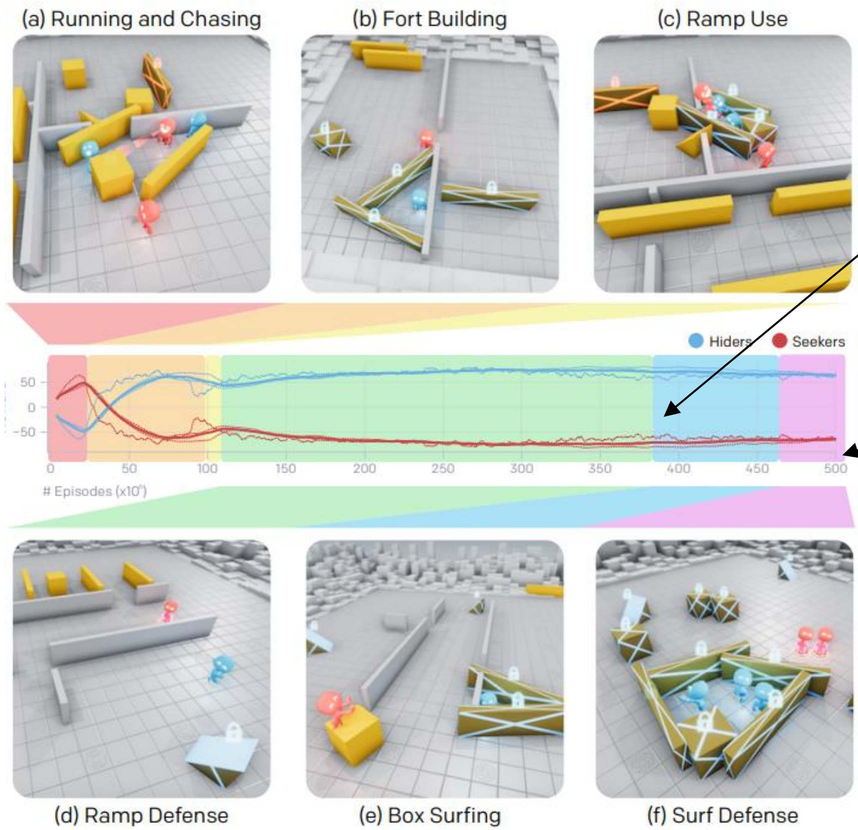
Deep learning in games: Algorithms based on single-agent RL

Brian Zhang

What if we just run single-agent RL, independently? ("self-play")

- Not guaranteed to converge to equilibrium, even in averages
- In practice: sometimes works, especially with very large amounts of compute



after ≈400M episodes: trained agents started exploiting a bug in the game's code!

total training:
≈ 600M episodes
≈ 32 billion frames
≈ 16 years of experience
(assuming 60 fps)

Today: More game-theoretically-motivated methods that use single-agent RL

Recap: Fictitious Play

$$x_i^{t+1} = \arg\max_{x_i} \frac{1}{t} \sum_{\tau=1}^{t} u_i(x_i, x_{-i}^{\tau})$$

*Best respond to the opponent's **average** strategy so far*

Converges to Nash in 2p0s games, but convergence rate is…

- **…slow** with adversarial tiebreaking [Daskalakis & Pan 2014]
- **…an open problem** with "reasonable" tiebreaking rules

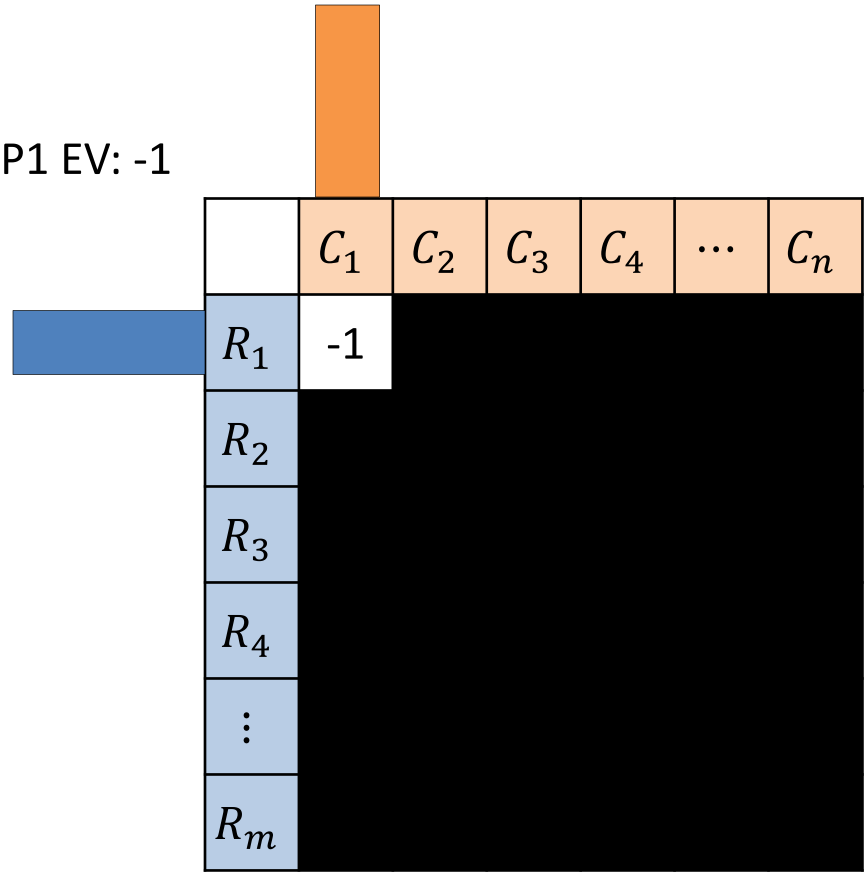
Only requires a best-response oracle!

⇒ We can use **single-agent RL methods** to run an approximate version of FP

⇒ "Neural fictitious self-play" (NFSP)

Double Oracle

P1 EV: -1



Double Oracle

Nash gap: 3

P1 EV: -1

BR →

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1	1	-1	1	$\cdots$	1
R_2	2					
R_3	1					
R_4	-1					
$\vdots$	$\vdots$					
R_m	-1					

Double Oracle

P1 EV: 2

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1					
R_2	2					
R_3						
R_4						
$\vdots$						
R_m						

Double Oracle

BR

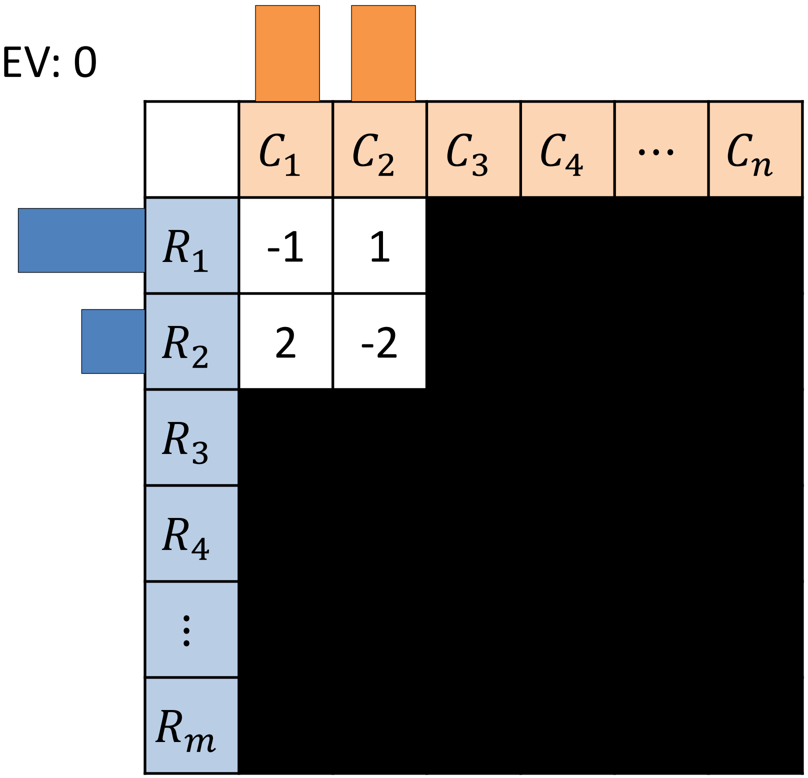
Nash gap: **4**

P1 EV: 2

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1					
R_2	2	-2	-1	1	$\cdots$	1
R_3	1					
R_4	-1					
$\vdots$	$\vdots$					
R_m	-1					

Double Oracle

P1 EV: 0



Double Oracle

Nash gap: 2

P1 EV: 0

BR ↓ (C_3)

BR → (R_3)

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1	1	-1	1	$\cdots$	1
R_2	2	-2	-1	1	$\cdots$	1
R_3	1	1				
R_4	-1	-1				
$\vdots$	$\vdots$	$\vdots$				
R_m	-1	-1				

Double Oracle

P1 EV: 0

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1	1	-1			
R_2	2	-2	-1			
R_3	1	1	0			
R_4						
$\vdots$						
R_m						

Double Oracle

Nash gap: **0 (done!)**

P1 EV: 0

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1	1	-1			
R_2	2	-2	-1			
R_3	1	1	0	1	$\cdots$	1
R_4			-1			
$\vdots$			$\vdots$			
R_m			-1			

Not explored, but that's OK!

Normal form: DO always finds an *exact equilibrium* in linearly many steps (obvious)

Extensive form:

- DO always converges in $\leq 2^N$ (N = number of nodes) steps (obvious—this bounds the number of total strategies)
- There exist 2p0s EFGs where, with *adversarial tiebreaking* (in both "meta-equilibrium" and best responses), DO takes $2^{\Omega(N)}$ steps to converge [Zhang & Sandholm IJCAI'24].

Like FP, DO only needs a best-response oracle!

Policy Space Response Oracles (PSRO)

Generalizes FP and DO.

n-player game; X_i = player i's pure strategy set

Meta-solver: takes finite subsets $\tilde{X}_i^t \subseteq X_i$ for each player i; outputs a *meta-strategy* π^t for the game restricted to the $\tilde{X}_i^t$s

FP: uniform over $\tilde{X}_i^t$
DO: Nash equilibrium of restricted game

Algorithm: Keep restricted strategy sets $\tilde{X}_1^t, \tilde{X}_2^t$, initialized arbitrarily
for $t = 1, \dots, T$:
  $\pi^t \leftarrow$ meta-strategy for game restricted to $\left(\tilde{X}_1^t, \tilde{X}_2^t\right)$
  for each player i: get best response $x_i^t \in X_i$ to π_{-i}^t, and set $\tilde{X}_i^{t+1} \leftarrow \tilde{X}_i^t \cup \left\{x_i^t\right\}$
output π^T

Today: approximate best responses with RL

The Rest of This Lecture: Fancy Versions of PSRO

- OpenAI Five and AlphaStar—large-scale practical achievements in zero-sum games
- More modern variants of PSRO

The Rest of This Lecture: Fancy Versions of PSRO

- **OpenAI Five and AlphaStar—large-scale practical achievements in zero-sum games**
- More modern variants of PSRO

Berner et al. (arXiv 2019)

OpenAI Five Plays Dota 2

Players act as a team, see the same things, and can communicate
⇒ it's really a two-*player* zero-sum game!

- Popular "5v5" zero-sum real-time strategy (RTS) game
- Continuous-time, continuous-action

Timeline:

- **2017:** OpenAI introduces initial Dota 2 AI; beat a professional player in 1v1
- **2018:** OpenAI Five plays full Dota 2 (5v5) against top human teams; *loses*
- **April 2019:** OpenAI Five plays and defeats the world champion team OG by 2-0 in a best-of-three match
- **June 2019:** OpenAI Five released on public server… and found to be exploitable!

Dota 2 Training

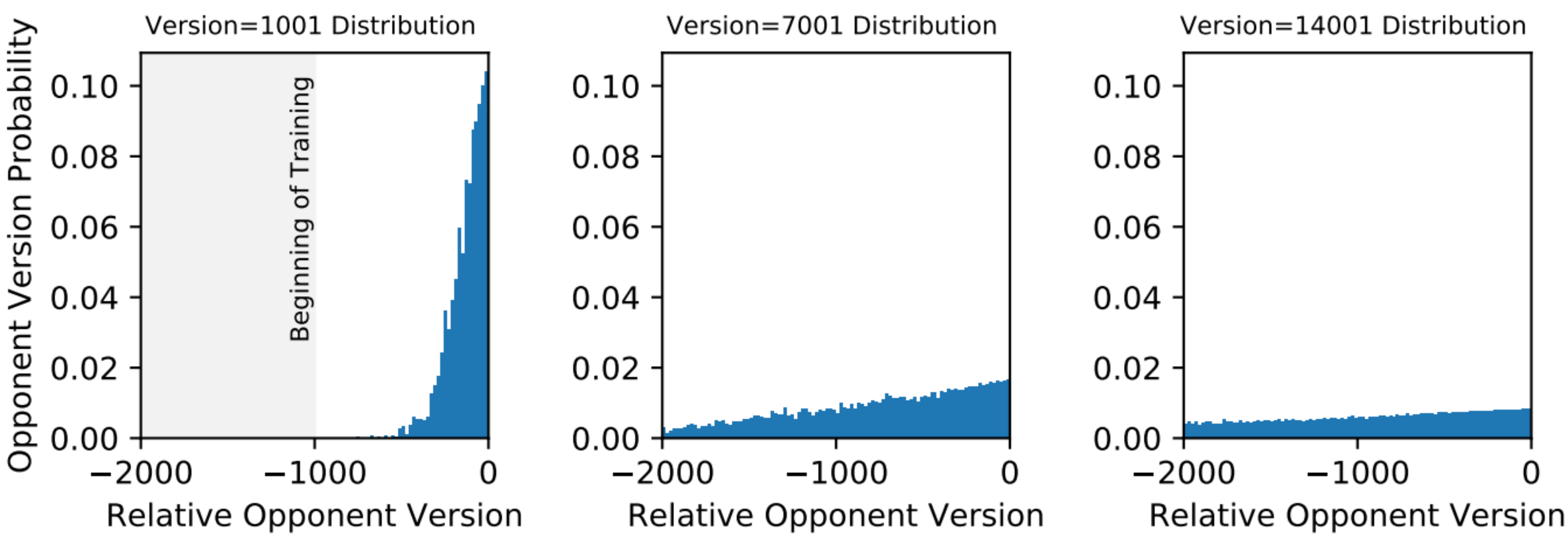
Agent trains against a **mixture**: 80% current strategy, 20% against past strategies

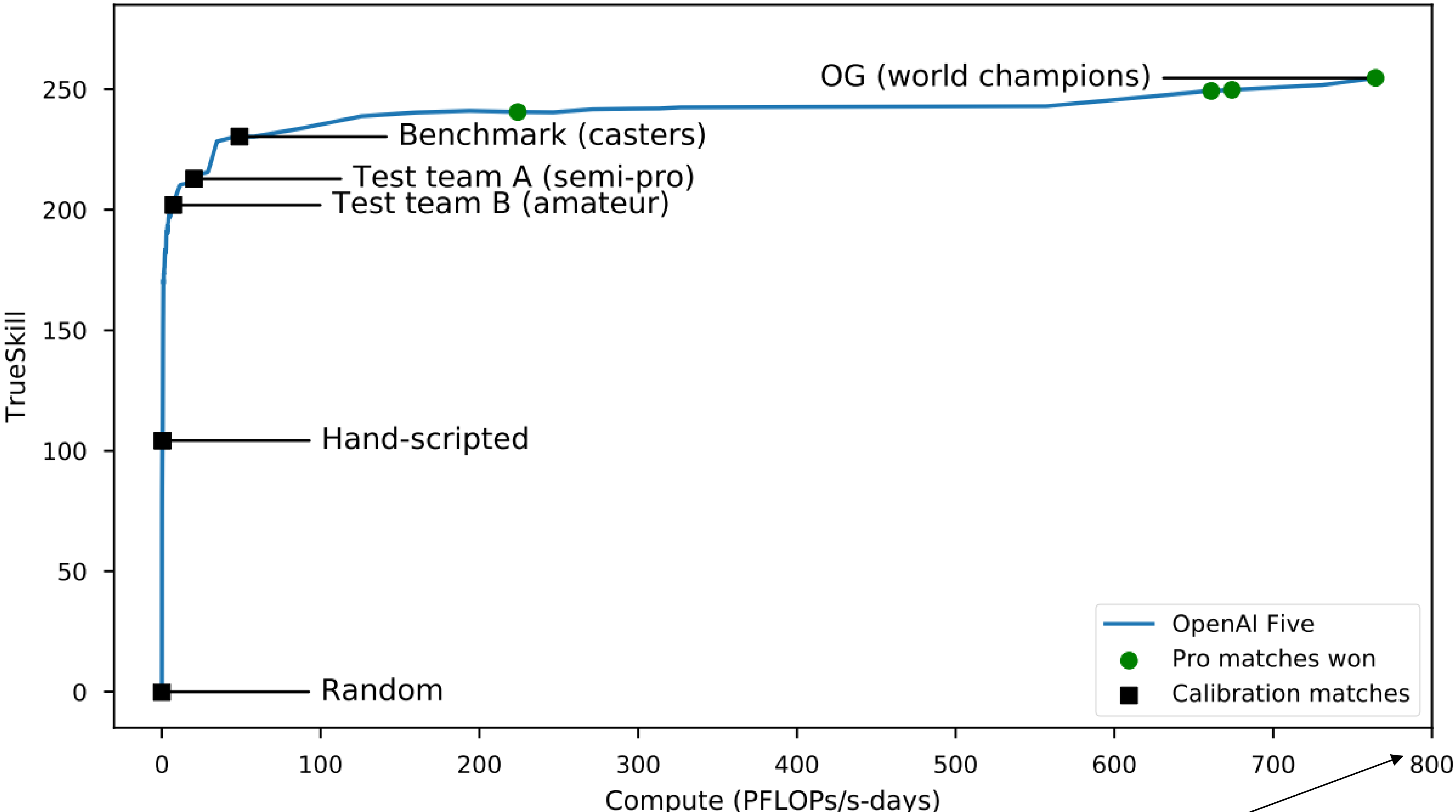
Past strategy k weighted by $p_k \propto e^{q_k}$, where q_k depends on how well the current strategy is doing against past strategy i:

$$q_k \leftarrow q_k - \frac{1}{100tp_k}$$

every time i loses a game to the current agent, where t is the current timestep.

⇒ "PSRO-like" training process





total training: $800\ \frac{\text{PFLOP}}{\text{s}} \cdot \text{days} \approx 7 \times 10^7\ \text{PFLOP} = 70\ \text{ZFLOP}$

57600 parallel games at $\frac{1}{2}$ speed $\times$ 180 days $\approx$ 14000 years of experience

Meanwhile…

DeepMind’s AlphaStar Plays StarCraft II

- Popular two-player zero-sum real-time strategy (RTS) game
- Continuous-time, continuous-action

Timeline:

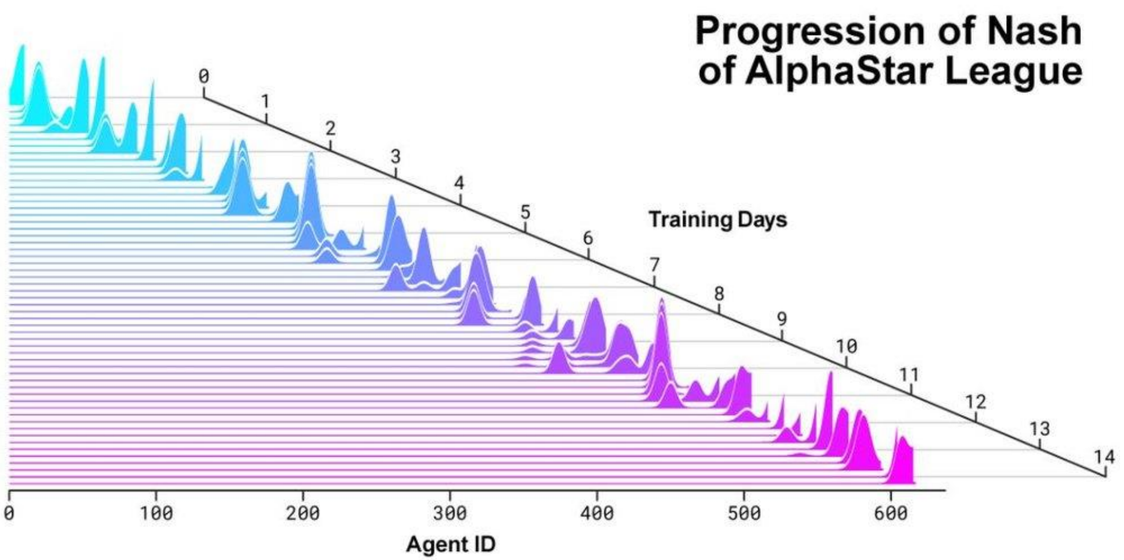
- **2016:** Partnership between DeepMind and Blizzard announced
- **2017:** Introduction of the StarCraft II Learning Environment (SC2LE)
- **Early-Mid 2019:** AlphaStar competes anonymously on public servers, achieving grandmaster-level performance
- **Late 2019:** AlphaStar paper published in Nature

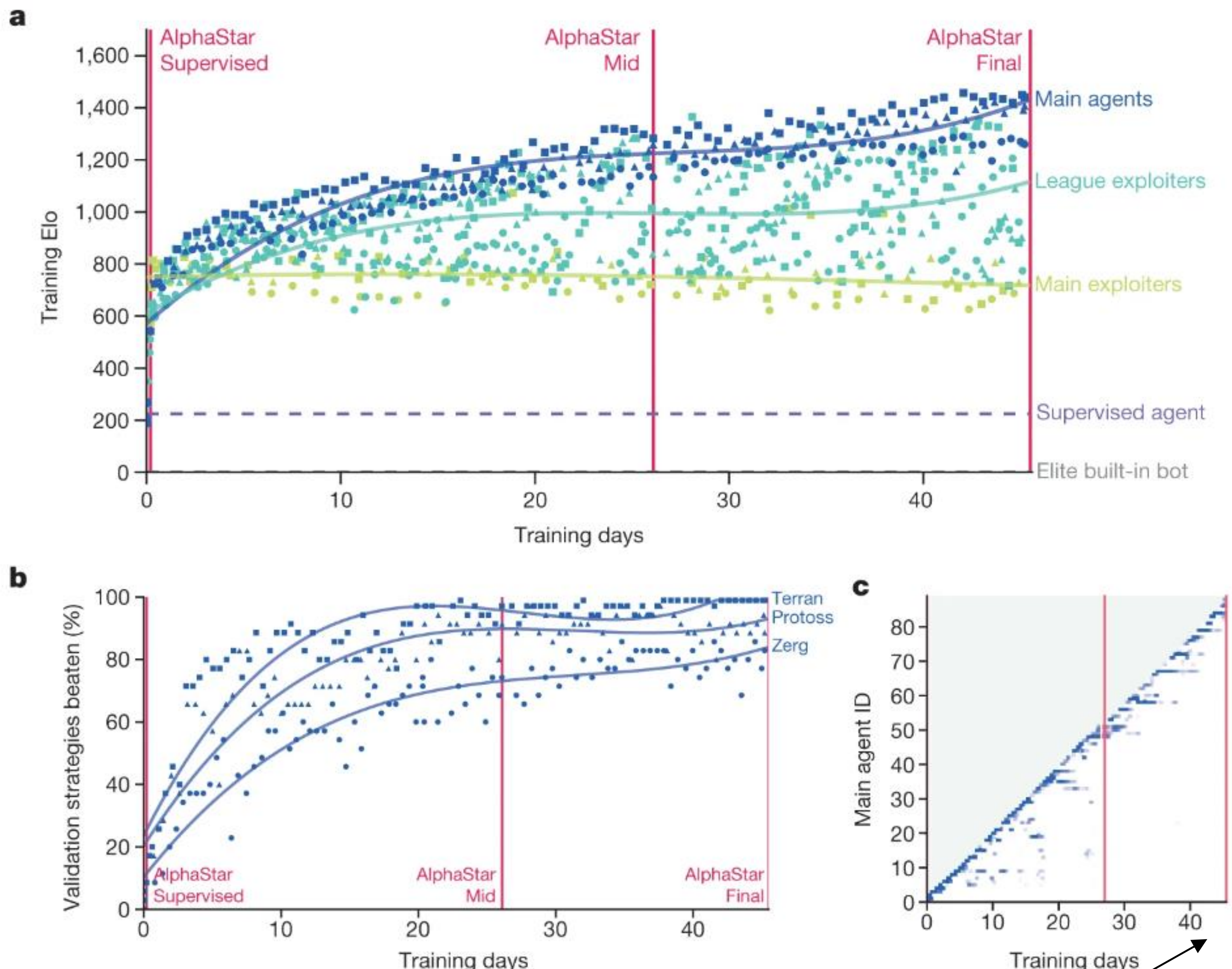
League Training (roughly)

Maintain a **league** of past agents (think: partial strategy set $\tilde{X}_i^t$)
League contains three types of agents: **main, main exploiter, league exploiter**

Prioritized fictitious self-play (PFSP): weight league player y by some function $f\big(w(y)\big)$ depending on $w(y)$, the winrate against y

Main agents: Trained by PFSP against the league
Main exploiters: Trained against **current** main agents
League exploiters: Trained by PFSP against the league (but not targeted by main exploiters)





total training time:
44 days × 16000 parallel games
≈1900 years of experience

The Rest of This Lecture: Fancy Versions of PSRO

- OpenAI Five and AlphaStar—large-scale practical achievements in zero-sum games

- **More modern variants of double oracle/PSRO**

Pros and Cons of Double Oracle/PSRO

Pros:

- Practically sometimes faster than FP or CFR, esp. with deep RL
- Easy to use: deep RL is "black-boxed" away
- Demonstrated excellent performance in e.g. Starcraft/Dota II

Cons:

- Requires re-computing best responses on every iteration ⇒ expensive
- Exponential-time worst-case performance
- Non-monotone exploitability
- Strategies added "greedily" (to optimize best-response value, not to decrease exploitability of the meta-Nash)

Parallelizing PSRO

Naïve: with n parallel workers, train n (approximate) best responses on each iteration

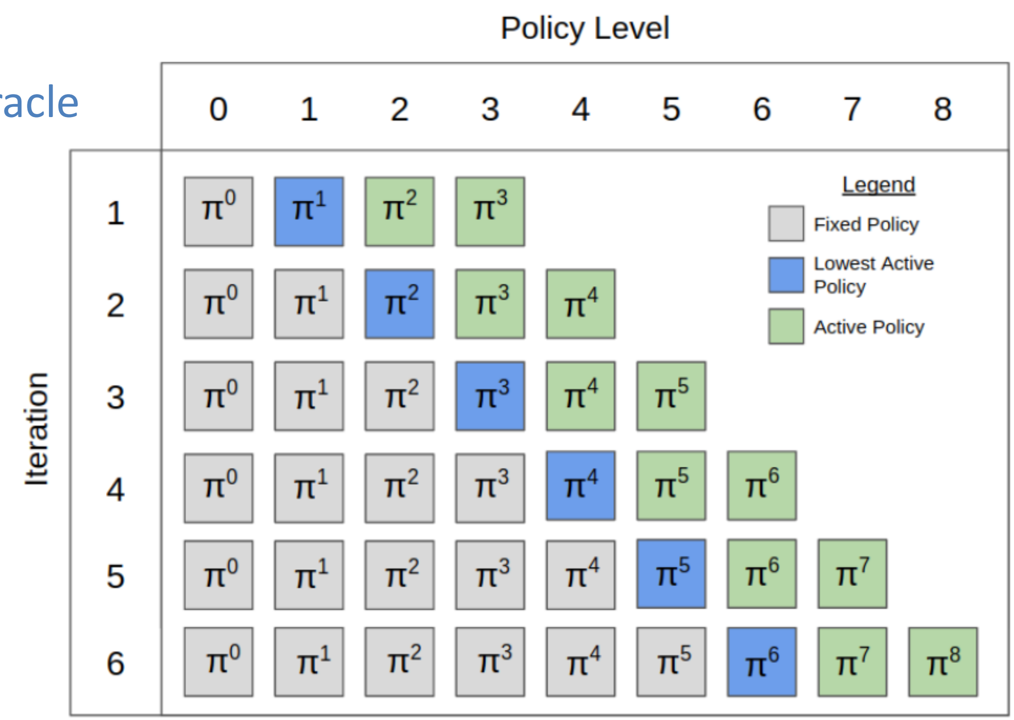
Can we do better?

Pipeline PSRO (P2SRO)

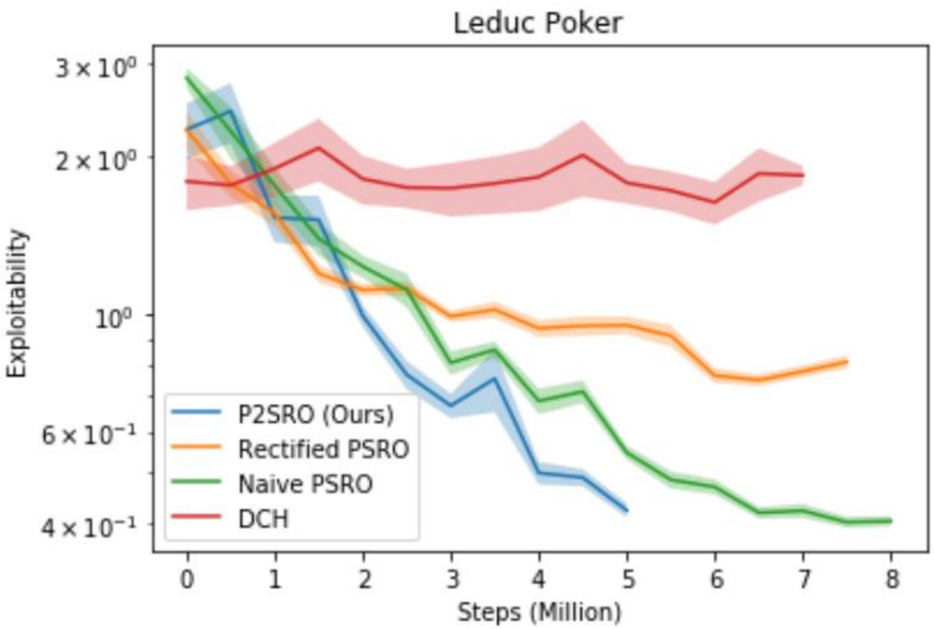
π_i^t := player i's BR at time t
Γ^t := subgame where each player i is restricted to $\{\pi_i^0, \dots, \pi_i^t\}$
on iteration t:
  strategies $\pi_i^0, \dots, \pi_i^t$ are fixed
  repeat until π_i^{t+1} plateaus:
    for $s \in \{t+1, t+2, \dots, t+k\}$:
      Compute meta-NE $\sigma^s \in \Delta([s])$ for subgame Γ^s
      Train π_i^{s+1} (for some number of steps) to best respond to σ_{-i}^s

For $k = 1$ this is just regular double oracle

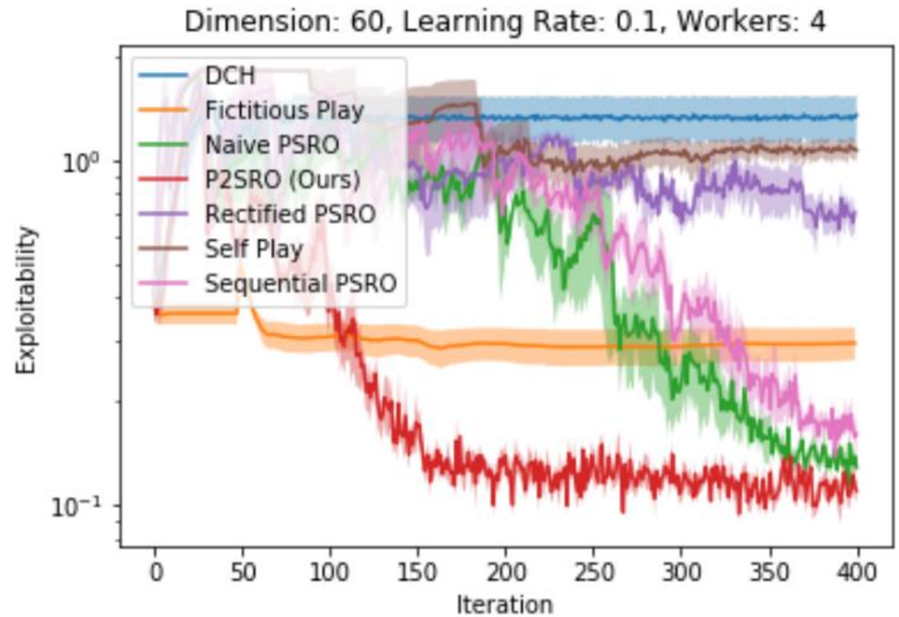
P2SRO to “pre-start” π_i^s long before (k iterations before) it is needed



Pipeline PSRO Experiments

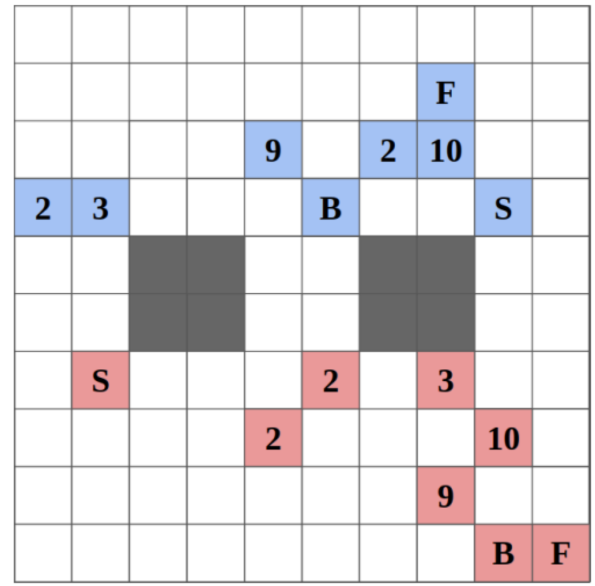


(a) Leduc poker



(b) Random Symmetric Normal Form Games

Pipeline PSRO Experiments: Barrage Stratego



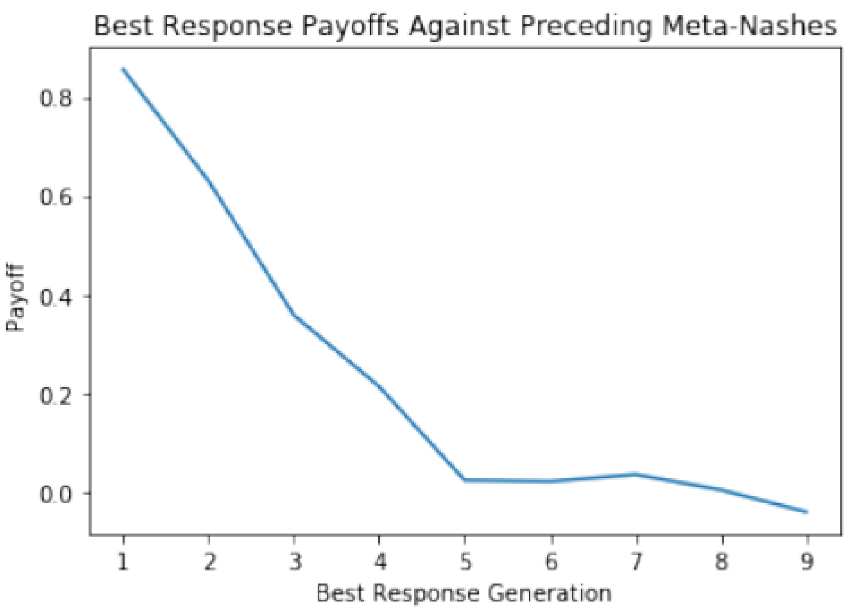


Figure 3: Barrage Best Response Payoffs Over Time

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

Table 1: Barrage P2SRO Results vs. Existing Bots

Anytime PSRO

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1					
R_2						
R_3						
R_4						
$\vdots$						
R_m						

Anytime PSRO

Nash gap: 3

BR → R_2

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1	1	-1	1	$\cdots$	1
R_2	2					
R_3	1					
R_4	-1					
$\vdots$	$\vdots$					
R_m	-1					

Anytime PSRO

Nash gap: 3

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1					
R_2	2					
R_3						
R_4						
$\vdots$						
R_m						

Anytime PSRO

Nash gap: 3

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1	1	-1	1	$\cdots$	1
R_2	2	-2	-1	1	$\cdots$	1
R_3						
R_4						
$\vdots$						
R_m						

Idea: Solve the *one-sided restricted game* to compute meta-strategies

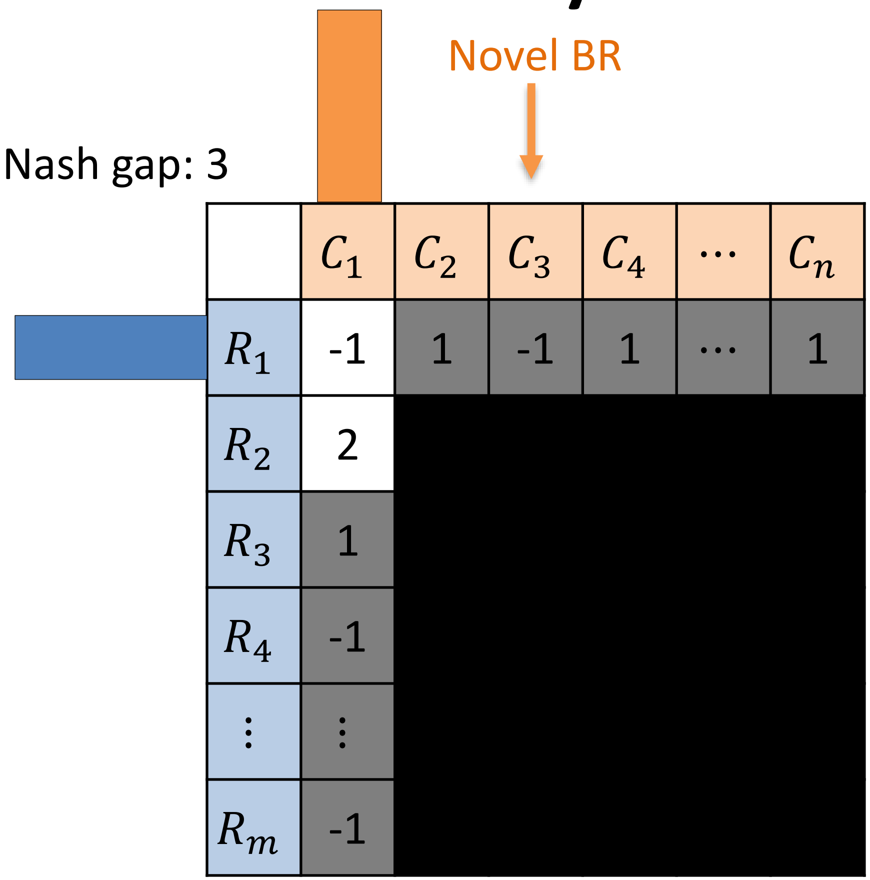
Something's wrong…

Requirement: Always find a *novel* best response if possible

Diversity is good! e.g.:

$$\pi_i^{t+1} = \arg\max_{\pi_i} \left\{ u(\pi_i, \sigma_{-i}^t) + \lambda \min_{\pi_i^k \in \mathcal{H}(\Pi_i^t)} \mathrm{dist}(\pi_i, \pi_i^k) \right\}$$

Anytime PSRO



Idea: Solve the *one-sided restricted game* to compute meta-strategies

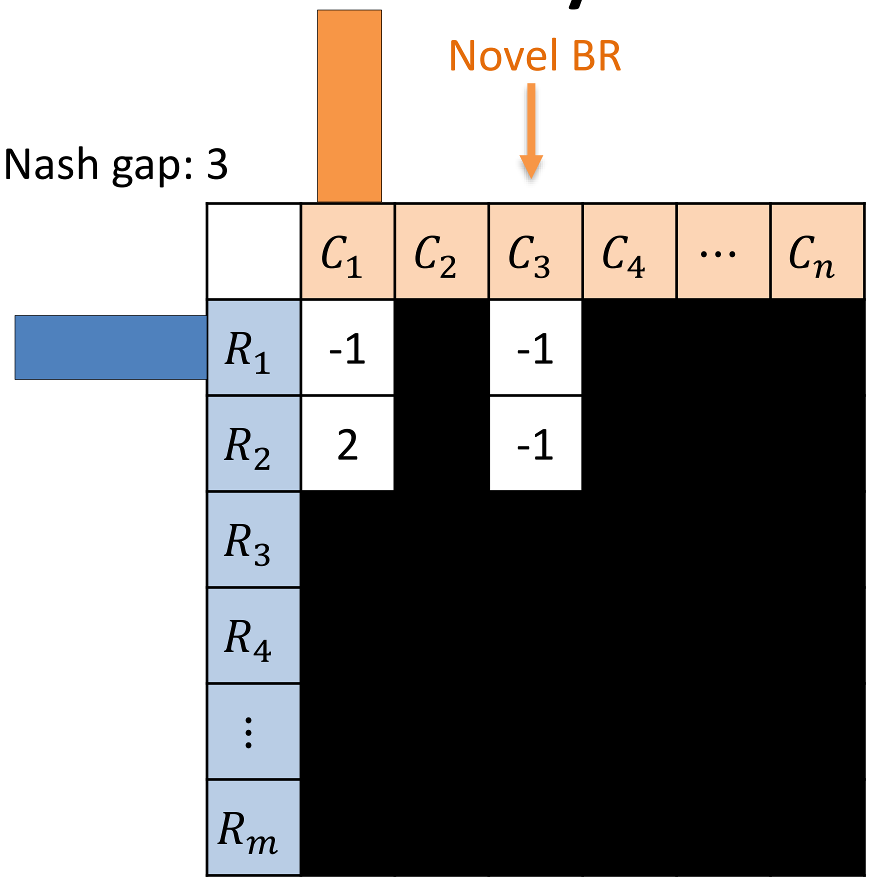
Something's wrong…

Requirement: Always find a *novel* best response if possible

Diversity is good! e.g.:

$$\pi_i^{t+1} = \arg\max_{\pi_i} \left\{ u(\pi_i, \sigma_{-i}^t) + \lambda \min_{\pi_i^k \in \mathcal{H}(\Pi_i^t)} \mathrm{dist}(\pi_i, \pi_i^k) \right\}$$

Anytime PSRO



Idea: Solve the *one-sided restricted game* to compute meta-strategies

Something's wrong…

Requirement: Always find a *novel* best response if possible

Diversity is good! e.g.:

$$\pi_i^{t+1} = \arg\max_{\pi_i} \left\{ u(\pi_i, \sigma_{-i}^t) + \lambda \min_{\pi_i^k \in \mathcal{H}(\Pi_i^t)} \operatorname{dist}(\pi_i, \pi_i^k) \right\}$$

Anytime PSRO

Nash gap: 3

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1		-1			
R_2	2		-1			
R_3	1		0			
R_4	-1		-1			
$\vdots$	$\vdots$		$\vdots$			
R_m	-1		-1			

Idea: Solve the *one-sided restricted game* to compute meta-strategies

Something's wrong…

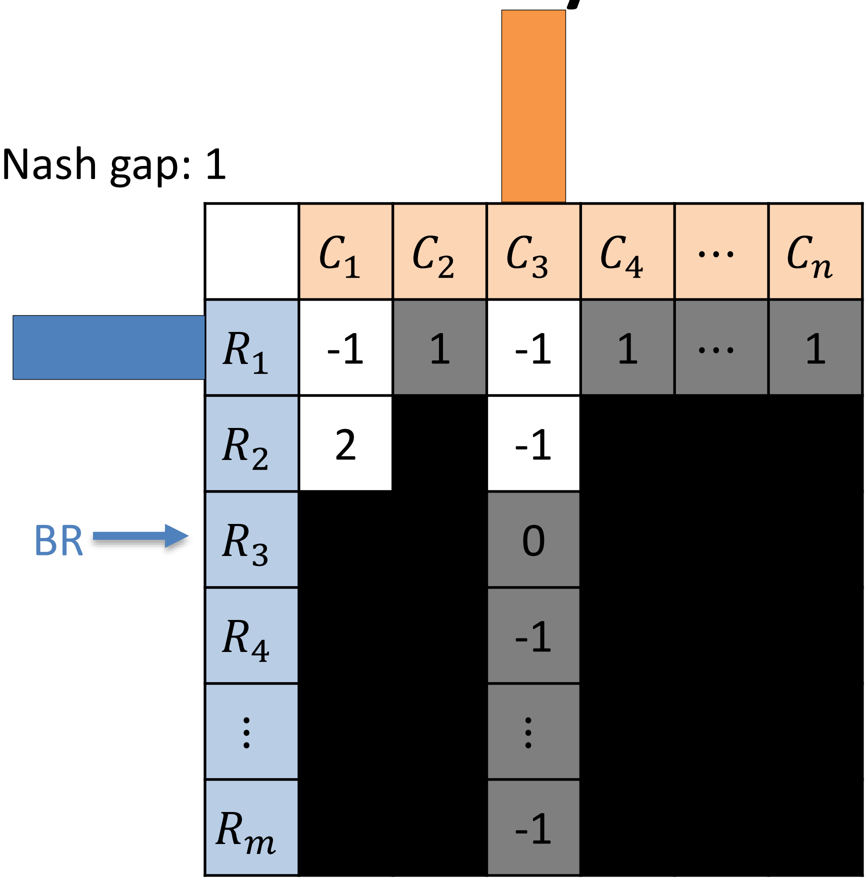
Requirement: Always find a *novel* best response if possible

Diversity is good! e.g.:

$$\pi_i^{t+1} = \arg\max_{\pi_i} \left\{ u(\pi_i, \sigma_{-i}^{t}) + \lambda \min_{\pi_i^k \in \mathcal{H}(\Pi_i^t)} \mathrm{dist}(\pi_i, \pi_i^k) \right\}$$

Anytime PSRO

Nash gap: 1



Idea: Solve the *one-sided restricted game* to compute meta-strategies

Something's wrong…

Requirement: Always find a *novel* best response if possible

Diversity is good! e.g.:

$$\pi_i^{t+1} = \arg\max_{\pi_i} \left\{ u(\pi_i, \sigma_{-i}^t) + \lambda \min_{\pi_i^k \in \mathcal{H}(\Pi_i^t)} \mathrm{dist}(\pi_i, \pi_i^k) \right\}$$

Anytime PSRO

Nash gap: 1

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1		-1			
R_2	2		-1			
R_3	1		0			
R_4						
$\vdots$						
R_m						

Idea: Solve the *one-sided restricted game* to compute meta-strategies

Something's wrong…

Requirement: Always find a *novel* best response if possible

Diversity is good! e.g.:

$$\pi_i^{t+1} = \arg\max_{\pi_i} \left\{ u(\pi_i, \sigma_{-i}^t) + \lambda \min_{\pi_i^k \in \mathcal{H}(\Pi_i^t)} \mathrm{dist}(\pi_i, \pi_i^k) \right\}$$

Anytime PSRO

Nash gap: 1

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1	1	-1	1	$\cdots$	1
R_2	2	-2	-1	1	$\cdots$	1
R_3	1	1	0	1	$\cdots$	1
R_4						
$\vdots$						
R_m						

Idea: Solve the *one-sided restricted game* to compute meta-strategies

Something's wrong…

Requirement: Always find a *novel* best response if possible

Diversity is good! e.g.:

$$\pi_i^{t+1} = \arg\max_{\pi_i} \left\{ u(\pi_i, \sigma_{-i}^t) + \lambda \min_{\pi_i^k \in \mathcal{H}(\Pi_i^t)} \mathrm{dist}(\pi_i, \pi_i^k) \right\}$$

Anytime PSRO

Nash gap: 0

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1	-1		-1			
R_2	2		-1			
R_3	1		0	1	$\cdots$	1
R_4			-1			
$\vdots$			$\vdots$			
R_m			-1			

Exploitability is monotonically nonincreasing ☺

Every iteration requires us to solve a full game ☹

…in which P1 has not too many strategies. Can we solve it efficiently?

How do we solve games where one side has a small number of strategies?

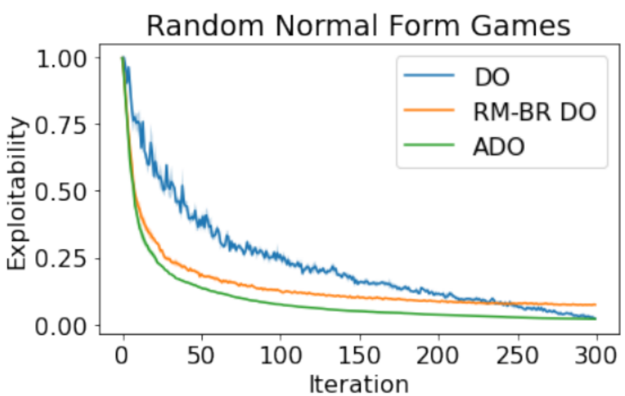
Recall (HW1): If P1 runs a regret minimizer and P2 best-responds on every step, then

$$\text{Nash gap} \leq \text{P1's regret} / T$$

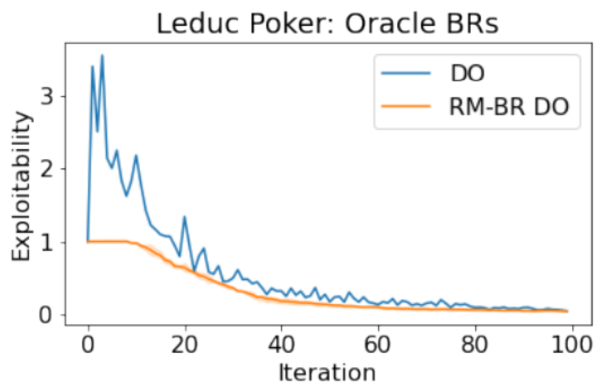
$\Rightarrow$ extremely efficient equilibrium computation when P1's strategy set is small!

Anytime PSRO = one-sided PSRO
+ this idea ("regret minimization with best responses"/"RM-BR")
+ RL best-response oracle for P2

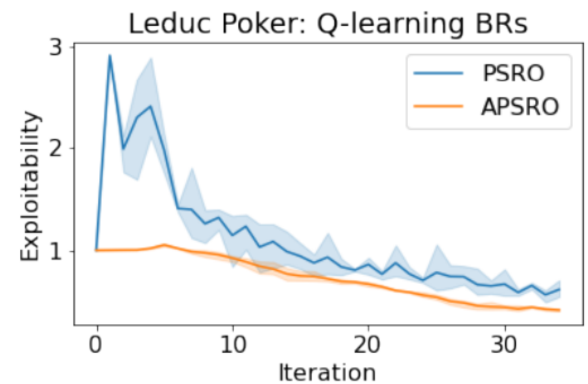
Anytime PSRO Experiments



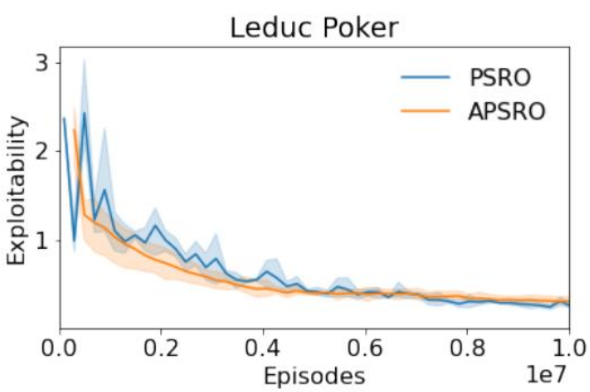
(a) Random Normal Form Games



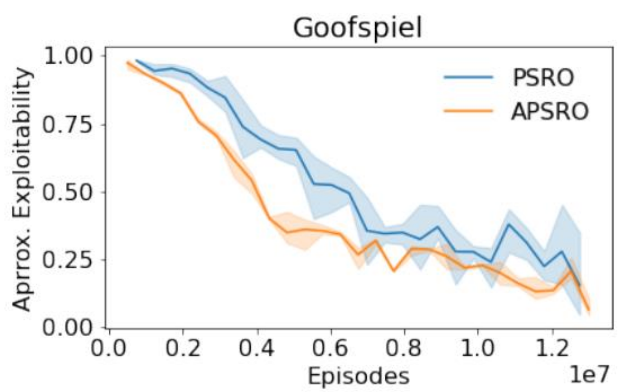
(b) Leduc with Oracle Best Responses



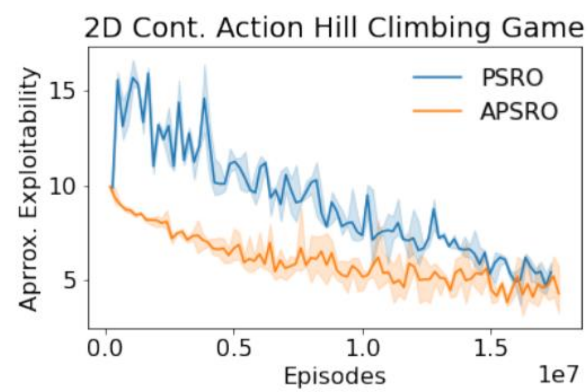
(c) Leduc with Q-Learning Best Responses



(a) Leduc with DDQN BRs



(b) Goofspiel with DDQN BRs



(c) Continuous-Action Hill-Climbing Game

RM-BR

$y^t = \mathrm{BR}(x^t)$

strategy x^t
selected by RM

	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1						
R_2						
R_3						

RM-BR?

$y^t = \mathrm{BR}(x^t)$

strategy x^t selected by RM

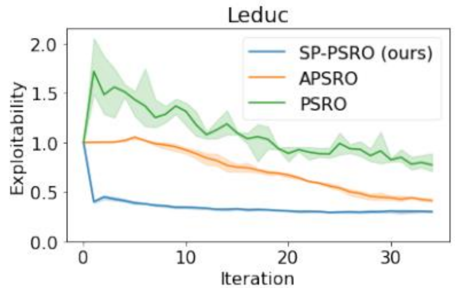
	C_1	C_2	C_3	C_4	$\cdots$	C_n
R_1						
R_2						
R_3						
$v^t = \mathrm{BR}(y^t)$						

After some time, add $\bar{v}^t$ to P1's strategy set and y^t to P2's strategy set

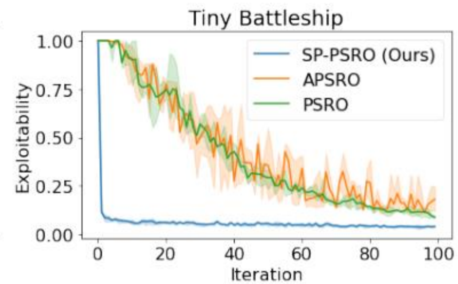
"Self-play PSRO"

Intuition: self-play "stabilized" by having strategies R_1, R_2, R_3 available to the row player ⇒ better PSRO performance in practice?

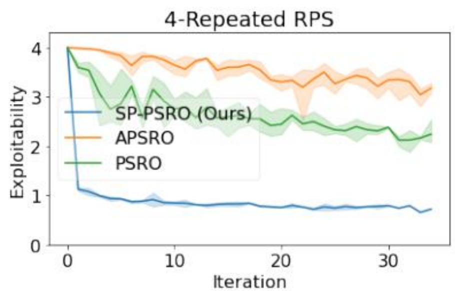
Self-play PSRO experiments



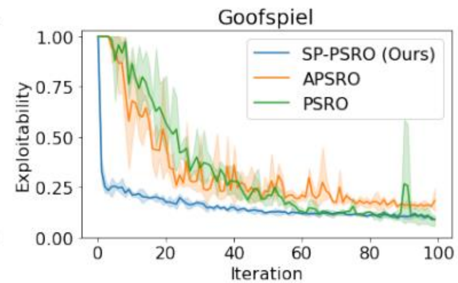
(a) Leduc Poker



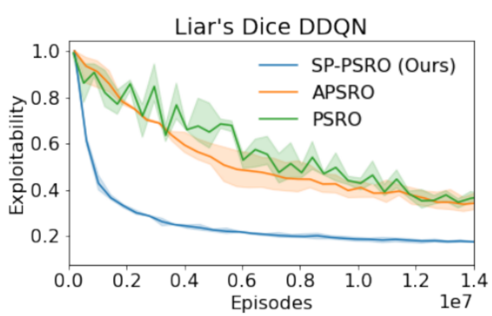
(b) Battleship



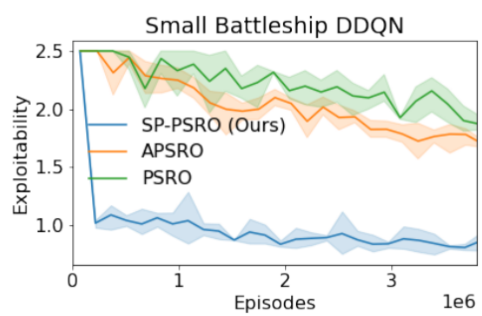
(c) Repeated RPS



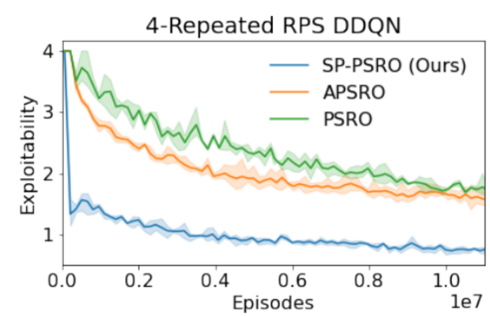
(d) Goofspiel



(a) DRL Liars Dice



(b) DRL Battleship



(c) DRL Repeated RPS

References

Constantinos Daskalakis, Qinxuan Pan (FOCS 2014) "A Counter-Example to Karlin's Strong Conjecture for Fictitious Play“

Brian Hu Zhang, Tuomas Sandholm (IJCAI 2024) “Exponential Lower Bounds on the Double Oracle Algorithm in Zero-Sum Games”

Double oracle: H Brendan McMahan, Geoffrey J Gordon, Avrim Blum (ICML 2003) “Planning in the Presence of Cost Functions Controlled by an Adversary”

PSRO: Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, Thore Graepel (NeurIPS 2017) “A unified game-theoretic approach to multiagent reinforcement learning”

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, Igor Mordatch (ICLR 2020) “Emergent Tool Use From Multi-Agent Autocurricula”

Oriol Vinyals et al. (Nature 2019) “Grandmaster level in StarCraft II using multi-agent reinforcement learning”

Christopher Berner et al. (arXiv 2019) “Dota 2 with Large Scale Deep Reinforcement Learning”

Stephen McAleer, John Lanier, Roy Fox, Pierre Baldi (NeurIPS 2020) “Pipeline PSRO: A Scalable Approach for Finding Approximate Nash Equilibria in Large Games”

Anytime and self-play PSRO: Stephen McAleer, JB Lanier, Kevin Wang, Pierre Baldi, Roy Fox, Tuomas Sandholm (ICLR 2024) “Toward Optimal Policy Population Growth in Two-Player Zero-Sum Games”

Jian Yao, Weiming Liu, Haobo Fu, Yaodong Yang, Stephen McAleer, Qiang Fu, Wei Yang (NeurIPS 2023) “Policy Space Diversity for Non-Transitive Games”