics is evolutionary stable strategy (ESS).
  - ESS is new way to define "optimality", similar to the optimality defined in Nash means best response.
  - ESS means the strategy cannot be invaded by any alternative strategies from natural selection.
  - ESS is a refinement of Nash, it is a special type of Nash that is evolutionary stable.
  - On a symmetrical game, Nash equilibrium is:

$$R(\pi, \pi) \geq R(\pi', \pi), \quad \pi' \neq \pi$$

  - ESS refines Nash: $R(\pi, \pi) \geq R(\pi', \pi)$ & $R(\pi, \pi') \geq R(\pi', \pi'), \quad \pi' \neq \pi$
  - Examples of Nash that is not ESS, (A,A)/(B,B) are Nash but only (B,B) is ESS. *A* is not an ESS, so *B* can neutrally invade a population of *A* strategists and predominate, because *B* scores higher against *B* than *A* does against *B*.

	A	B
A	2, 2	1, 2
B	2, 1	2, 2

Harm thy neighbor

A cannot dominate B, since R(B,A)=R(A,A)
but B can dominate A, since R(B,B)>R(A,B)

Pros & Cons of Replicator Dynamics

- Pros of RD
  - RD offers continuous-time dynamics, compared to fixed point Nash, provide insights into micro-dynamical structures of games, e.g., flows, basins of attraction, and equilibria.
  - It provides a new angel to evaluate the policies in a game from a population perspective.
  - The solution concept describes the stability in the sense of evolution (优胜劣汰).
  - It can sift out unstable Nash equilibrium, e.g. the (2/5, 3/5) in battle of sexes.

- Cons of RD
  - It can only apply on two-player several-policy meta game due to the inherently-coupled dynamics.
  - It cannot work on general-sum games, the payoff has to be either symmetrical game $A = B^\top$, or asymmetrical games $A = -B^\top$.
  - The equilibrium is not unique.

Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☑ When self-play does not work
  - ☑ The landscape of real-world games
  - ☑ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☑ Elo rating
  - ☑ Nash Equilibrium
  - ☑ Replicator dynamics
  - ☐ α-Rank & α^α-Rank
- **Policy Improvement in Meta-games**
  - ☐ Iterated self-plays
  - ☐ Fictitious play & generalised weaken fictitious play
  - ☐ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

Weakness of Evaluation Metrics for Meta-games so far.

- **Elo rating:**
  - cannot deal with in-transitive games.
  - cannot tell the dynamics of strategy strength/weakness.
  - cannot stay unbiased to redundant weak agents.

- **Nash equilibrium:**
  - cannot scale to more than two players in non-zerosum games.
  - cannot guarantee uniqueness of equilibrium.
  - cannot tell the dynamics of strategy strength/weakness.

- **Replicator dynamics:**
  - cannot scale to more than two players.
  - cannot deal with general-sum games (either $A = B^\top$ or $A = -B^\top$).
  - cannot guarantee uniqueness of equilibrium.

- **Key requirements:** in-transitive, dynamical, multi-player, general-sum, tractable, unique, stable.

α-Rank: A General Solution Concept for Game Evaluation

α-Rank: Multi-Agent Evaluation by Evolution

father of PPAD class

Shayegan Omidshafiei[*1], Christos Papadimitriou[*3], Georgios Piliouras[*2], Karl Tuyls[*1], Mark Rowland[1], Jean-Baptiste Lespiau[1], Wojciech M. Czarnecki[1], Marc Lanctot[1], Julien Perolat[1], and Remi Munos[1]

[1]DeepMind, 6 Pancras Square, London, UK
[2]Singapore University of Technology and Design, Singapore
[3]Columbia University, New York, USA
[*]Equal contributors, ordered alphabetically. Corresponding author: Karl Tuyls <karltuyls@google.com>.

α-Rank is a new type of evaluation metric that can

- deal with both transitive and in-transitive game dynamics.
- model the flow of dynamics of strategy evolutions, rather than being a fixed point.
- scale to multi-player general-sum cases.
- tractable to be computed, equilibrium can be solved in polynomial time w.r.t the size of meta game.
- equilibrium point is unique, and, (evolutionary) stable.

α-Rank: A General Solution Concept for Game Evaluation

- We knew functional-form games and normal-form games can be decomposed:

[Balduzzi 2019] FFG = Transitive game ⊕ In-transitive/Cyclic game

[Candogan 2010] Normal-form Game = Potential Game ⊕ Hamonic Game

Unifying them can be a very good research topic 😄

- α-Rank is inspired by the Conley's fundamental theorem on dynamical system:

[Conley1978] Any flow on a compact metric space decomposes into a gradient-like part that leads to a recurrent part

- This suggests that a flow is either a part of a "*recurrent chain*", or on its way to converge to a "*recurrent chain*".

- The "recurrent *chain" component* of a game corresponds to the Sink Strongly Connected Component (SSCC) of the response graph.

The Sink Strongly Connected Component of the Response Graph

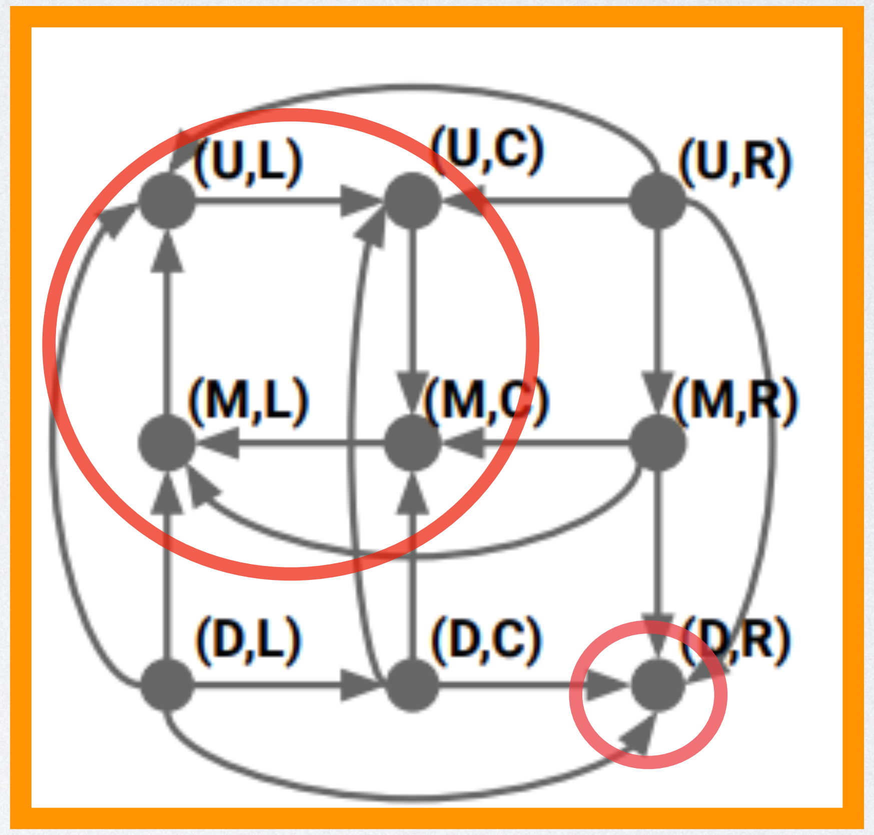
- The response graph of a game is the graph in which the nodes are joint strategy profiles, edges indicates if the deviating player can achieve larger reward.

- Response graph assume one player changes its policy at each time. The graph is sparse!

Game

		II: L	II: C	II: R
I	U	2, 1	1, 2	0, 0
I	M	1, 2	2, 1	1, 0
I	D	0, 0	0, 1	2, 2

Response Graph



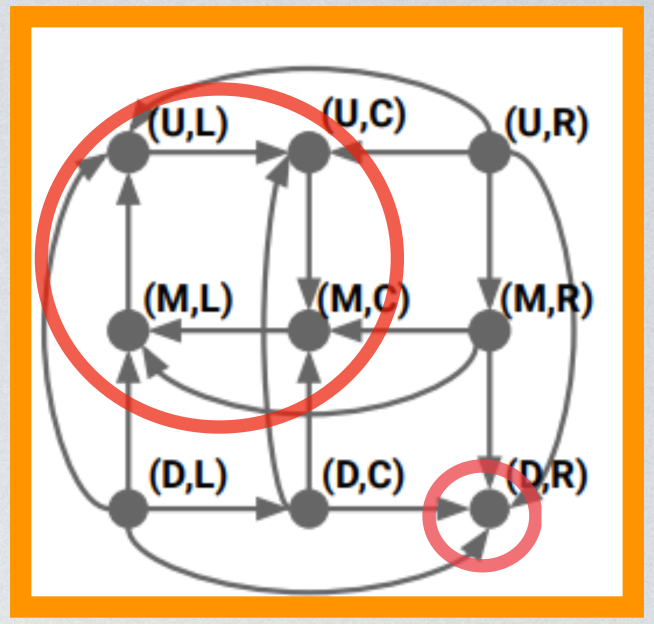
two SSCC here.

- The Sink Strongly Connected Component (SSCC) of the response graph is the subset of nodes in which there are no outbound edges but only inbound edges.

- A node in the flow is either a part of a "*recurrent chain*", or on its way to a "*recurrent chain*".

Modelling the SSCC through a Markov Chain

Response Graph



- SSCC captures the long-term dynamical interactions between agents.
- On the response graph, considering a random walk, following the edges, no matter which node you start from, you will end up converging to the SSCC.
- This process can be modelled through a Markov Chain, and the stationary distribution of the Markov Chain is exactly SSCC.
- To make sure the stationary distribution exists and unique. The chain has to be *irreducible*, meaning every nodes can "travel" to every other nodes.
- To meet such requirement, α-Rank creates a so-called, *Markov-Conley chain*, where the edges are "soft".

α-Rank Algorithm

- α-Rank [Shayegan et al 2019] defines the transitional probability between nodes by

$$\rho_{\pi_{i,a},\hat{\pi}_{i,b}}\left(\pi_{-i}\right) = \frac{1 - e^{-\alpha\left(\mathscr{P}_i\left(\pi_{i,a},\pi_{-i}\right) - \mathscr{P}_i\left(\hat{\pi}_{i,b},\pi_{-i}\right)\right)}}{1 - e^{-m\alpha\left(\mathscr{P}_i\left(\pi_{i,a},\pi_{-i}\right) - \mathscr{P}_i\left(\hat{\pi}_{i,b},\pi_{-i}\right)\right)}}$$

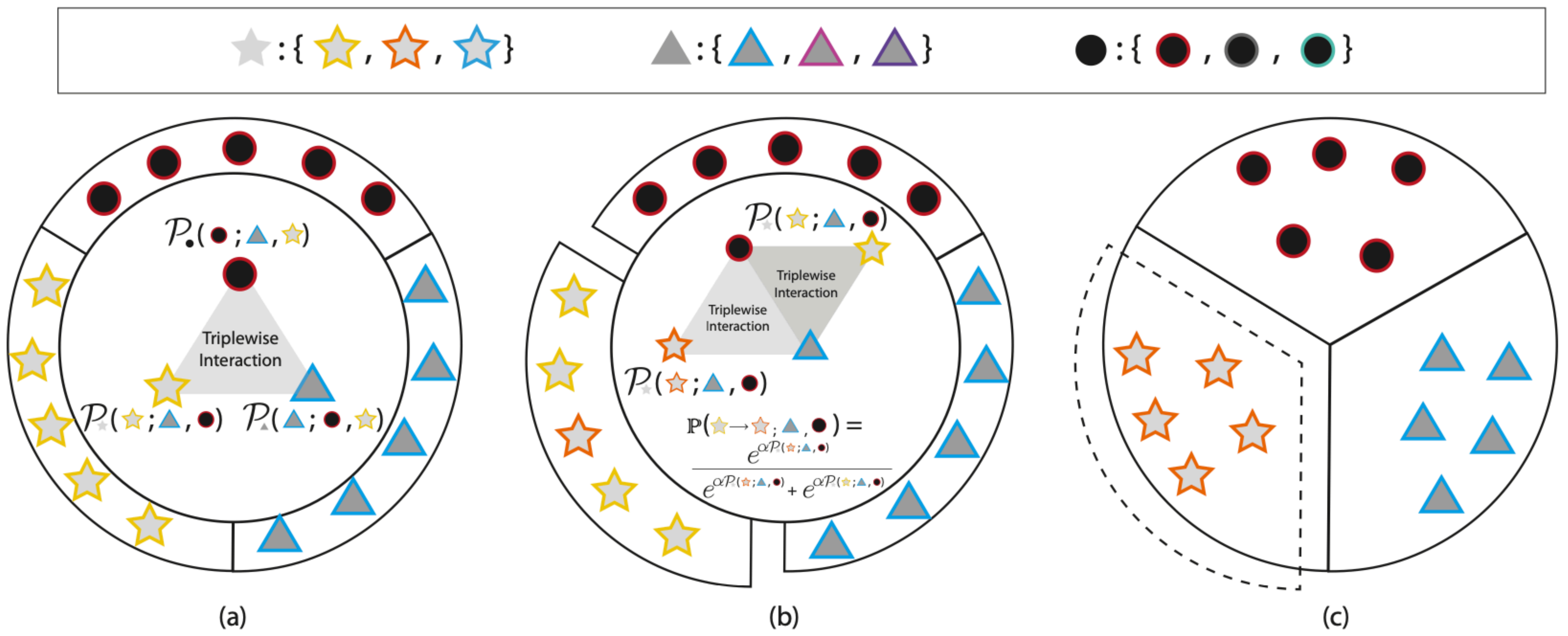


Figure 1: Example of population based evaluation on $N = 3$ players (star, triangle, circle) each with $|s| = 3$ strategies (denoted by the colours) and $m = 5$ copies. a) Each population obtains a fitness value $\mathcal{P}_i$ depending on the strategies chosen, b) one mutation strategy (red star) occurs, and c) the population either selects the original strategy, or being fixated by the mutation strategy.

- Physical meaning of $\rho_{\pi_{i,a},\hat{\pi}_{i,b}}\left(\pi_{-i}\right)$ can be thought of as an evolutionary process above.

transition probability of the Markov Chain

$$[T]_{\pi_{\text{join}},\hat{\pi}_{\text{joint}}} = \begin{cases} \frac{1}{\sum_{l=1}^{N}(k_l-1)}\rho_{\pi_{i,a},\hat{\pi}_{i,b}}\left(\pi_{-i}\right), & \text{if } \left|\pi_{\text{joint}} \backslash \hat{\pi}_{\text{joint}}\right| = 1 \\ 1 - \sum_{\hat{\pi} \neq \pi_{\text{jont}}} [T]_{\pi_{\text{joint}},\hat{\pi}}, & \text{if } \pi_{\text{joint}} = \hat{\pi}_{\text{joint}} \\ 0, & \text{if } \left|\pi_{\text{joint}} \backslash \hat{\pi}_{\text{joint}}\right| \geq 2 \end{cases}$$

α-Rank Algorithm

- α-Rank uses α in $\rho_{\pi_{i,a},\hat{\pi}_{i,b}}\left(\pi_{-i}\right)$ to control the "softness" of edges in the response graph, so that the Markov Chain can be irreducible.
- α means how likely a sub-optimal joint strategy is going to dominate an optimal joint strategy. In experiments, it is usually set as a large number.
- The unique stationary distribution of the Markov chain is

$$\boldsymbol{v} = \lim_{t \to \infty} [T]^t \boldsymbol{v}_0$$

- The rank of probability mass of $\boldsymbol{v}$ is the output of α-Rank. Computing $\boldsymbol{v}$ is polynomial-time.
- The physical meaning is the evolutionary strength/stability of joint strategy profile in terms of how strong it can resisting mutations's invasions. Caveat: this is not the same idea as ESS.
- The connection of α-Rank equilibrium to Nash equilibrium/ESS is unclear yet.

α-Rank Summary

- α-Rank answers the question of how to evaluate/rank joint-policies.
  - A solution concept based on Conley's theorem & graph theory.
    - it can model recurrent chains (limited cycles) in dynamical system, e.g. Rock-Paper-Scissor game.
    - it is tractable in multi-player general-sum games.

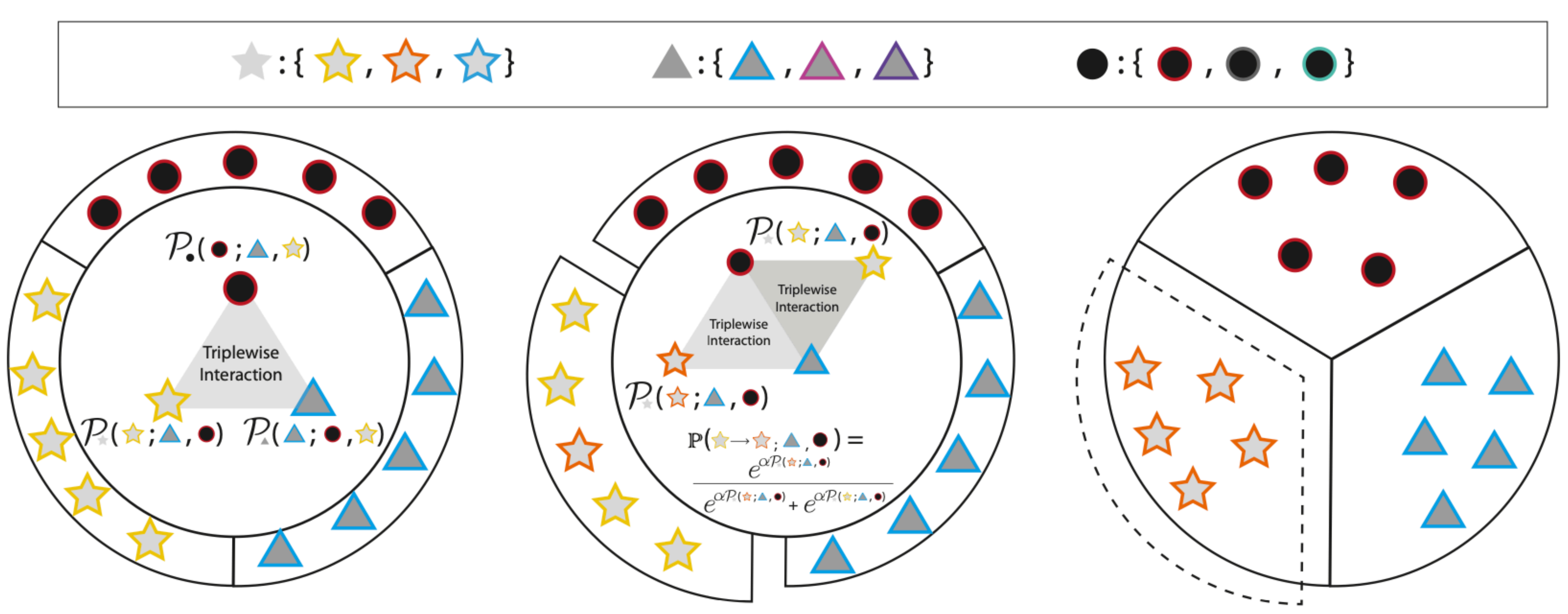
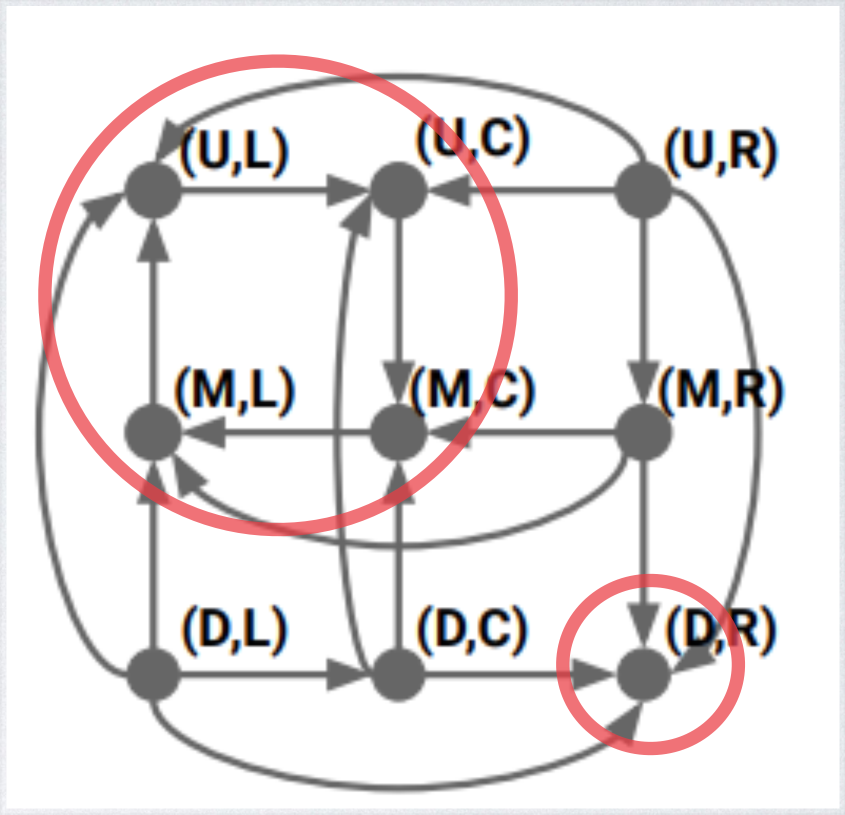


Figure 1: Example of population based evaluation on $N = 3$ players (star, triangle, circle) each with $|s| = 3$ strategies (denoted by the colours) and $m = 5$ copies. a) Each population obtains a fitness value $\mathcal{P}_i$ depending on the strategies chosen, b) one mutation strategy (red star) occurs, and c) the population either selects the original strategy, or being fixated by the mutation strategy.

$$\rho_{\pi_{i,a},\hat{\pi}_{i,b}}\left(\pi_{-i}\right) = \frac{1 - e^{-\alpha\left(\mathcal{P}_i\left(\pi_{i,a},\pi_{-i}\right) - \mathcal{P}_i\left(\hat{\pi}_{i,b},\pi_{-i}\right)\right)}}{1 - e^{-m\alpha\left(\mathcal{P}_i\left(\pi_{i,a},\pi_{-i}\right) - \mathcal{P}_i\left(\hat{\pi}_{i,b},\pi_{-i}\right)\right)}}$$

Example:

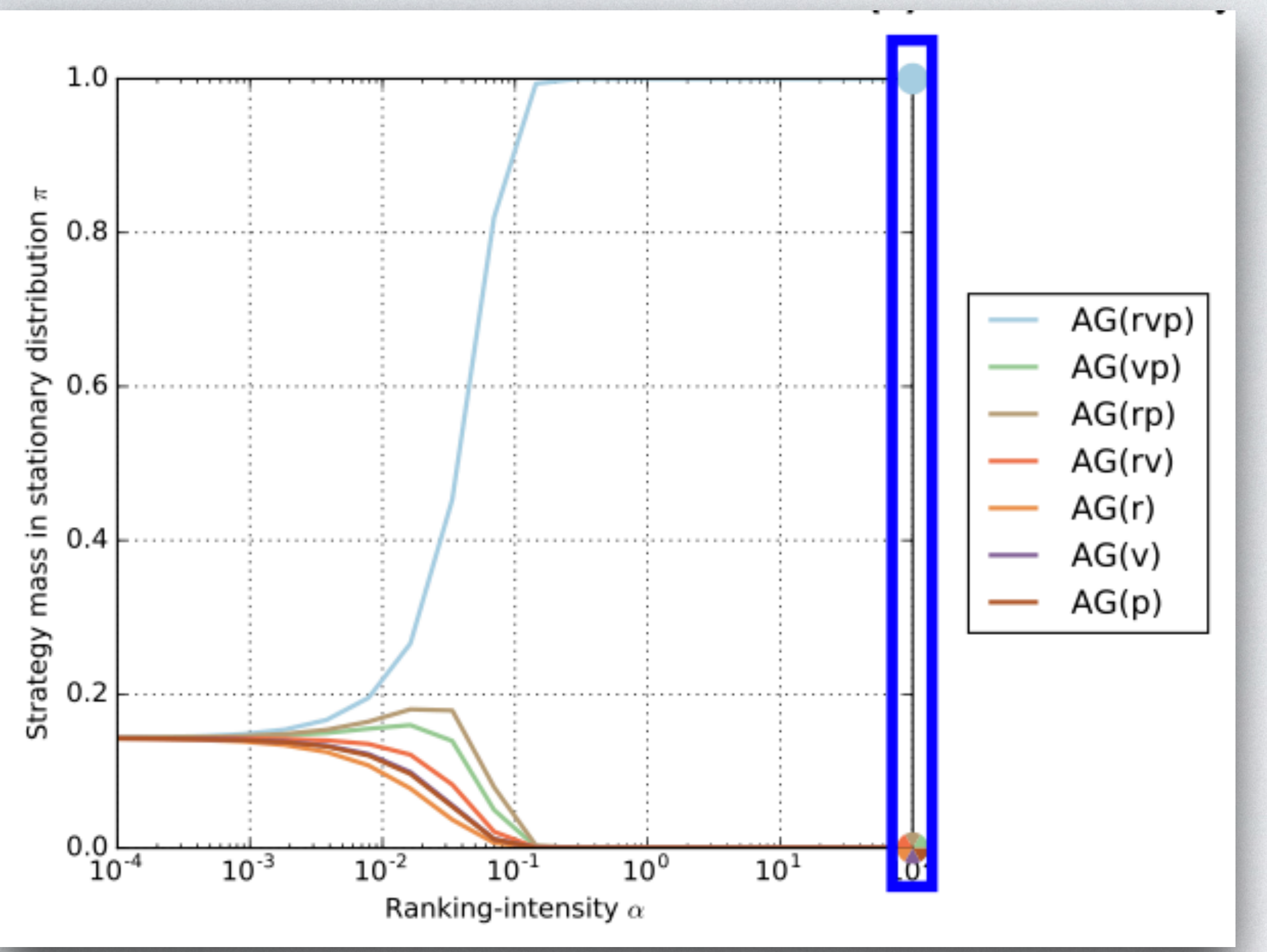
		II		
		L	C	R
	U	2, 1	1, 2	0, 0
I	M	1, 2	2, 1	1, 0
	D	0, 0	0, 1	2, 2

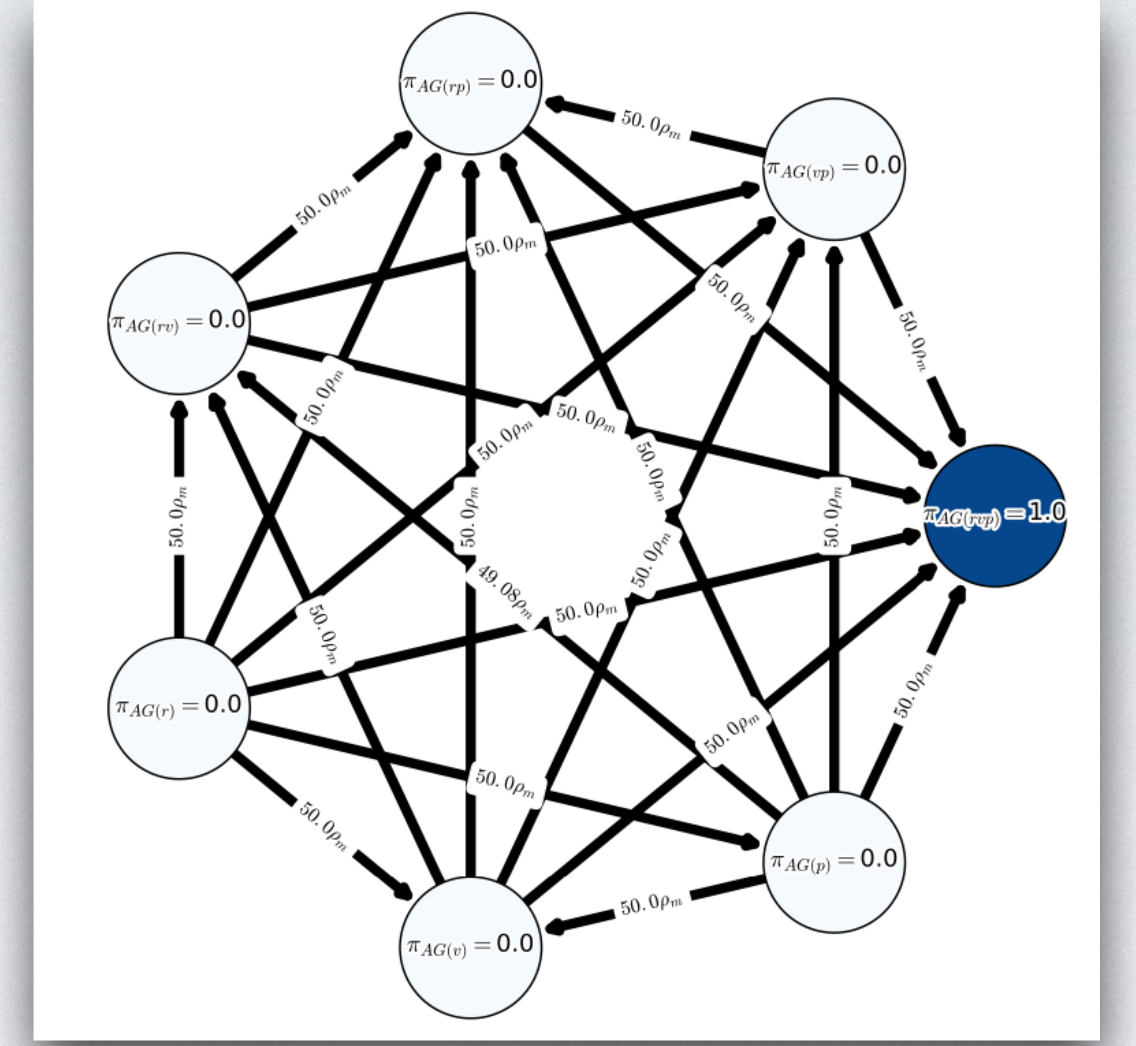


1. Collect the pay-off values for different strategy profiles.
2. Construct the Markov Chain based on $\rho_{\pi_{i,a},\hat{\pi}_{i,b}}\left(\pi_{-i}\right)$
3. Compute the stationary distribution $\boldsymbol{v} = \lim_{t\to\infty} [T]^t \boldsymbol{v}_0$
4. Rank the joint strategy profile based on probability of $\boldsymbol{v}$.

α-Rank Results

AlphaGo version comparison



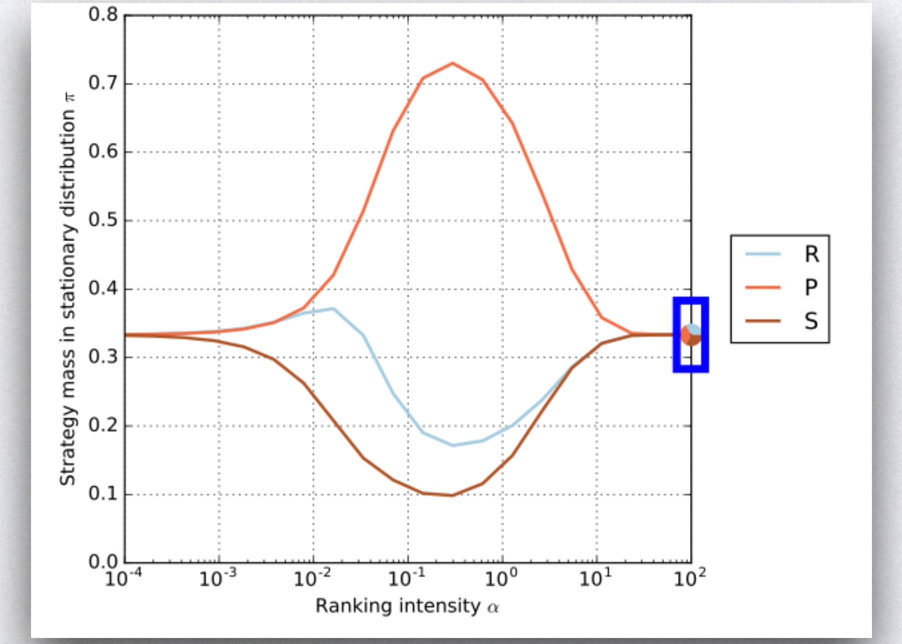


Agent	Rank	Score
$AG(rvp)$	1	1.0
$AG(vp)$	2	0.0
$AG(rp)$	2	0.0
$AG(rv)$	2	0.0
$AG(r)$	2	0.0
$AG(v)$	2	0.0
$AG(p)$	2	0.0

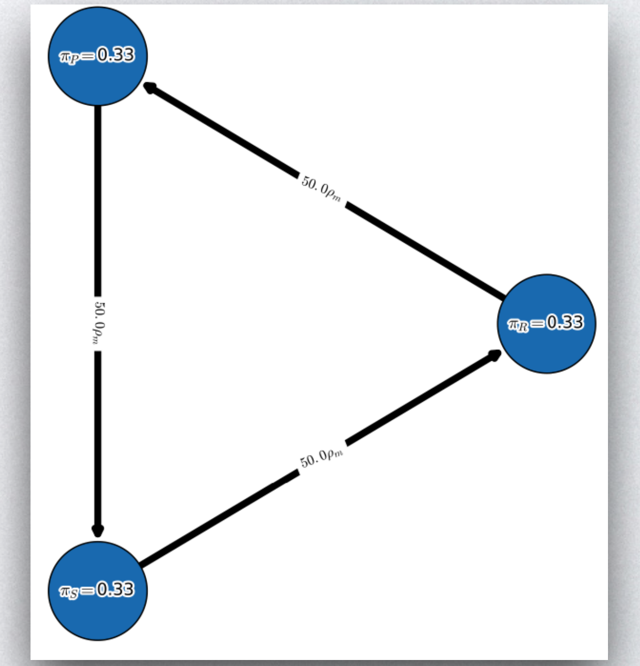
Biased RPS

	R	P	S
R	0	−0.5	1
P	0.5	0	−0.1
S	−1	0.1	0

(a) Payoff matrix.



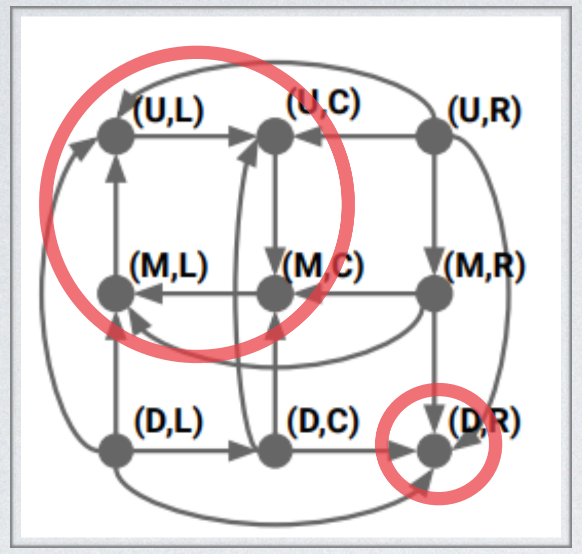
Agent	Rank	Score
R	1	0.33
P	1	0.33
S	1	0.33



α^{α}-Rank: A Scalable Solution for α-Rank [Yang 2020]

Example:

		II		
		L	C	R
I	U	2, 1	1, 2	0, 0
	M	1, 2	2, 1	1, 0
	D	0, 0	0, 1	2, 2



1. Collect the pay-off values for different strategy profiles.

2. Construct the Markov Chain based on $\rho_{\pi_{i,a},\hat{\pi}_{i,b}}(\pi_{-i})$

3. Compute the stationary distribution $\boldsymbol{\nu} = \lim_{t\to\infty} [T^T]^t \boldsymbol{\nu}_0$

4. Rank the joint strategy profile based on probability of $\boldsymbol{\nu}$.

Conclusion:

1. We conjecture that solving α-Rank is still **NP-Hard** because the size of the Markov Chain is exponential to the number of agents.

2. A polynomial-time solver on exponential-sized input cannot be claimed as tractable.

3. Take TSP as example, one cannot claim a NP-Hard problem solvable by just creating an exponentially-sized input.

news.eeworld.com.cn › EEWorld ▾ Translate this page
DeepMind阿尔法系列被华为怒怼 - 电子工程世界
23 Nov 2019 - 从AlphaGo 到AlphaStar, 每一项研究都取得了举世瞩目的成就, 但就在最近, DeepMind 的一篇有关多智能体强化学习的论文被华为英国研究中心「打 ...

www.cnkanglong.com › news1 ▾ Translate this page
yabo：你的算法耗尽全球GPU算力都实现不了，DeepMind ...
22 Nov 2019 - 呆板之心报导介入：一鸣、杜伟近日，DeepMind 以前时间发表于Nature 子刊的论文被严峻质疑。来自华为英国研发中央的研究者测验考试试验 ...

my.oschina.net › _blog_detail_list_rel ▾ Translate this page
前言摘要：2D 的俄罗斯方块已经被人玩烂了，突发奇想就做了个 ...
近日，DeepMind 之前时间发表在Nature 子刊的论文被严重质疑。来自华为英国研发中心的研究者尝试实验了DeepMind 的方法，并表示该论文需要的算力无法实现。

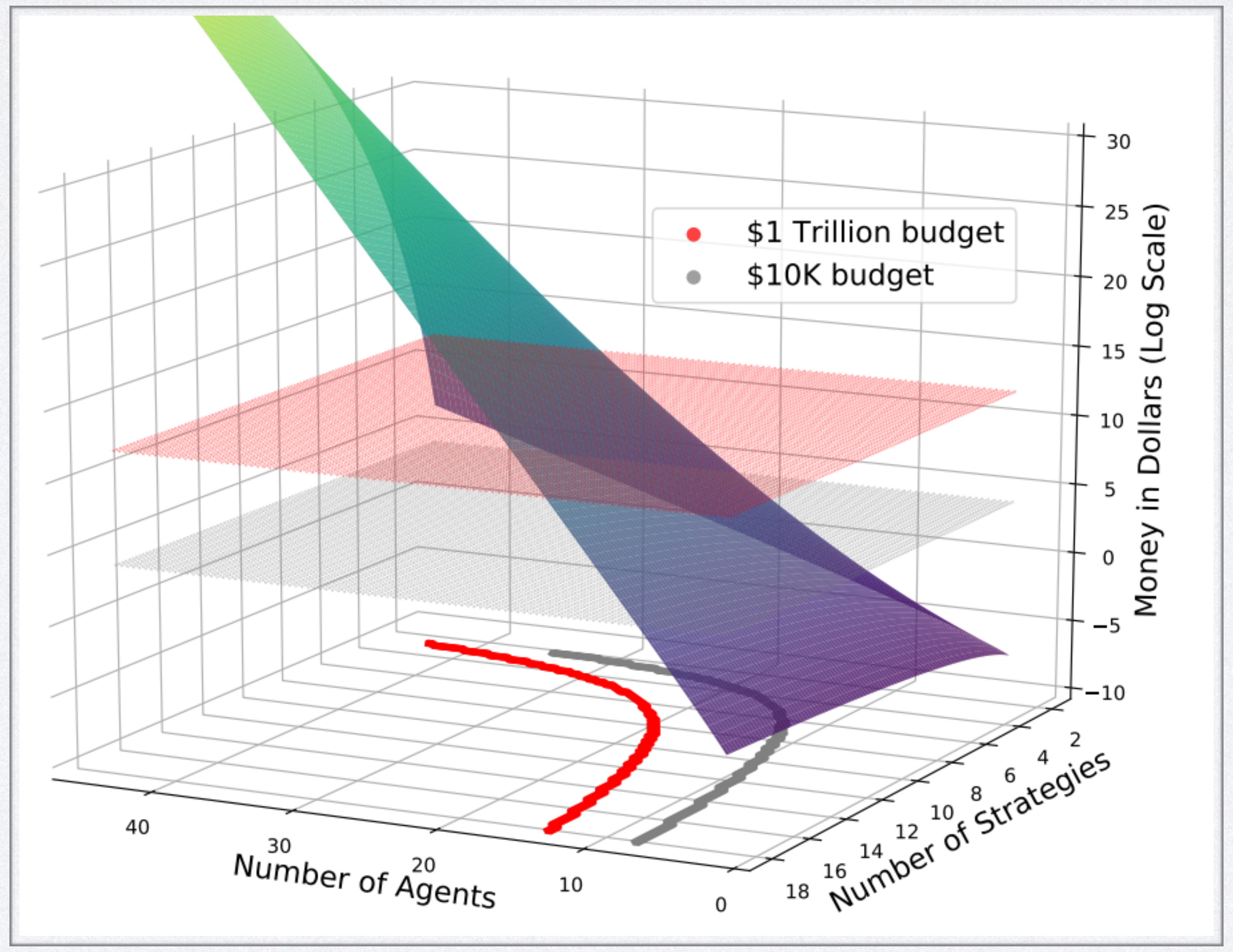
www.myzaker.com › channel - Translate this page
机器之心--ZAKER新闻：传递价值资讯
华为中兴再遭禁令：禁止采购，限期移除已有设备，美国联邦通信委员会引争议 ... 你的算法耗尽全球GPU算力都实现不了，DeepMind阿尔法系列被华为怒怼，曾 ...

kknews.cc › 科技
DeepMind阿爾法被華為怒懟 - 每日頭條
21 Nov 2019 - 來自華為英國研發中心的研究者嘗試實驗了DeepMind 的方法，並表示該 ... 你的算法耗盡全球GPU算力都實現不了，DeepMind阿爾法被華為怒懟.

1 million reads
10K comments

Game Env.	PetaFlop/s-days	Cost ($)	Time (days)
AlphaZero Go [29]	$1,413 \times 7$	$207M$	$1.9M$
AlphaGo Zero [28]	$1,181 \times 7$	$172M$	$1.6M$
AlphaZero Chess [29]	17×1	$352K$	$3.2K$
MuJoCo Soccer [18]	0.053×10	$4.1K$	72
Leduc Poker [15]	0.006×9	420	7
Kuhn Poker [11]	$< 10^{-4} \times 256$	< 1	–
AlphaStar [31]	$52,425$	$244M$	$1.3M$

Cost of Step 1



Cost of Step 2

Table 1: Time and space complexity comparison given N(number of agents) $\times$ k(number of strategies) table as inputs.

Method	Time	Memory
Power Method	$O(k^{N+1}N)$	$O(k^{N+1}N)$
PageRank	$O(k^{N+1}N)$	$O(k^{N+1}N)$
Eig. Decomp.	$O(k^{N\omega})$	$O(k^{N+1}N)$
Mirror Descent	$O(k^{N+1}\log k)$	$O(k^{N+1}N)$

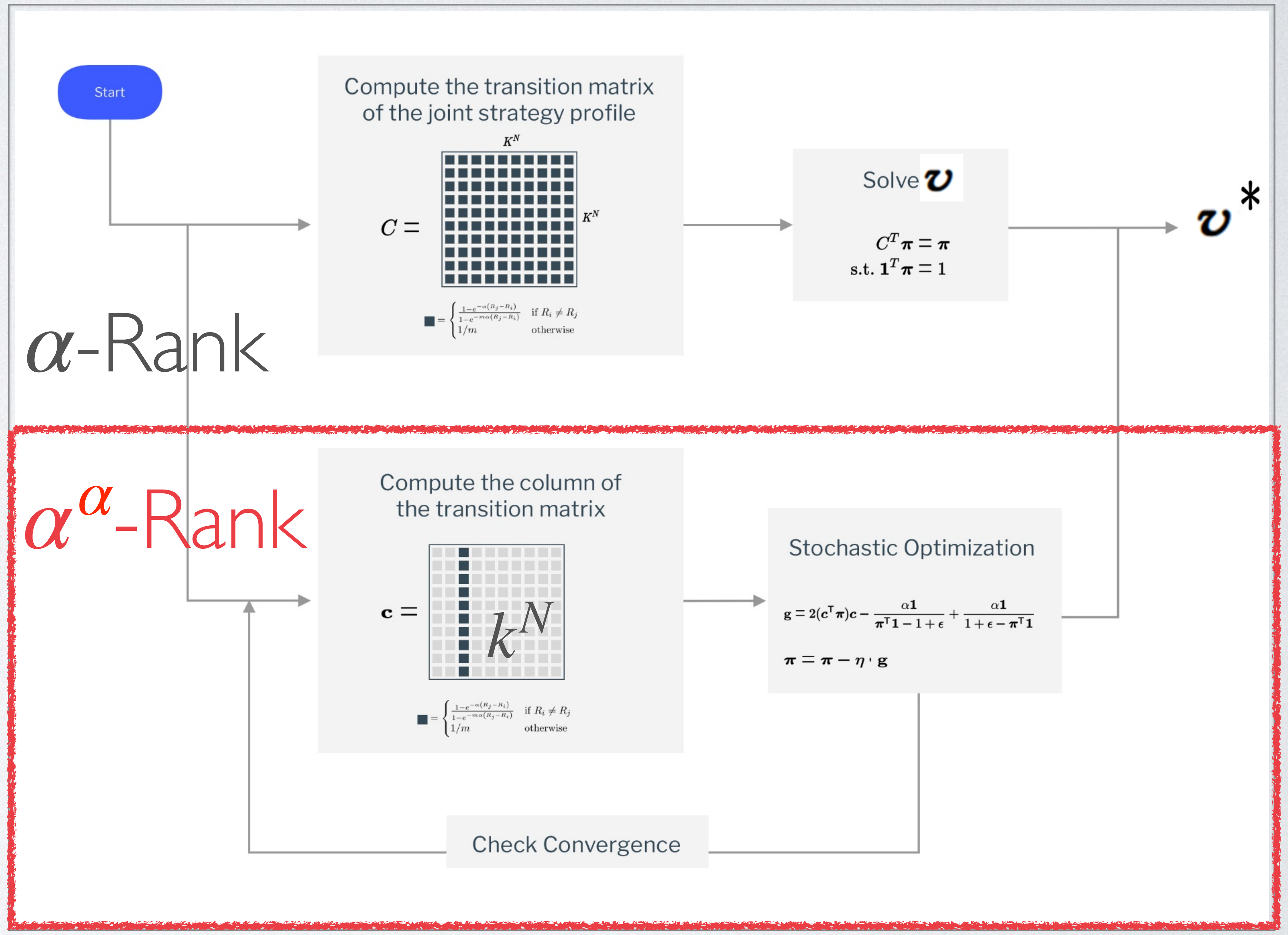
Cost of Step 3

α^{α}-Rank: A Scalable Solution for α-Rank [Yang 2020]

- Novelty 1: reformulate as a stochastic optimisation problem
  - Though cannot improve the time-complexity, but now can do early stopping for large meta-game solutions.
  - Saves time in getting the payoff values for the transition matrix of Markov chain.

Table 1: Time and space complexity comparison given N(number of agents) $\times$ k(number of strategies) table as inputs.

Method	Time	Memory
Power Method	$\mathcal{O}\left(k^{N+1}N\right)$	$\mathcal{O}\left(k^{N+1}N\right)$
PageRank	$\mathcal{O}\left(k^{N+1}N\right)$	$\mathcal{O}\left(k^{N+1}N\right)$
Eig. Decomp.	$\mathcal{O}\left(k^{N\omega}\right)$	$\mathcal{O}\left(k^{N+1}N\right)$
Mirror Descent	$\mathcal{O}\left(k^{N+1}\log k\right)$	$\mathcal{O}\left(k^{N+1}N\right)$



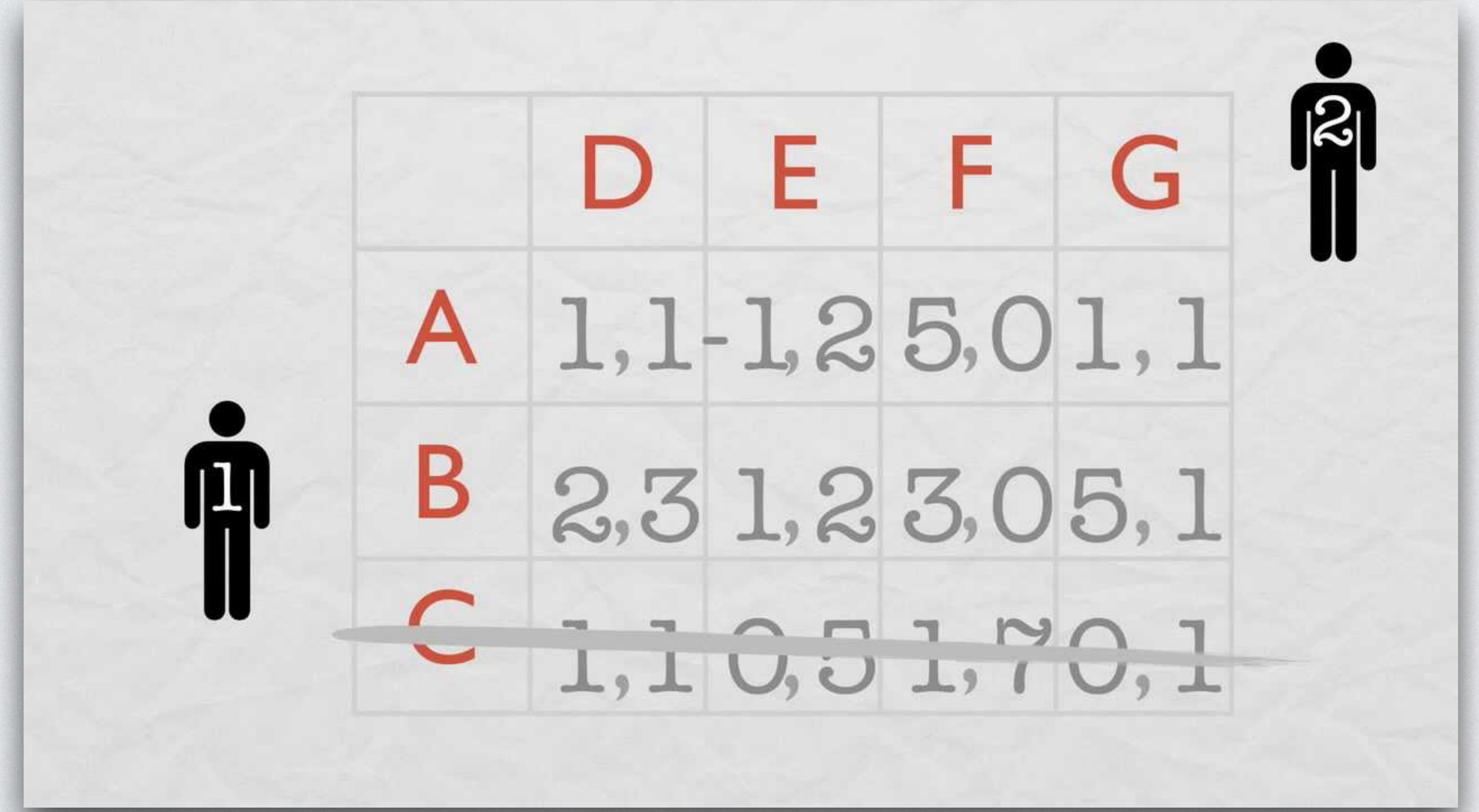
$$v = \lim_{t\to\infty} [T]^t v_0$$

$$\min_{v\in\mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n}\left(v^T c_i\right)^2 - \lambda \log\left(\delta^2 - \left[v^T \mathbf{1} - 1\right]^2\right) - \frac{\lambda}{n}\sum_{i=1}^{n}\log\left(v_i\right)$$

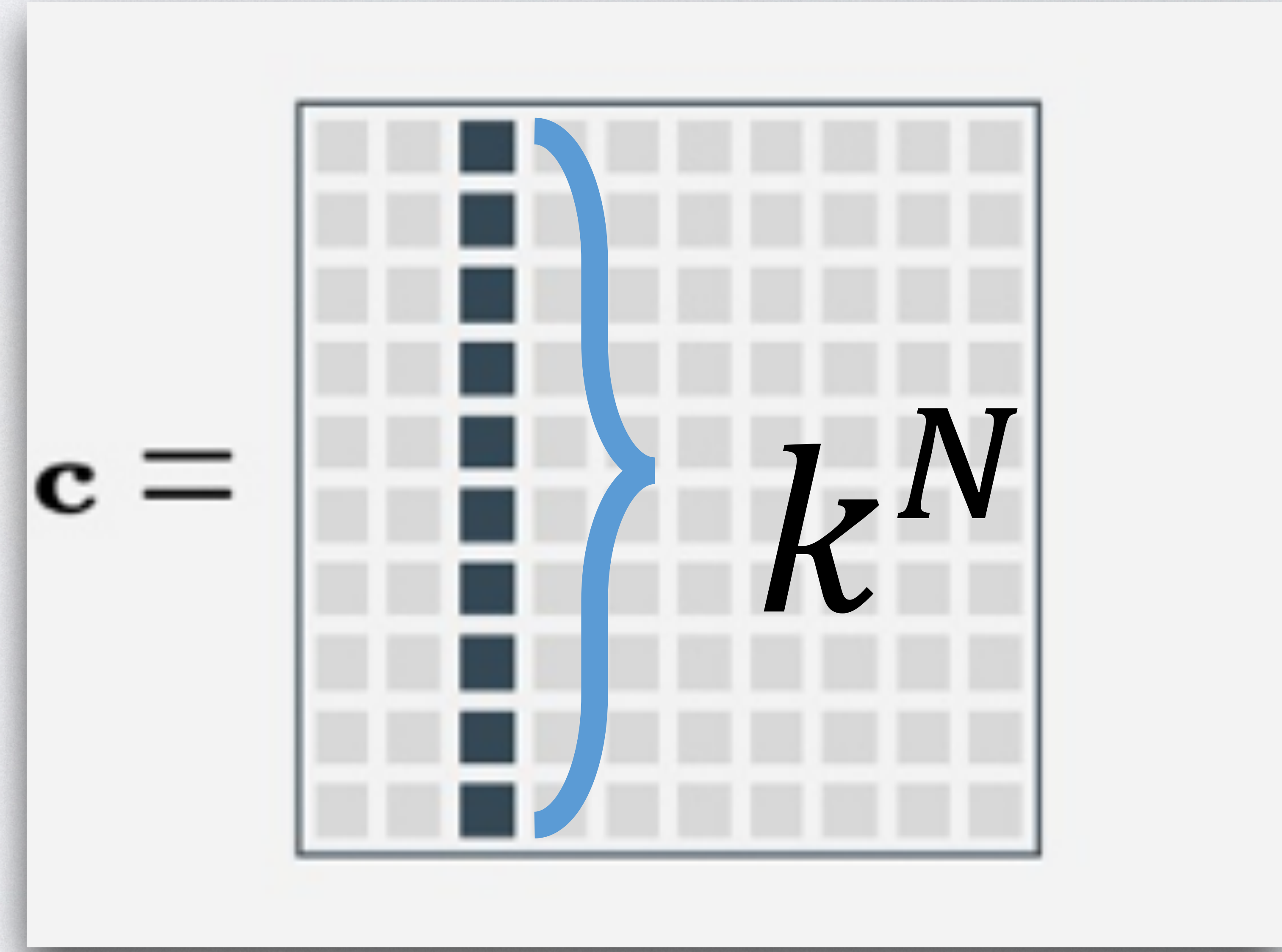
Adam/SGD/…

α^{α}-Rank: A Scalable Solution for α-Rank [Yang 2020]

- Novelty 2: Introducing a heuristics to start with a subset of strategies and then increasingly expand the strategy space of each agent, we can decrease k further.
    - Intuition: remove dominated strategy from the beginning and save the exploration time, and add any good strategy back if we miss them wrongly in the initialisation.



All joint strategy profile involving “C" will not be SSCC, removing “C” can save exploration time.



Still exponential size, make k smaller

Scalability of α^{α}-Rank on Large Meta-games

Random matrices

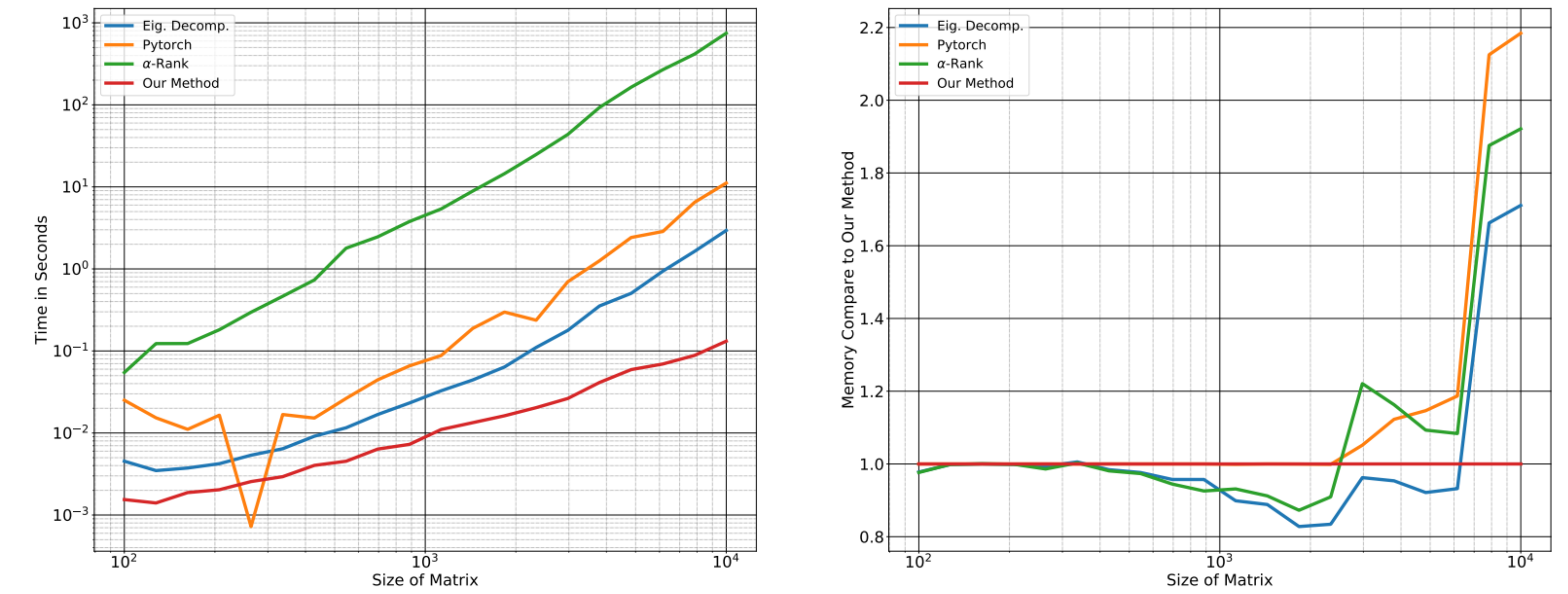
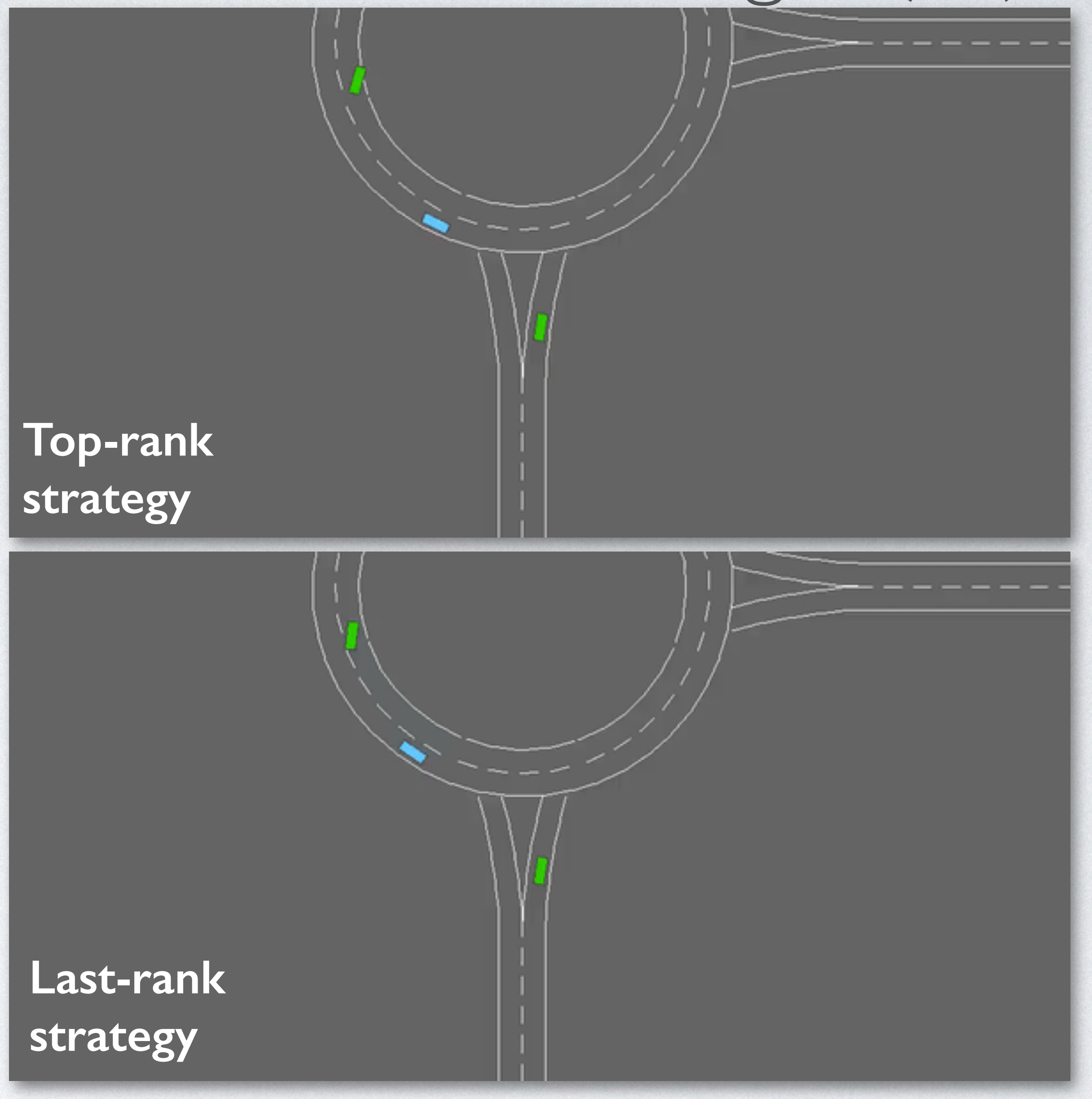
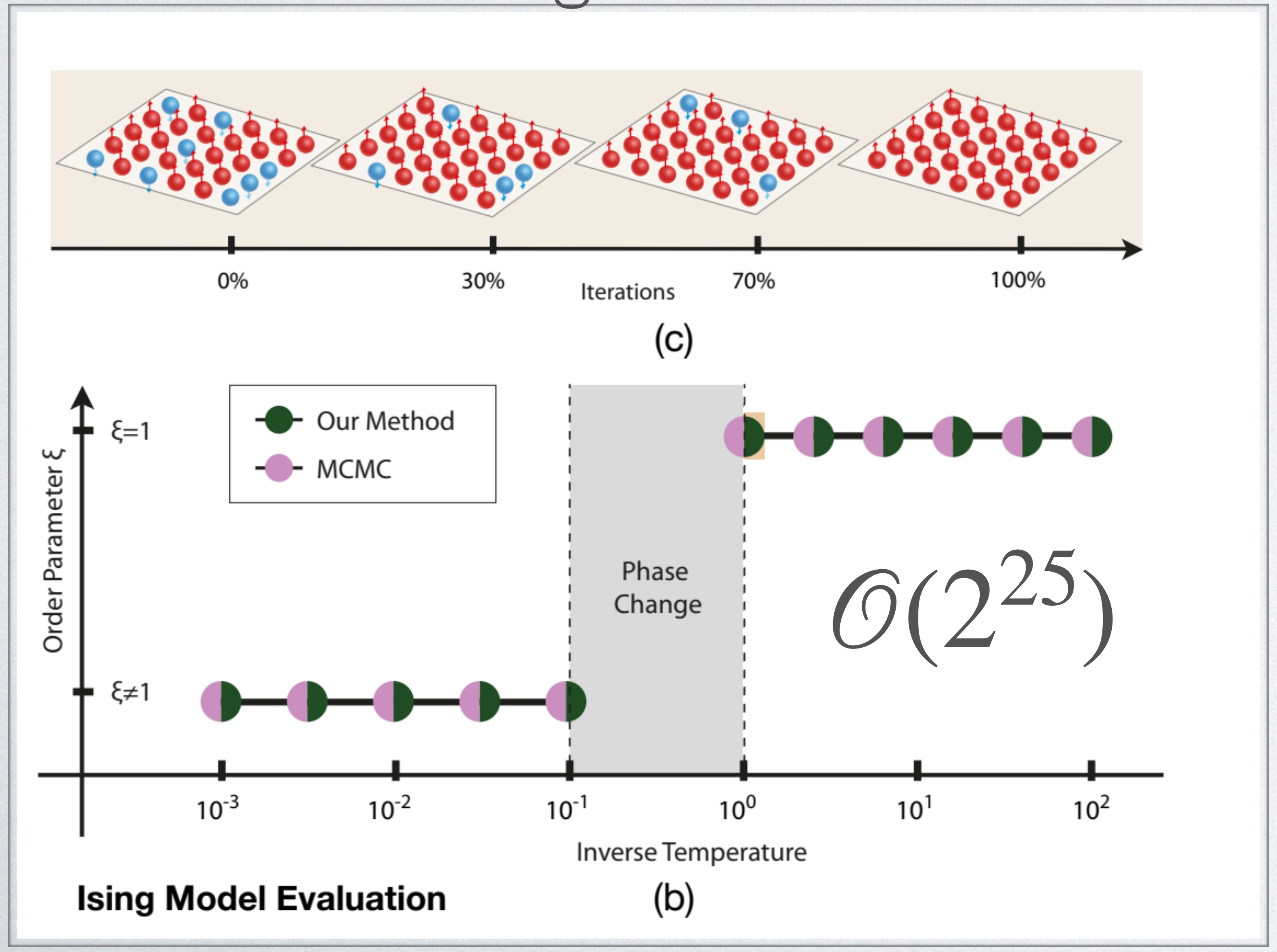


Figure 4: Comparisons of time and memory complexities on varying sizes of random matrices.

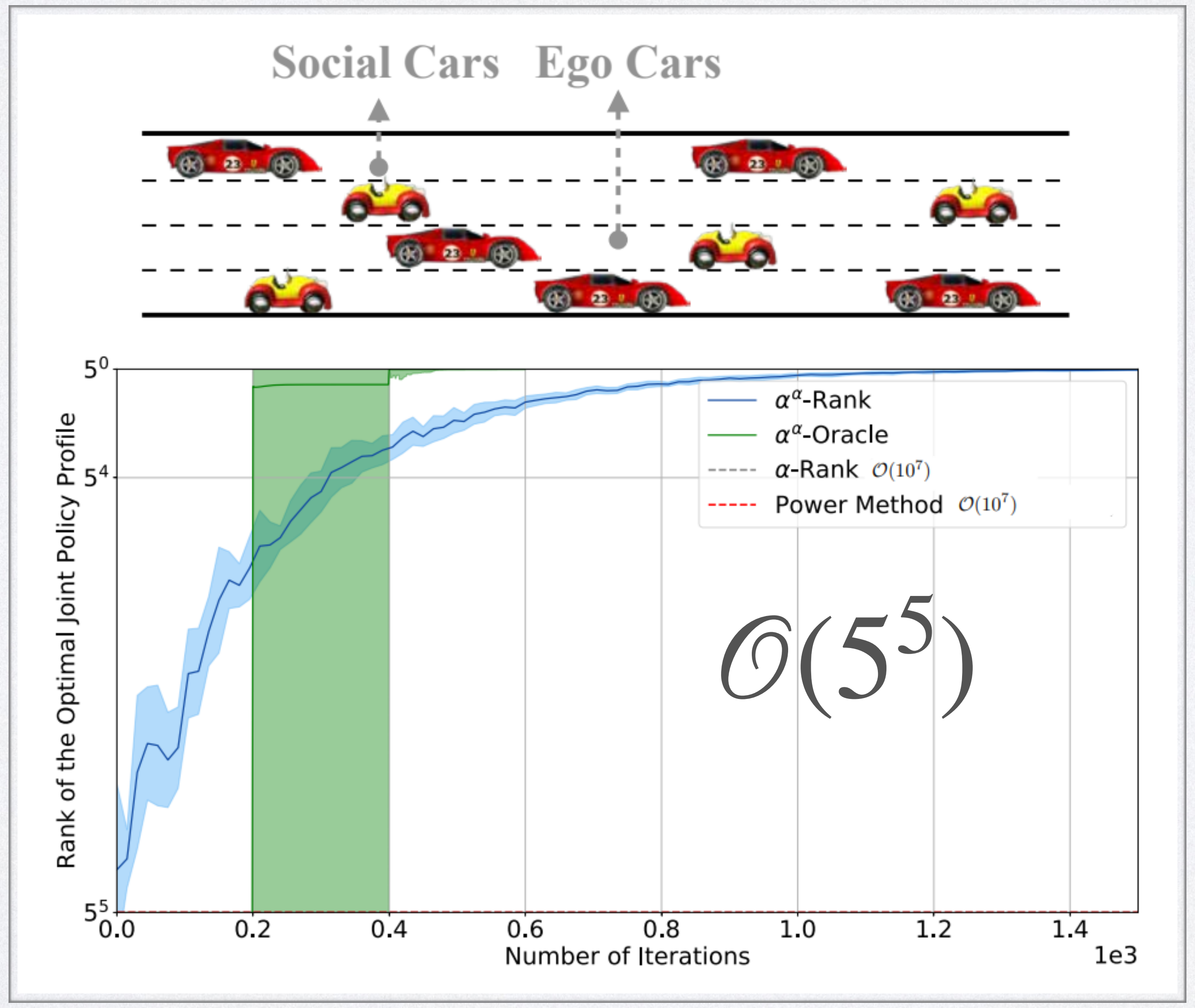
Roundabout driving $\mathcal{O}(5^3)$



Ising Model



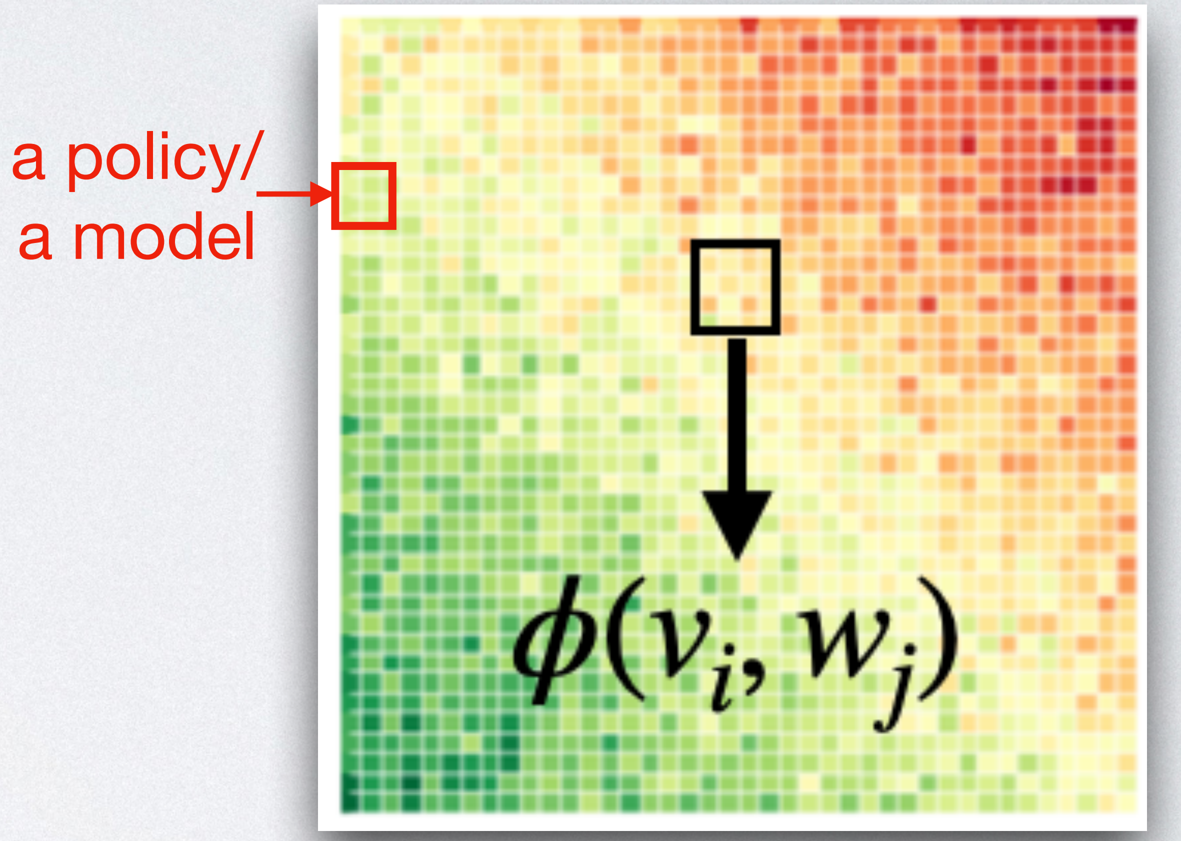
Autonomous driving



Highway Driving $\mathcal{O}(10^5)$

Summary of Meta-game Policy Evaluation

- Give a meta-game with fixed set of players and strategies, we have introduced methods to answer the questions of which joint strategy profile is "optimal", specifically, we can know
  - what is definition of "optimality"
  - which metric suits transitive/in-transitive games
  - which metric is tractable in multi-player games
  - which metric can deal with general-sum games
  - which metric can induce stable equilibrium
  - which metric can induce unique equilibrium
  - which metric can model the flow of dynamics or being a fixed point

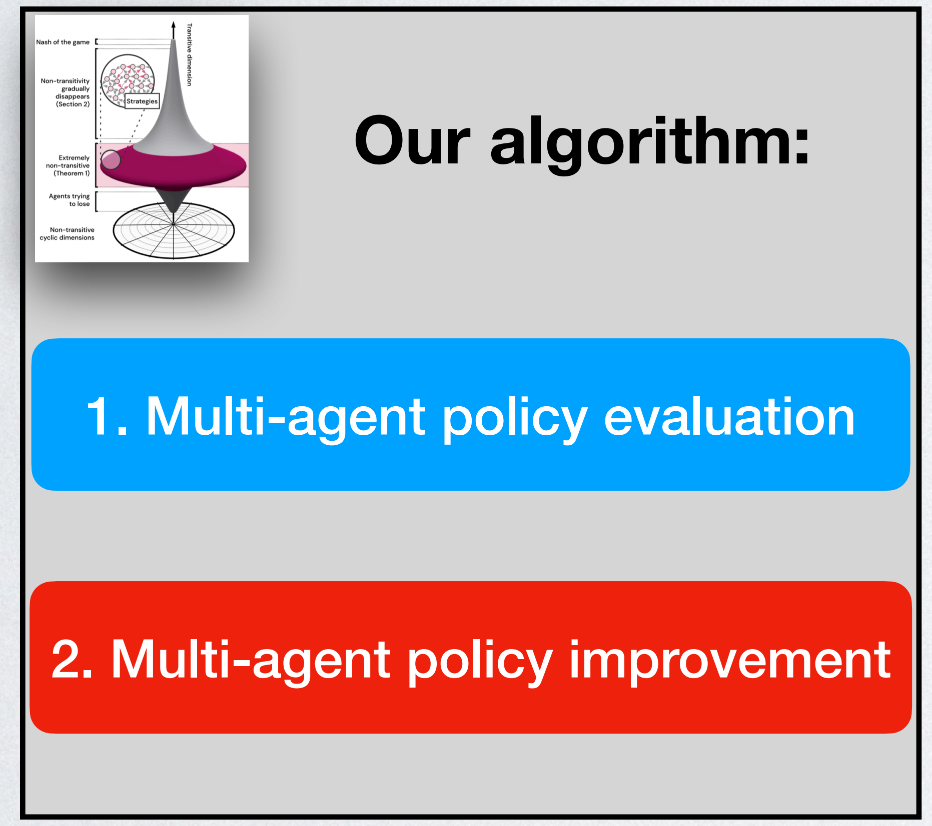


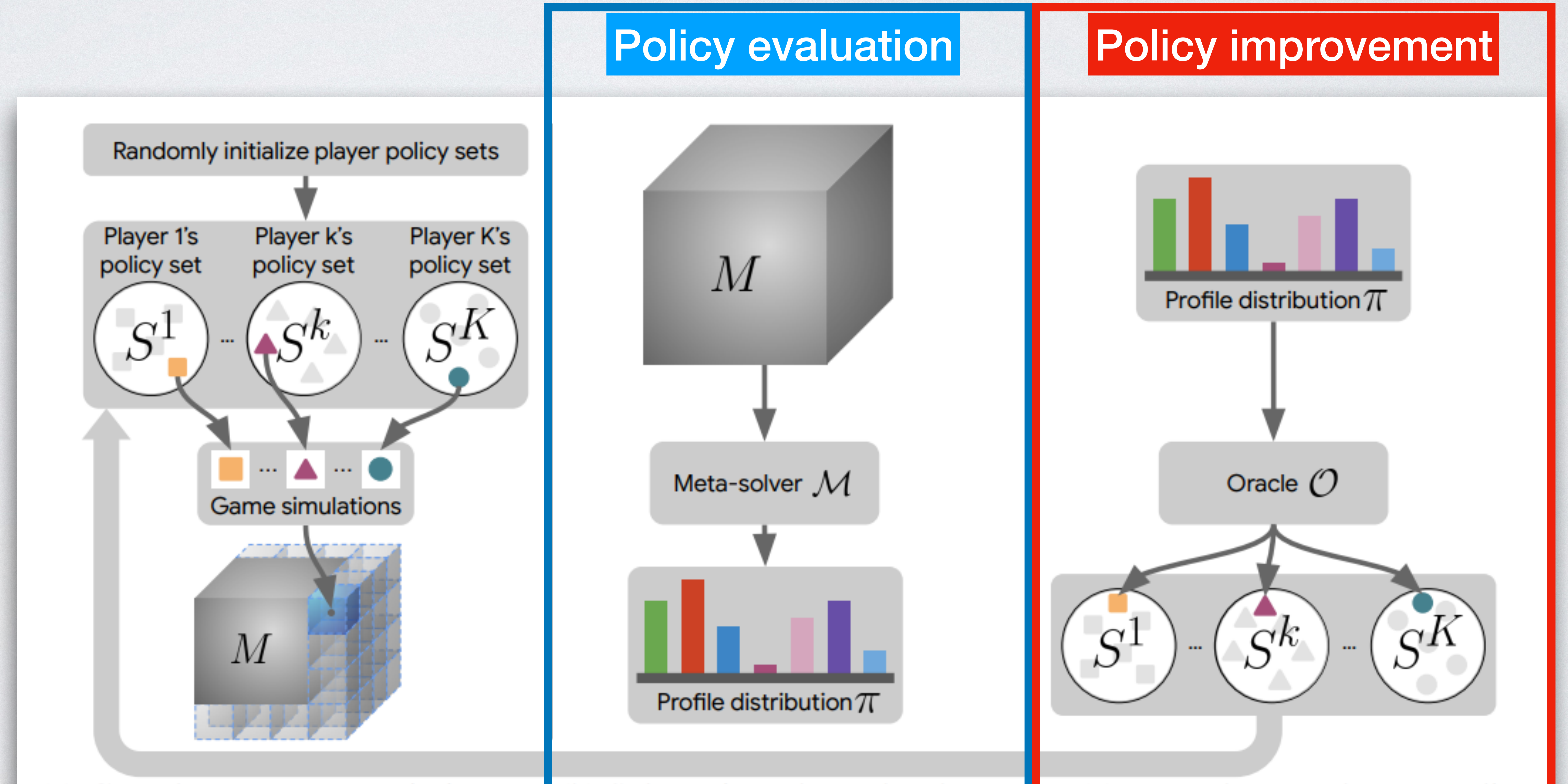
Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☑ When self-play does not work
  - ☑ The landscape of real-world games
  - ☑ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☑ Elo rating
  - ☑ Nash Equilibrium
  - ☑ Replicator dynamics
  - ☑ α-Rank & α^α-Rank
- **Policy Improvement in Meta-games**
  - ☐ Iterated self-plays
  - ☐ Fictitious play & generalised weaken fictitious play
  - ☐ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

Policy Improvement on Meta-games

- The next step is to develop new policies based on the existing policies in the pool.
- Combining both gives us generalised approaches of multi-agent learning for games.





(a) Complete: compute missing payoff tensor $\boldsymbol{M}$ entries via game simulations.

(b) Solve: given the updated payoff tensor $\boldsymbol{M}$, calculate meta-strategy $\boldsymbol{\pi}$ via meta-solver $\mathcal{M}$.

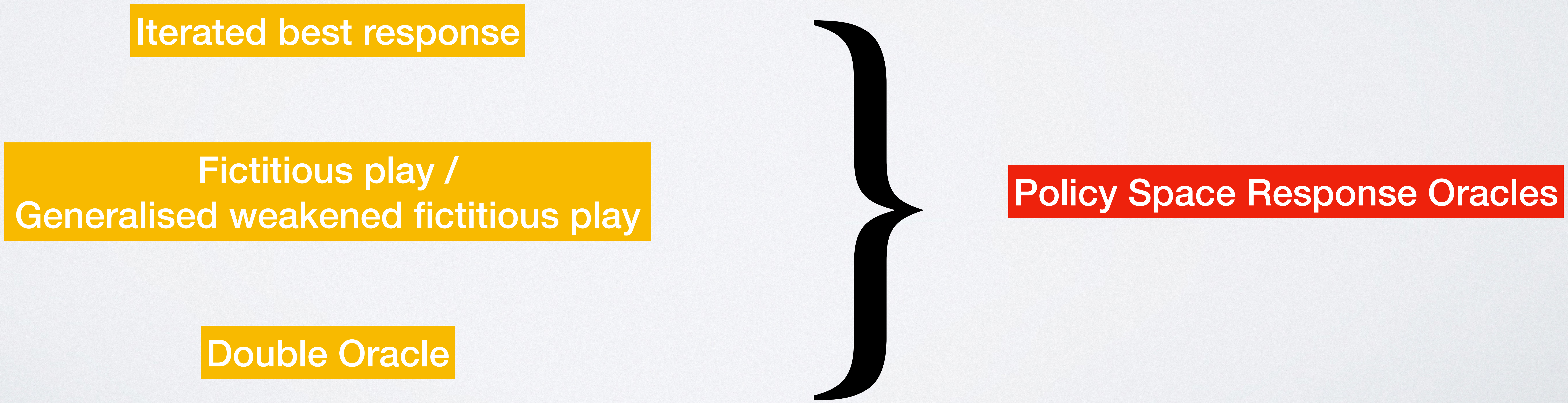
(c) Expand: append a new policy to each player's policy space using the oracle $\mathcal{O}$.

Figure 1: Overview of PSRO($\mathcal{M}$, $\mathcal{O}$) algorithm phases.

[Muller 2020]

Policy Improvement on Meta-games

- Let's understand first how to do policy improvement on normal-form games where the total number of strategy is known in advance, the best response is equivalent to a search problem, e.g., Rock-Paper-Scissor. Two famous types of algorithms are fictitious play & double oracle.
- Then we move on to the general cases where there are infinite number of strategies, and at each iteration we have to use RL algorithms to find a new policy that is the best response, e.g. Poker, Go. This gives us Policy Space Response Oracles (PSRO).



Iterated Best Response

- Recall that we have seen a naive approach to do policy improvement via self-play.
- Given the best response is defined by

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

- Self-play is essentially doing best response in an iterated way, given the opponent's latest policy π^{-i}, find a best response, e.g., through an RL algorithm.

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

- Self-play only focuses on responding to the opponent's latest strategy, which can lead to cyclic behaviours in in-transitive games.
- A better way is to look at the historical actions, which is the fictitious play.

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 distribution.

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

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

- It guarantees to converge, in terms of the Nash value, in two-player zero-sum games, and, potential games which include fully-cooperative games.

- 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.

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

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

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

- Recovers normal Fictitious Play when $\alpha^t = 1/t, \epsilon_t = 0, M_t = 0$.

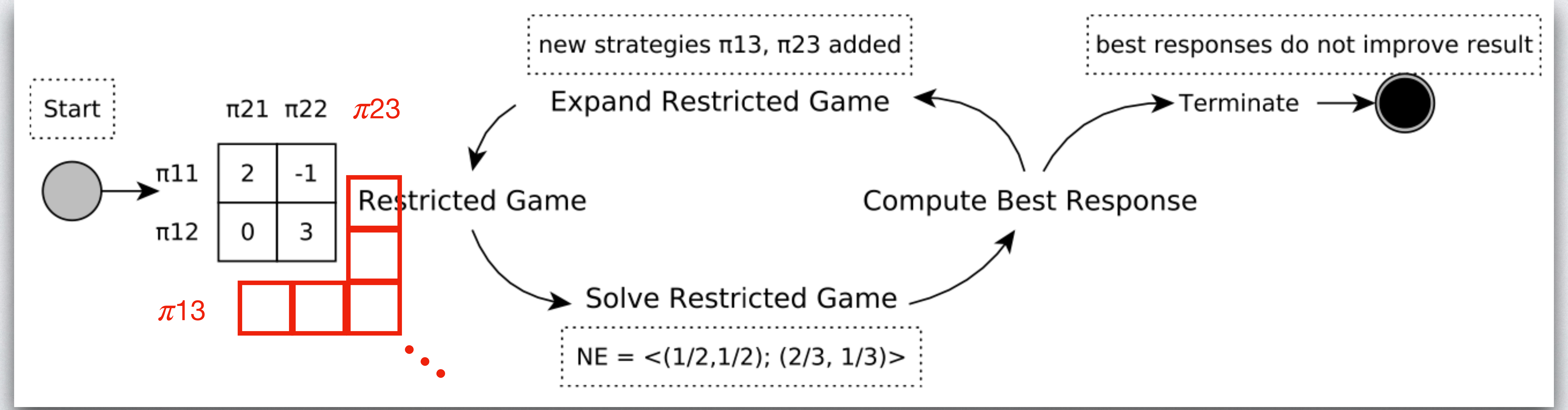
- **Why important:** it allows us to use a broad class of best responses such as RL algorithms, and also, the policy exploration, e.g., the entropy term in soft-Q learning, can now be considered through the M term.

Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☑ When self-play does not work
  - ☑ The landscape of real-world games
  - ☑ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☑ Elo rating
  - ☑ Nash Equilibrium
  - ☑ Replicator dynamics
  - ☑ α-Rank & α^{α}-Rank
- **Policy Improvement in Meta-games**
  - ☑ Iterated self-plays
  - ☑ Fictitious play & generalised weaken fictitious play
  - ☐ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

Double Oracle [McMahan 2003]

- Double Oracle is also an iterated best response method, the difference is that, it best responds to the opponent's Nash equilibrium at each iteration.
- If the newly added best response is already in the strategy pool, then terminate.
- Why "oracle": the search for a best response is guided by an oracle.
- It guarantees to converge to minimax equilibrium in finite games.



[Bosansky 2016]

Double Oracle [McMahan 2003]

- Example on solving RPS games.
- Agents are initialised with only a subset of the all strategies, the intuition is that they can solve the game before seeing all strategies of the game. In the worst-case scenario, it recovers to solve the original game.

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

- **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)

Time comparison to Linear Program

Table 1. Sample problem discretizations, number of sensor placements available to the opponent, solution time using Equation 4, and solution time and number of iterations using the Double Oracle Algorithm.

	grid size	k	LP	Double	iter
A	54 x 45	32	56.8 s	1.9 s	15
B	54 x 45	328	104.2 s	8.4 s	47
C	94 x 79	136	2835.4 s	10.5 s	30
D	135 x 113	32	1266.0 s	10.2 s	14
E	135 x 113	92	8713.0 s	18.3 s	30
F	269 x 226	16	-	39.8 s	17
G	269 x 226	32	-	41.1 s	15

Policy Space Response Oracle [Lanctot 2017]

- A generalisation of double oracle methods on meta-games.
- Given opponents' existing Nash meta-policies, 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 $(\Pi_1^0, \ldots, \Pi_1^T)$, PSRO generalises all previous methods by setting different forms of 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: $\sigma_{-i} = \mathbf{Nash}\left(\Pi^{T-1}, U\right)$ or $\mathbf{RD}\left(\Pi^{T-1}, U\right)$

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 $\sigma_i = \text{UNIFORM}(\Pi_i)$

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

BTW, some MARL Techniques and its Deep Counterparts

Foundational Algorithm	Modern and/or Deep RL Counterpart
Fictitious Play [Brown, 1951]	Extensive-form Fictitious Play [Heinrich et al., 2015] Neural Fictitious Self-Play [Heinrich & Silver, 2016]
Independent Q-learning [Tan, 1993]	Multi-agent Deep Q-Networks [Tampuu et al., 2015]
Double Oracle [McMahan et al., 2003]	Policy-Space Response Oracles [Lanctot et al., 2017]
Hysteretic Q-learning [Matignon et al., 2007]	Recurrent Hysteretic Q-Networks [Omidshafiei et al., 2017]
Extended Replicator Dynamics [Tuyls et al., 2003]	Learning with Opponent-Learning Awareness [Foerster et al., 2017]
Lenient Learning [Panait et al., 2006; Panait, Tuyls, Luke, 2008]	Lenient Deep Q-Networks [Palmer, Tuyls et al., 2018]
Replicator Dynamics [Taylor & Jonker, 1978; Smith, 1982; Schuster & Sigmund, 1983]	Neural Replicator Dynamics [Omidshafiei et al., 2019]

[DeepMind MAS tutorial # 219]

Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☑ When self-play does not work
  - ☑ The landscape of real-world games
  - ☑ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☑ Elo rating
  - ☑ Nash Equilibrium
  - ☑ Replicator dynamics
  - ☑ α-Rank & α^α-Rank
- **Policy Improvement in Meta-games**
  - ☑ Iterated self-plays
  - ☑ Fictitious play & generalised weaken fictitious play
  - ☑ Double oracle & PSRO
  - ☐ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

Retrospection on the Naive Self-play Approach

Only work in transitive games

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

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

Solution for in-transitive games

key changes: instead of only looking at the latest opponent's policy, best responding to the Nash combination of its set of policies

Algorithm 3 Response to Nash (PSRO$_\text{N}$)

input: population $\mathfrak{P}_1$ of agents
for $t = 1, \ldots, T$ **do**
  $\mathbf{p}_t \leftarrow$ Nash on $\mathbf{A}_{\mathfrak{P}_t}$
  $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \sum_{\mathbf{w}_i \in \mathfrak{P}_t} \mathbf{p}_t[i] \cdot \phi_{\mathbf{w}_i}(\bullet)\right)$
  $\mathfrak{P}_{t+1} \leftarrow \mathfrak{P}_t \cup \{\mathbf{v}_{t+1}\}$
end for
output: $\mathfrak{P}_{T+1}$

[Balduzzi 2019]

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

Rectifying Nash for Diversity [Balduzzi 2019]

PSRO-Rectified-Nash: promoting diversity in PSRO

key changes: only selecting opponents that I have already won over, further rectifying the Nash equilibrium

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

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: $\mathfrak{P}_{T+1}$

[Balduzzi 2019]

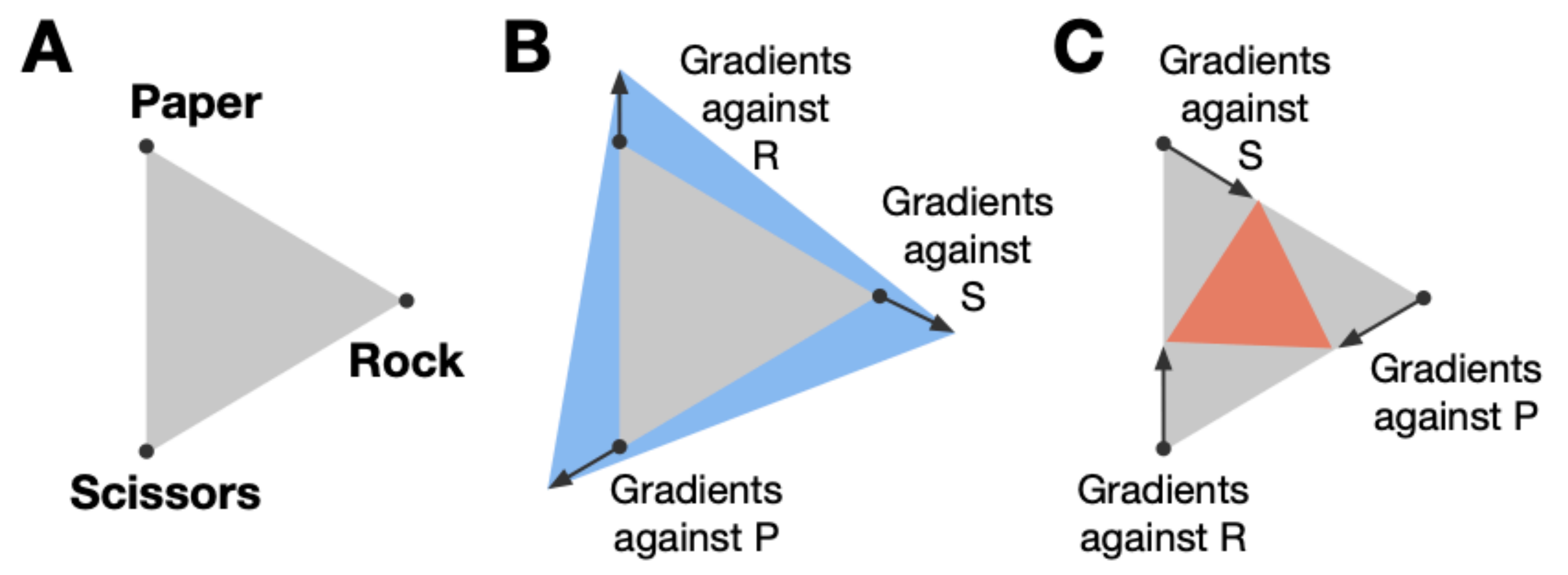
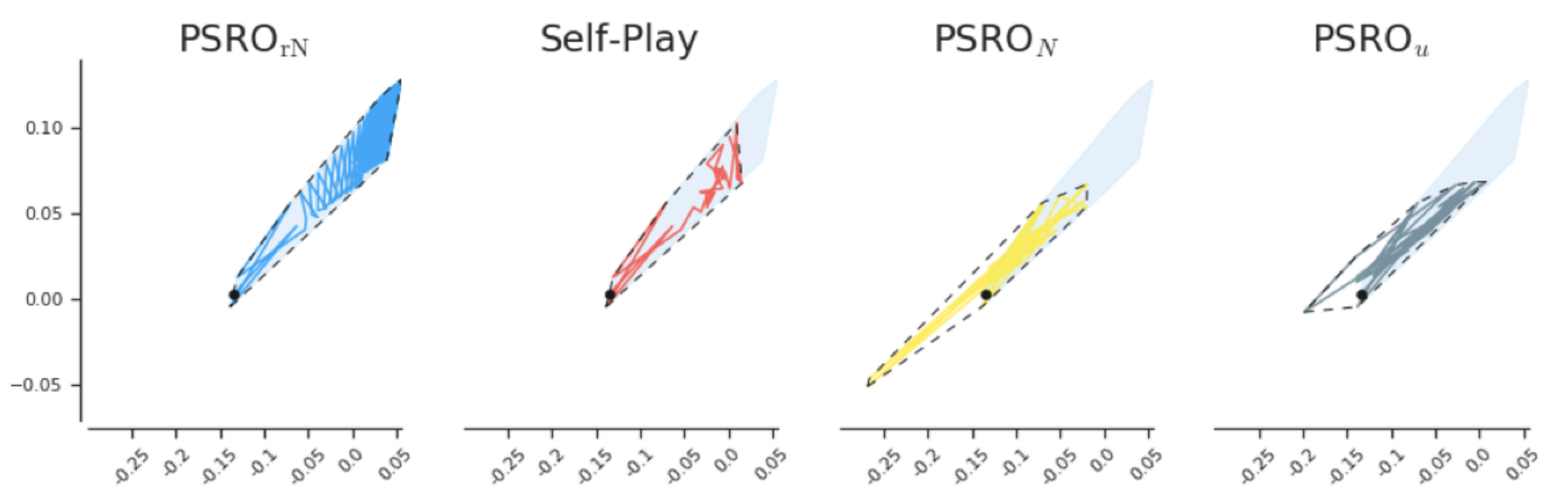


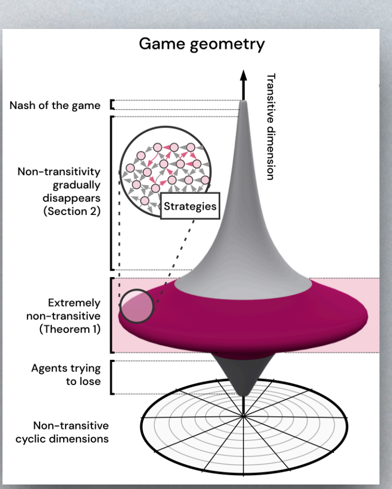
Figure 3. **A:** Rock-paper-scissors. **B:** Gradient updates obtained from PSRO$_{\text{rN}}$, amplifying strengths, grow gamescape (gray to blue). **C:** Gradients obtained by optimizing agents to reduces their losses shrink gamescape (gray to red).

Intuition: maintaining strength can keep exploring larger and large strategy space (强者恒强/马太效应)



diversity can also help explore the strategy space more efficiently and effectively

My Comments on Modelling Diversity



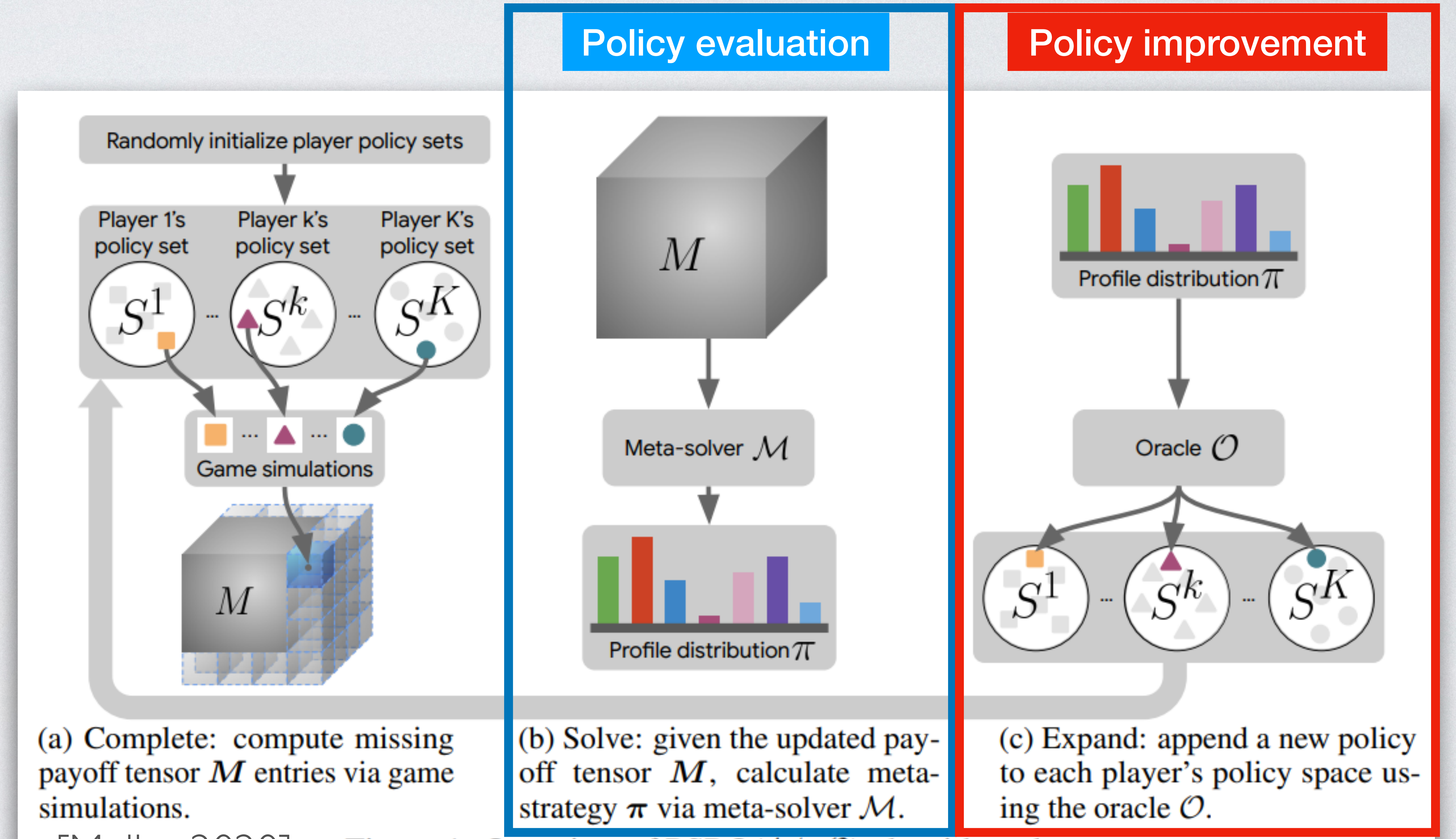
- Diversity matters because the more diverse your strategy pool is, the more un-exploitable you are. Promoting diversity can help you walk out of the in-transitive region faster.
- In real-world AI applications, you want your policies to be diverse enough, covering different skill levels. This is a realistic need in autonomous driving and Gaming AI.
- It is also a hot research topic. How to add diversity on the meta-game level is still unclear yet. Existing approaches are mainly based on heuristics. One cannot solve by simply adding an entropy term, because the diversity is among the policies in a meta-game.
- PSRO-Rectified-Nash suggests to compare more against losers, but the prioritised fictitious play in AlphaStar suggests completely the opposite. They contradict!
- A very promising direction is to use Determinantial Point Process, see [Yang 2020b], which is theoretically grounded in modelling repulsive particles from physics.

What We Have Learned so far

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

Black-box multi-agent game engine

input



(a) Complete: compute missing payoff tensor $\boldsymbol{M}$ entries via game simulations. (b) Solve: given the updated payoff tensor $\boldsymbol{M}$, calculate meta-strategy $\boldsymbol{\pi}$ via meta-solver $\mathcal{M}$. (c) Expand: append a new policy to each player's policy space using the oracle $\mathcal{O}$.

[Muller 2020] Figure 1: Overview of PSRO($\mathcal{M}$, $\mathcal{O}$) algorithm phases.

output π^*

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

Policy evaluation:
- Elo rating
- Nash equilibrium
- Replicator dynamics
- α-Rank/α^α-Rank

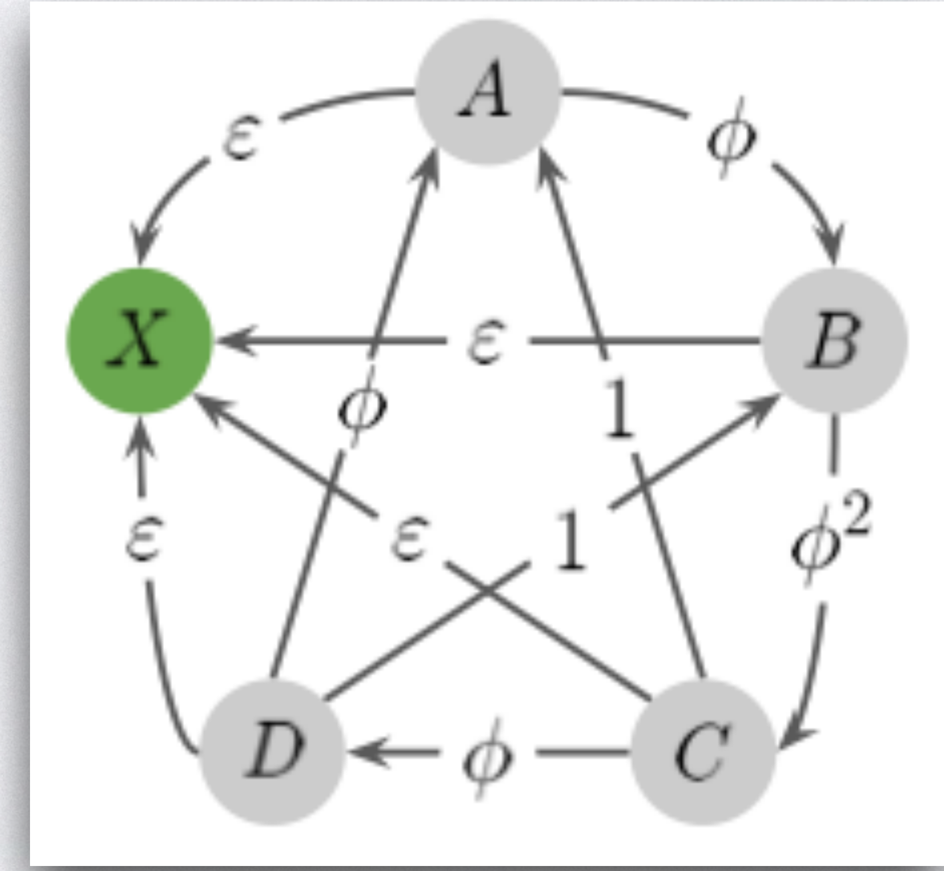
Policy improvement:
- iterated best response
- fictitious play
- double oracle
- **PSRO**
- PSRO-Nash/
- PSRO-Rectified-Nash

PSRO-α-Rank: A Generalised Approach for Multi-agent Training

- PSRO-Nash limitation: it relies on solving Nash, which is challenging in general cases.
- A generalised approach is expected to solve more than two-player zero-sum games.
- Remember the α-Rank benefits: multi-player, general-sum, tractable, unique.
- However, best response + α-Rank simply does not converge to SSCC.
- See the counter-example in [Muller 2020].

Player 1 \ Player 2	A	B	C	D	X
A	0	$-\phi$	1	ϕ	$-\varepsilon$
B	ϕ	0	$-\phi^2$	1	$-\varepsilon$
C	-1	ϕ^2	0	$-\phi$	$-\varepsilon$
D	$-\phi$	-1	ϕ	0	$-\varepsilon$
X	ε	ε	ε	ε	0

zero-sum game, $0 < \epsilon \leq 1, \phi > 1$



response graph, X is the only SSCC

Note: a node should be a joint strategy, since the strategy set is the same for both agents, we only write for player 1

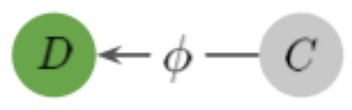
A Counter-example of Best Response + α-Rank

- Best response + α-Rank may not recover the SSCC of the response graph.
- For bad initialisation, the best response will be trapped in bad "local" strategy subset.

Iteration 1

Player 2

		A	B	C	D	X
Player 1	A	0	$-\phi$	1	ϕ	$-\varepsilon$
	B	ϕ	0	$-\phi^2$	1	$-\varepsilon$
	C	-1	ϕ^2	0	$-\phi$	$-\varepsilon$
	D	$-\phi$	-1	$\boldsymbol{\phi}$	0	$-\varepsilon$
	X	ε	ε	ε	ε	0

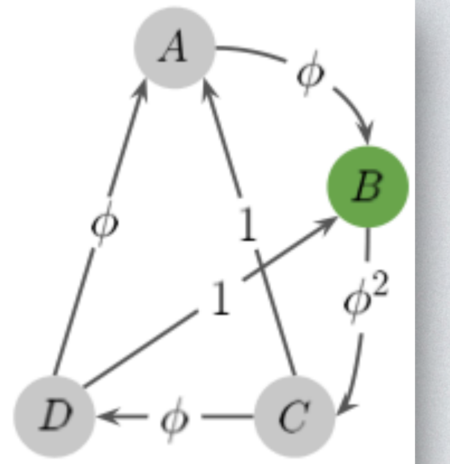


(b) Consider an initial strategy space consisting only of the strategy C; the best response against C is D.

Iteration 3

Player 2

		A	B	C	D	X
Player 1	A	0	$-\phi$	1	ϕ	$-\varepsilon$
	B	$\boldsymbol{\phi}$	0	$-\phi^2$	1	$-\varepsilon$
	C	-1	ϕ^2	0	$-\phi$	$-\varepsilon$
	D	$-\phi$	-1	ϕ	0	$-\varepsilon$
	X	ε	ε	ε	ε	0

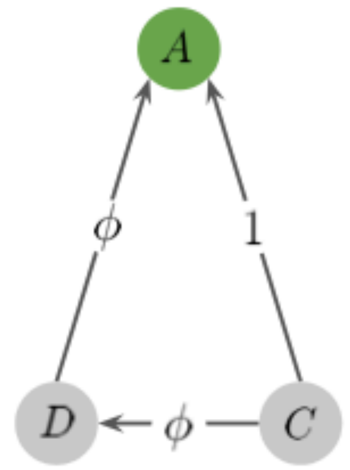


(d) The α-Rank distribution over $\{C, D, A\}$ puts all mass on A; the best response against A is B.

Iteration 2

Player 2

		A	B	C	D	X
Player 1	A	0	$-\phi$	1	$\boldsymbol{\phi}$	$-\varepsilon$
	B	ϕ	0	$-\phi^2$	1	$-\varepsilon$
	C	-1	ϕ^2	0	$-\phi$	$-\varepsilon$
	D	$-\phi$	-1	ϕ	0	$-\varepsilon$
	X	ε	ε	ε	ε	0

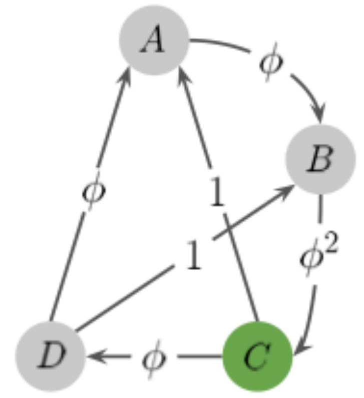


(c) The α-Rank distribution over $\{C, D\}$ puts all mass on D; the best response against D is A.

Iteration 4 - END

Player 2

		A	B	C	D	X
Player 1	A	0	$-\phi$	1	ϕ	$-\varepsilon$
	B	ϕ	0	$-\phi^2$	1	$-\varepsilon$
	C	-1	$\boldsymbol{\phi^2}$	0	$-\phi$	$-\varepsilon$
	D	$-\phi$	-1	ϕ	0	$-\varepsilon$
	X	ε	ε	ε	ε	0



(e) The α-Rank distribution over $\{C, D, A, B\}$ puts mass $(1/3, 1/3, 1/6, 1/6)$ on (A, B, C, D) respectively. For ϕ sufficiently large, the payoff that C receives against B dominates all others, and since B has higher mass than C in the α-Rank distribution, the best response is C.

best response of C is C, so it terminates

α-PSRO: A Bespoke PSRO for α-Rank

- Standard best response:

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

- Preference-based Best Response (PBR): make the oracle return strategies that will receive highest mass in the response graph of α-Rank when added to the population.

$$\mathbf{PBR}_i\left(\pi^{-i}\right) \subseteq \arg\max_{a \in S^1} \mathbf{E}_{a^{-i} \sim \pi^{-i}} \left[\mathbf{1}\left[R^i(a, a^{-i}) > R^i(a^i, a^{-i}) \right] \right]$$

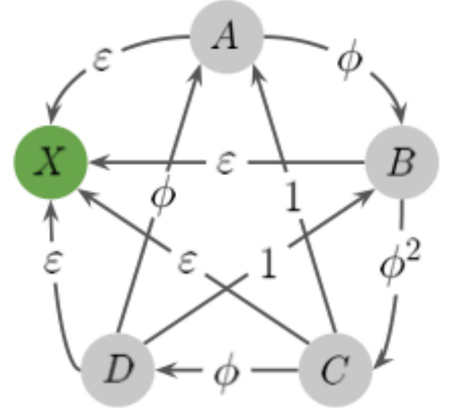
count which node has the largest input probability weights of SSCC

- If there exists multiple SSCC, then run PBR for every SSCC and return multiple PBRs.
- Back to the previous example:

suppose π^{-i} = (1/3, 1/3, 1/6, 1/6) on {A, B, C, D}
A beats C/D: 1/6 + 1/6 = 1/3
B beats A/D: 1/3 + 1/6 = 1/2
C beats B: 1/3
D beats C: 1/6
X beats ABCD: 1
PBR is X, add X into the response graph.

Player 2

Player 1	A	B	C	D	X
A	0	$-\phi$	1	ϕ	$-\varepsilon$
B	ϕ	0	$-\phi^2$	1	$-\varepsilon$
C	-1	ϕ^2	0	$-\phi$	$-\varepsilon$
D	$-\phi$	-1	ϕ	0	$-\varepsilon$
X	$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}$	0

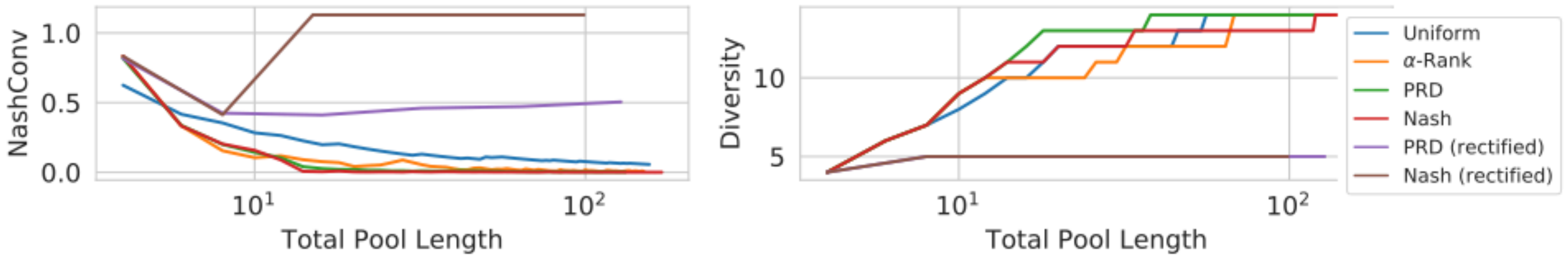


(e) The α-Rank distribution over $\{C, D, A, B\}$ puts mass $(1/3, 1/3, 1/6, 1/6)$ on (A, B, C, D) respectively. A beats C and D, and therefore its PBR score is $1/3$. B beats A and D, therefore its PBR score is $1/2$. C beats B, its PBR score is therefore $1/3$. D beats C, its PBR score is therefore $1/6$. Finally, X beats every other strategy, and its PBR score is thus 1. There is only one strategy maximizing PBR, X, which is then chosen, and the SSCC of the game, recovered.

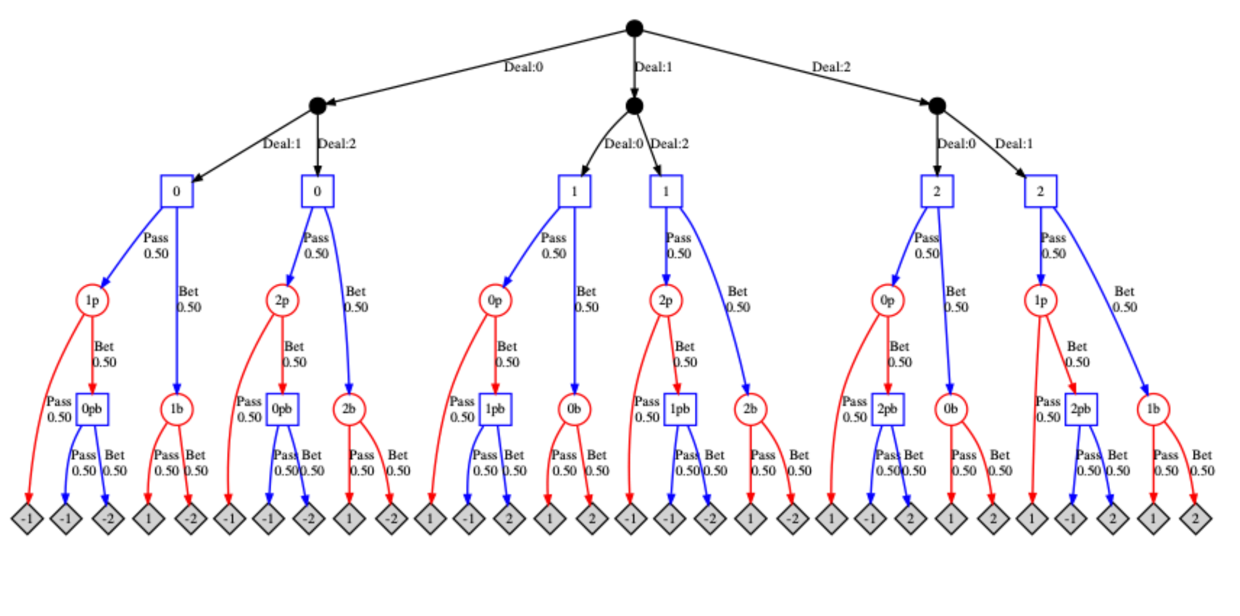
Testing Beds for PSRO-related Methods

- A common benchmark is Kuhn Poker, and Leduc Poker via [OpenSpiel].
- They are the "MNIST" for multi-agent gaming AI design. StarCraft is too heavy for testing.
- The metric is the called NashConv/exploitability/distance to Nash. Unbeatable if reaching 0.

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



(a) Kuhn poker Exploitability. (b) Kuhn poker Diversity. [Muller 2020]



(a) Initial (uniform) policies.

- Caveat: there are content missing about the sequence form of extensive-form games in this lecture, readers are recommended to read extensive-form fictitious play first [Heinrich 2015].

Contents

- **Recap of Past Lectures**
  - ☑ Multi-agent learning basics
  - ☑ Tractability of multi-agent problems
- **Multi-agent Learning for Games**
  - ☑ Motivation of studying games
  - ☑ When self-play does not work
  - ☑ The landscape of real-world games
  - ☑ The necessity of studying meta-games
- **Policy Evaluation in Meta-games**
  - ☑ Elo rating
  - ☑ Nash Equilibrium
  - ☑ Replicator dynamics
  - ☑ α-Rank & α^α-Rank
- **Policy Improvement in Meta-games**
  - ☑ Iterated self-plays
  - ☑ Fictitious play & generalised weaken fictitious play
  - ☑ Double oracle & PSRO
  - ☑ PSRO-Nash, PSRO-Rectified-Nash, α-PSRO

Take-home Messages

- Multi-agent RL is challenging in general, a bottleneck is the infeasibility of Nash computation.
- A useful application for MARL technique is on the meta-game analysis in designing Gaming AI.
- In Gaming AI, a naive approach of self-plays will not be the general solution.
- Understanding the game structures is very important, transitive/in-transitive games have very different policy evaluation and policy improvement methods.
- Never use "reinforcement learning" to design reinforcement learning algorithms! We need to know why and why not it works.

References

[Mazumdar 2019] Policy-Gradient Algorithms Have No Guarantees of Convergence in Linear Quadratic Games. Eric Mazumdar, Lillian J. Ratliff, Michael I. Jordan, S. Shankar Sastry

[Shoham 2007] MULTIAGENT SYSTEMS: Algorithmic, Game-Theoretic, and Logical Foundations. Yoav Shoham, Kevin Leyton-Brown

[Conitzer 2002] Complexity Results about Nash Equilibria. Vincent Conitzer, Tuomas Sandholm

[Badue et. al 2019] Self-Driving Cars: A Survey Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Paixão, Filipe Mutz, Lucas Veronese, Thiago Oliveira-Santos, Alberto Ferreira De Souza

[Balduzzi 2019] Open-ended Learning in Symmetric Zero-sum Games. David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech M. Czarnecki, Julien Perolat, Max Jaderberg, Thore Graepel

[Vinyals 2019] Grandmaster level in StarCraft II using multi-agent reinforcement learning. Oriol Vinyals, Igor Babuschkin, […]David Silver

[Jaderberg 2017] Population Based Training of Neural Networks. Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, Koray Kavukcuoglu

[candogan 2010] Flows and Decompositions of Games: Harmonic and Potential Games. Ozan Candogan, Ishai Menache, Asuman Ozdaglar, Pablo A. Parrilo

[Czarnecki 2020] Real World Games Look Like Spinning Tops. Wojciech Marian Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, Max Jaderberg

[Silver 2016] Mastering the game of Go with deep neural networks and tree search. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis

[Alphastar blog] https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

[Tuyls 2018] A Generalised Method for Empirical Game Theoretic Analysis Karl Tuyls, Julien Perolat, Marc Lanctot, Joel Leibo, Thore Graepel.

[Yang 2020b] Multi-agent determinantal Q-learning. Yaodong Yang, Ying Wen, .., Jun, Wang

[Yang 2020] Practically Scaling α-Rank through Stochastic Optimisation. Yaodong Yang, Rasul Tutunov, Phu Sakulwongtana, Haitham Bou Ammar

References

[Balduzzi 2018] Re-evaluating Evaluation. David Balduzzi, Karl Tuyls, Julien Perolat, Thore Graepel

[Kianercy 2012] Dynamics of Boltzmann Q-Learning in Two-Player Two-Action Games. Ardeshir Kianercy, Aram Galstyan

[Hennes 2020] Neural Replicator Dynamics. Daniel Hennes, Dustin Morrill, Shayegan Omidshafiei, Remi Munos, Julien Perolat, Marc Lanctot, Audrunas Gruslys, Jean-Baptiste Lespiau, Paavo Parmas, Edgar Duenez-Guzman, Karl Tuyls

[Bloembergen 2015] Evolutionary Dynamics of Multi-Agent Learning: A Survey. Daan Bloembergen. Karl Tuyls. Daniel Hennes. Michael Kaisers

[Conley 1978] Isolated invariant sets and the Morse index. CC Conley

[Shayegan et al 2019] α-Rank: Multi-Agent Evaluation by Evolution. Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M. Czarnecki, Marc Lanctot, Julien Perolat & Remi Munos

[Muller 2020] A Generalized Training Approach for Multiagent Learning. Paul Muller, Shayegan Omidshafiei, Mark Rowland, Karl Tuyls, Julien Perolat, Siqi Liu, Daniel Hennes, Luke Marris, Marc Lanctot, Edward Hughes, Zhe Wang, Guy Lever, Nicolas Heess, Thore Graepel, Remi Munos

[Brown 1951] G.W. Brown, Iterative solution of games by fictitious play, in: Activity analysis of production and allocation (T.C. Koopmans, Ed.), pp. 374-376, Wiley: New York, 1951.

[Leslie 2006] Generalised weakenened fictitious pla. David S. Leslie 1. E. J.Collins ∗

[McMahan 2003] Planning in the Presence of Cost Functions Controlled by an Adversary H. Brendan McMahan Geoffrey J Gordon Avrim Blum

[Lanctot 2017] A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, Thore Graepel

[DeepMind MAS tutorial] https://bit.ly/2Um0T9d

[Heinrich 2015] Fictitious Self-Play in Extensive-Form Games. Johannes Heirich, Marc Lanctot, David Silver

[Openspiel] OpenSpiel: A Framework for Reinforcement Learning in Games. https://github.com/deepmind/open_spiel

[Jiang 2011] Statistical ranking and combinatorial Hodge theory. Xiaoye Jiang, Lek-Heng Lim, Yuan Yao & Yinyu Ye