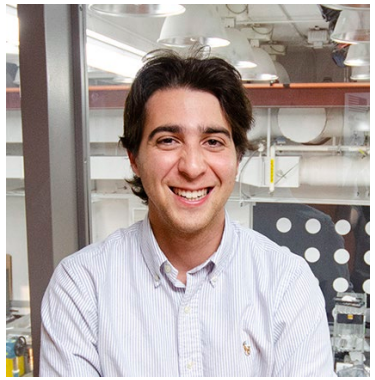
Reevaluating Policy Gradient Methods for Imperfect Information Games

IMSI Workshop
Eugene Vinitsky – NYU – EMERGE Lab

1. A scalable way to evaluate MARL algorithms in 2p0s games
2. The policy gradient hypothesis

Reevaluating Policy Gradient Methods for IIGs – ICLR 2026



Max Rudolph*

Nathan Lichtlé*

Sobhan Mohammadpour*

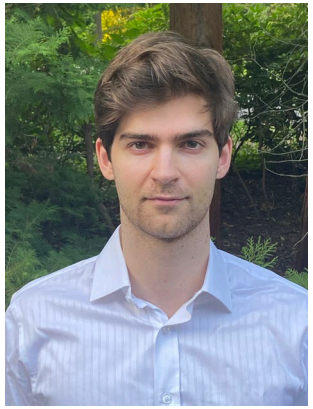
Alexandre Bayen

Zico Kolter

Amy Zhang

Gabriele Farina

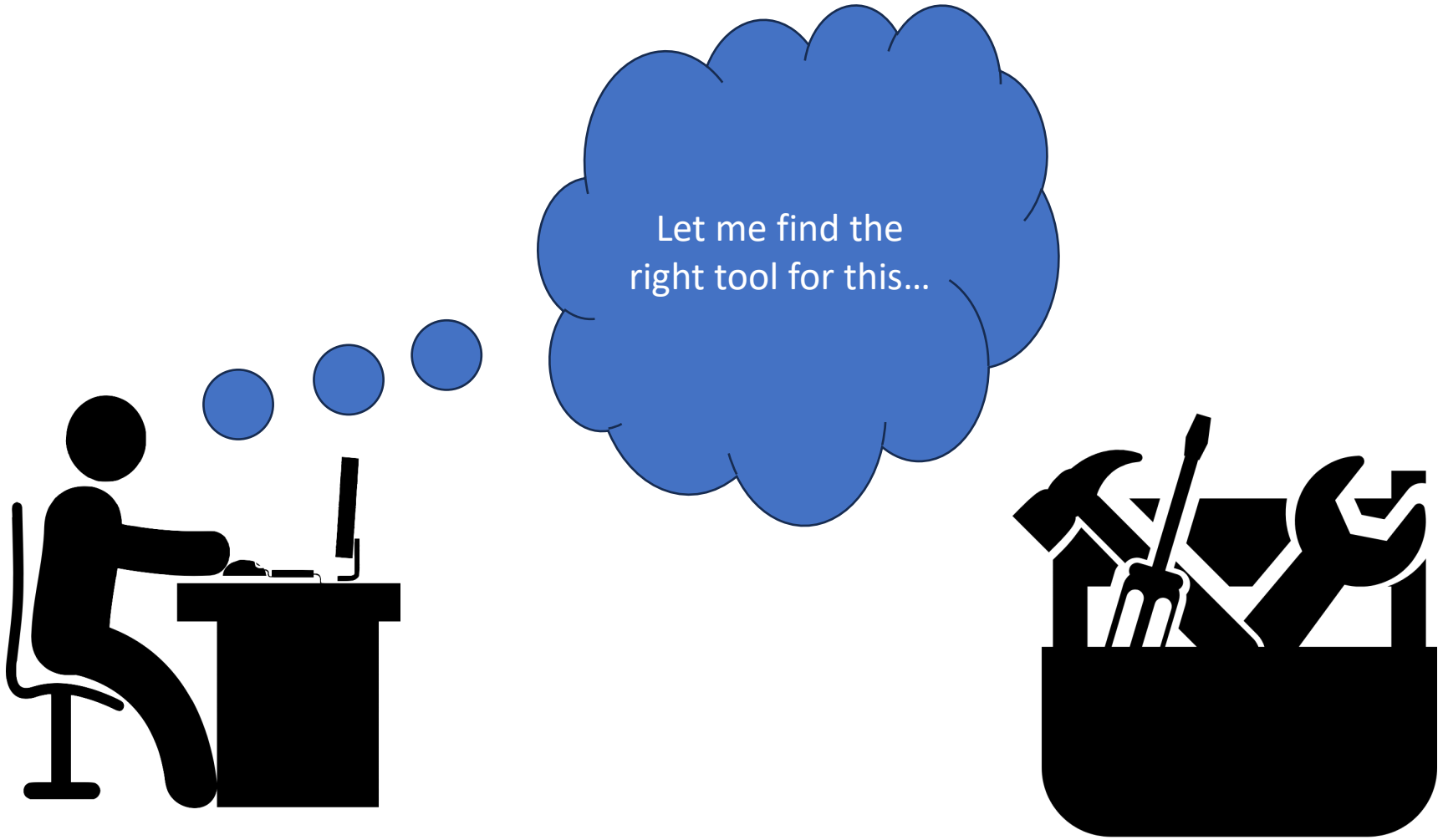
Some guy



Samuel Sokota

Toward reliable engineering tools for imperfect information games

Toward reliable engineering tools for imperfect information games



A proliferation of algorithms!

PDO

Fusion PSRO

XDO

Policy-Space Response Oracles (PSRO)

EPSRO

Mixed-Opponents

ARMAC

Prioritized Fictitious Self-Play

DREAM

FPEM

ADO

ODO

Self-play PSRO

Anytime PSRO

IPPO

ESCHER

PSD-PSRO

AdaptFSP

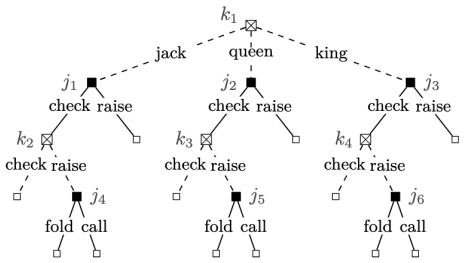
R-NaD

Pipeline PSRO

Fictitious Play

Neural Fictitious Self-Play

How did we get here? Benchmark issues.



A missing middle

- Every state visited many times
- Exploitability efficient
- No generalization

- Almost every state never visited
- Exploitability intractable
- Human experimentation

Frictionless Reproducibility*

- The broad success of ML / data science stems from
  - Code sharing
  - Competitive challenges
  - Data sharing

*”Data Science at the Singularity”, David Donoho

Frictionless Reproducibility*

- The broad success of ML / data science stems from
  - Code sharing 
  - Competitive challenges 
  - Data sharing 

*”Data Science at the Singularity”, David Donoho

Frictionless Reproducibility*

- The broad success of ML / data science stems from
  - Code sharing 
  - Competitive challenges  ← **Today's topic**
  - Data sharing 

*"Data Science at the Singularity", David Donoho

1. We have superhuman performance on **so many games**
2. Even experts are unaware of which algorithm to pick

1. We have superhuman performance on **so many games**
2. Even experts are unaware of which algorithm to pick

So lets find a way out!

So what does the landscape of scalable benchmarking look like?

3 imperfect solutions

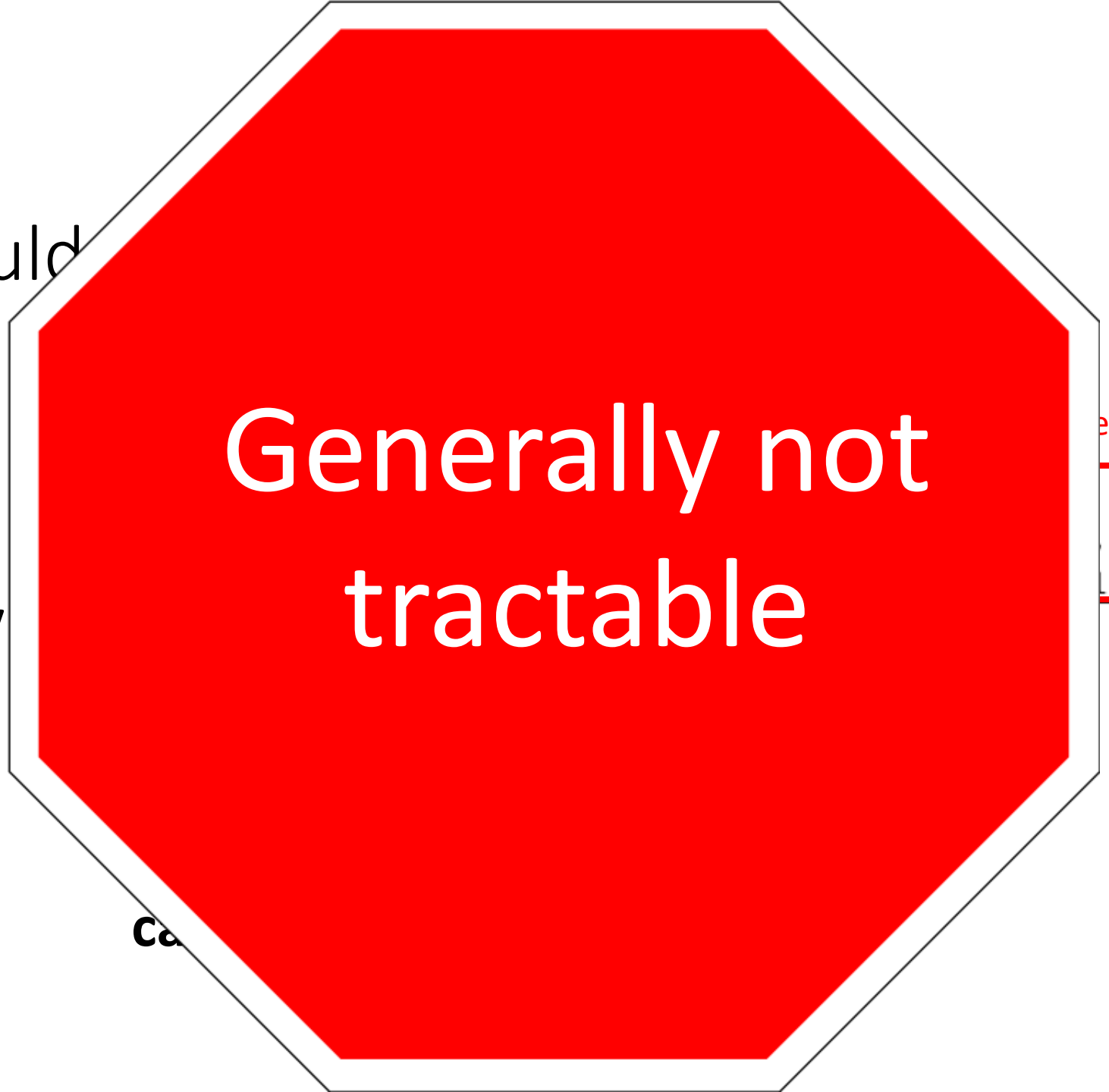
Small-game exploitability

Approximate exploitability

Head-to-head

How would

Compute exploitability



e RL problem

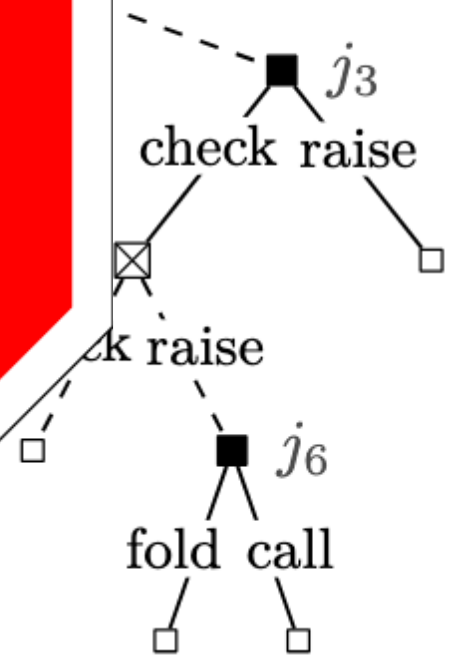
$\mathcal{J}(\pi_0, \pi_1')$

Ca

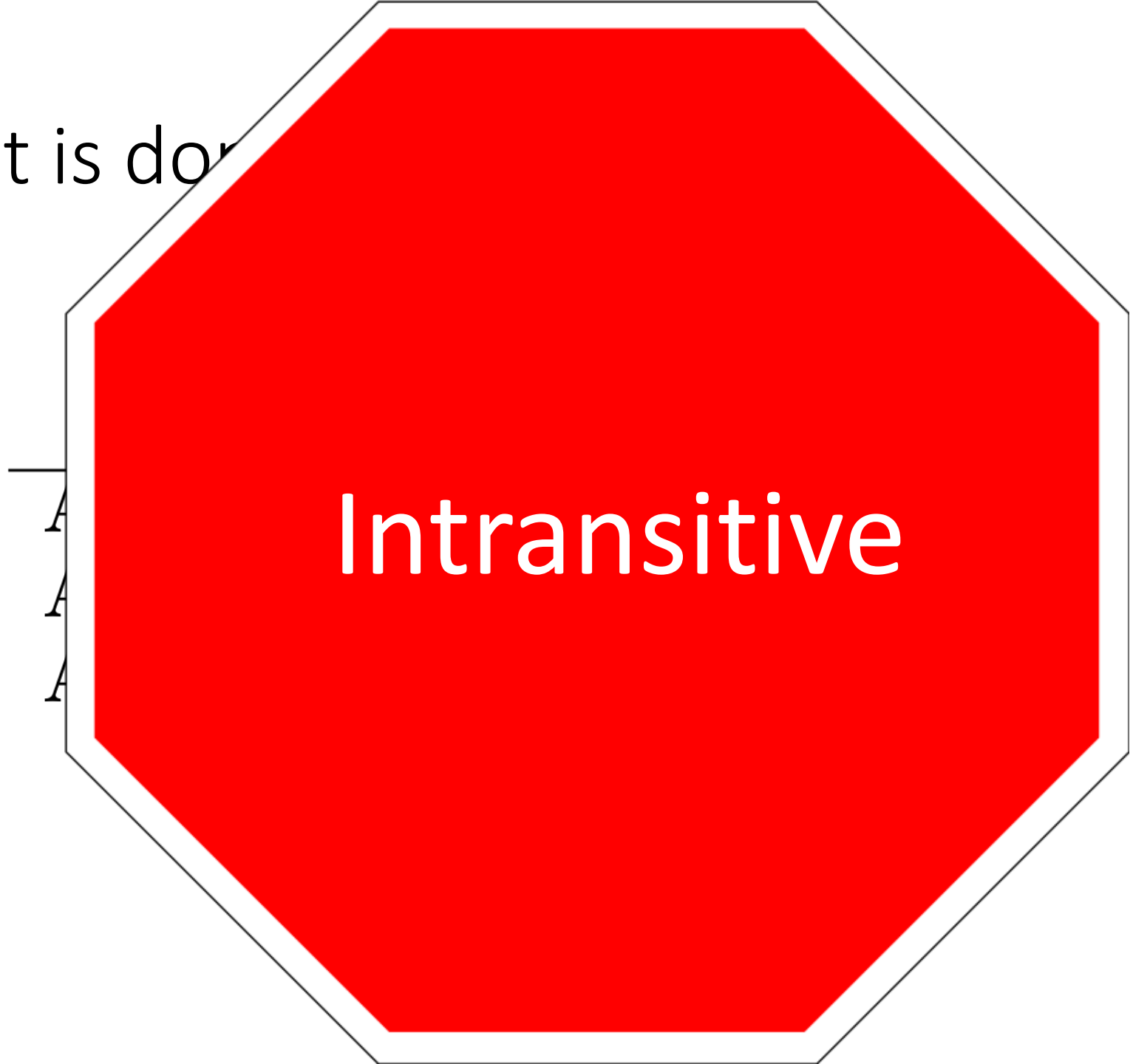
So what is do

Mostly use sm
games

Uninformative About Large Games



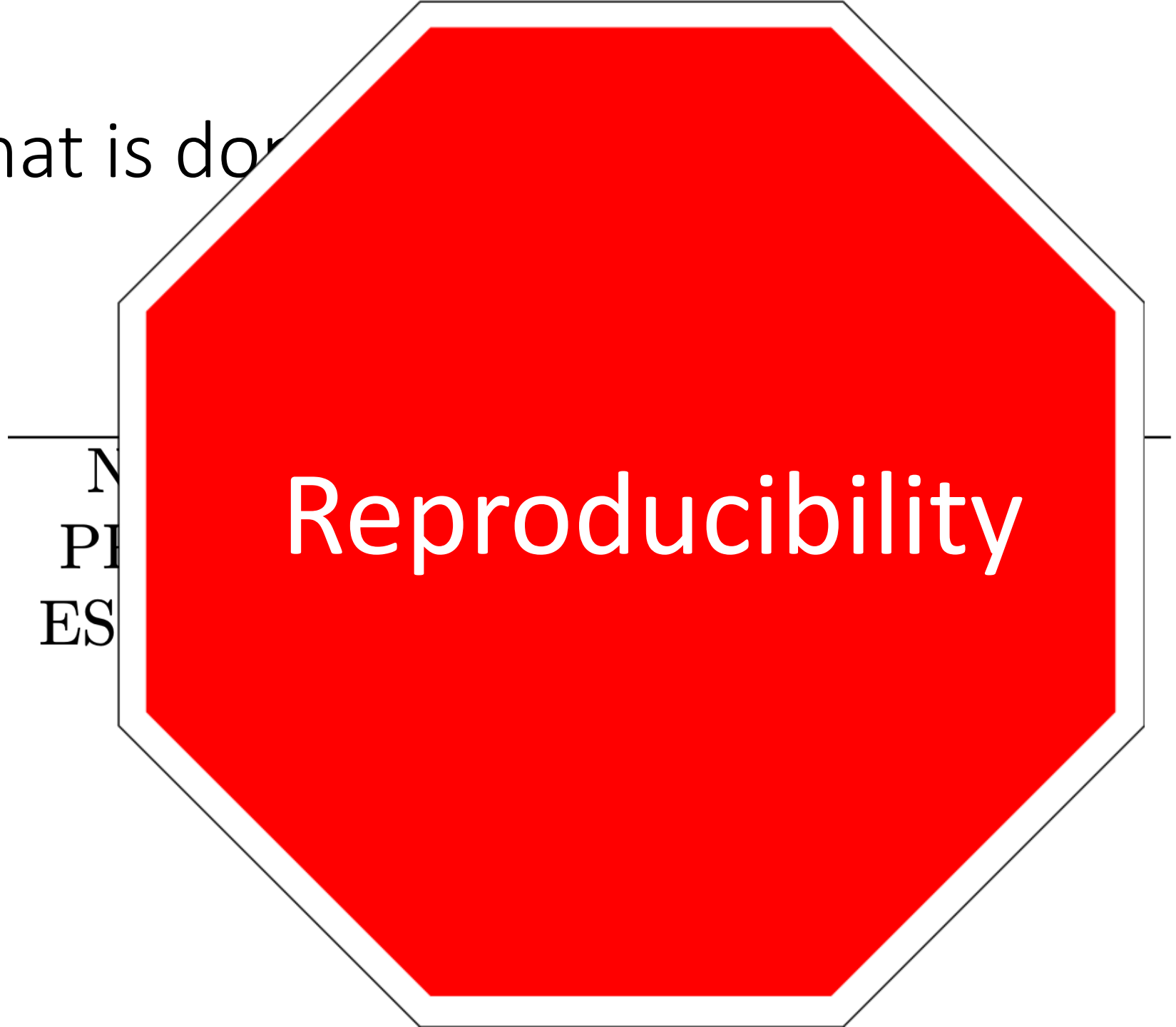
So what is do



So what is done? Head-to-heads

	NSFP	PPO-v1	PSRO
NSFP	0.0	−0.6	0.6
PPO-v1	0.6	0.0	−0.6
PSRO	−0.6	0.6	0.0

So what is do



Exp-a-spiel[1]: a new evaluation mechanism

- Extremely large games
- For which we can still solve exploitability exactly

Game	States	Info. states	Nash value ± error
DH3	19.12B	6.07M	1.00000 ± 0.00000
ADH3	29.31B	27.33M	0.38443 ± 0.00021
PTTT	19.93B	5.99M	0.66665 ± 0.00026
APTTT	27.12B	23.31M	0.55017 ± 0.00014

1: https://github.com/gabrfarina/exp-a-spiel

Exp-a-spiel: a new evaluation mechanism

- Exploitability in 1-2 minutes
- How? Multi-threading and efficient representations (sequence form).

Game	States	Info. states	Nash value ± error
DH3	19.12B	6.07M	1.00000 ± 0.00000
ADH3	29.31B	27.33M	0.38443 ± 0.00021
PTTT	19.93B	5.99M	0.66665 ± 0.00026
APTTT	27.12B	23.31M	0.55017 ± 0.00014

Computing exploitability

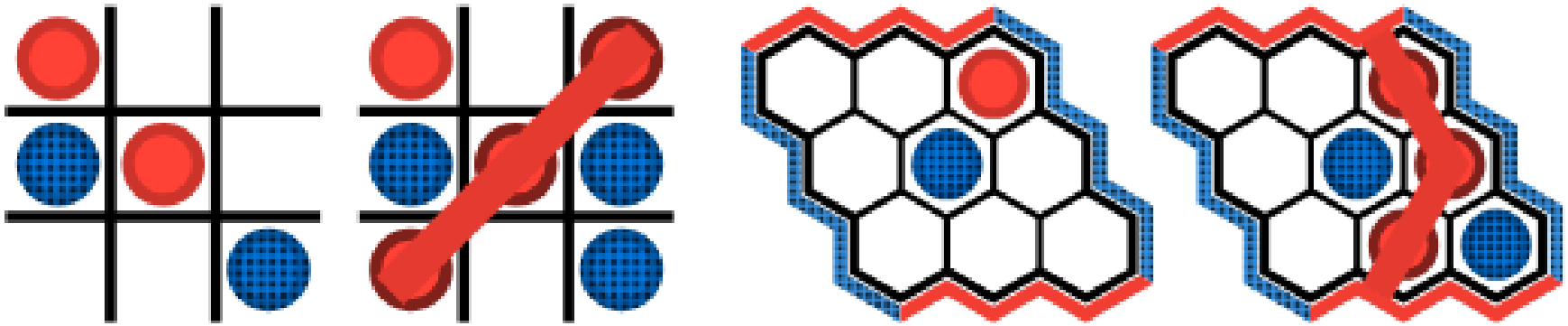
- Convert π_1, π_2 into their sequence form x, y: size in order of information states
- Compute gradients $\mathrm{g}_2 = -x^T A$ and $\mathrm{g}_1 = Ay$ materializing A on the fly
- Careful multi-threading to avoid c rrency issues
- Compute exploitability by solving

$$\max_{\hat{x} \in \mathcal{X}} \hat{x}^\top g_1 + \max_{\hat{y} \in \mathcal{Y}} y^\top g_2$$

Computing exploitability

Game	OpenSpiel	`exp-a-spiel`
Liar's Dice 1d4f	1	0.03
Liar's Dice 1d5f	1	0.03
Liar's Dice 1d6f	5	0.04
Liar's Dice 2d3f	10	0.04
Liar's Dice 2d4f	527	0.45
Liar's Dice 2d5f	OOM	11
Classical PTTT	OOM	27
Abrupt PTTT	OOM	27
Classical DH3	OOM	47
Abrupt DH3	OOM	47

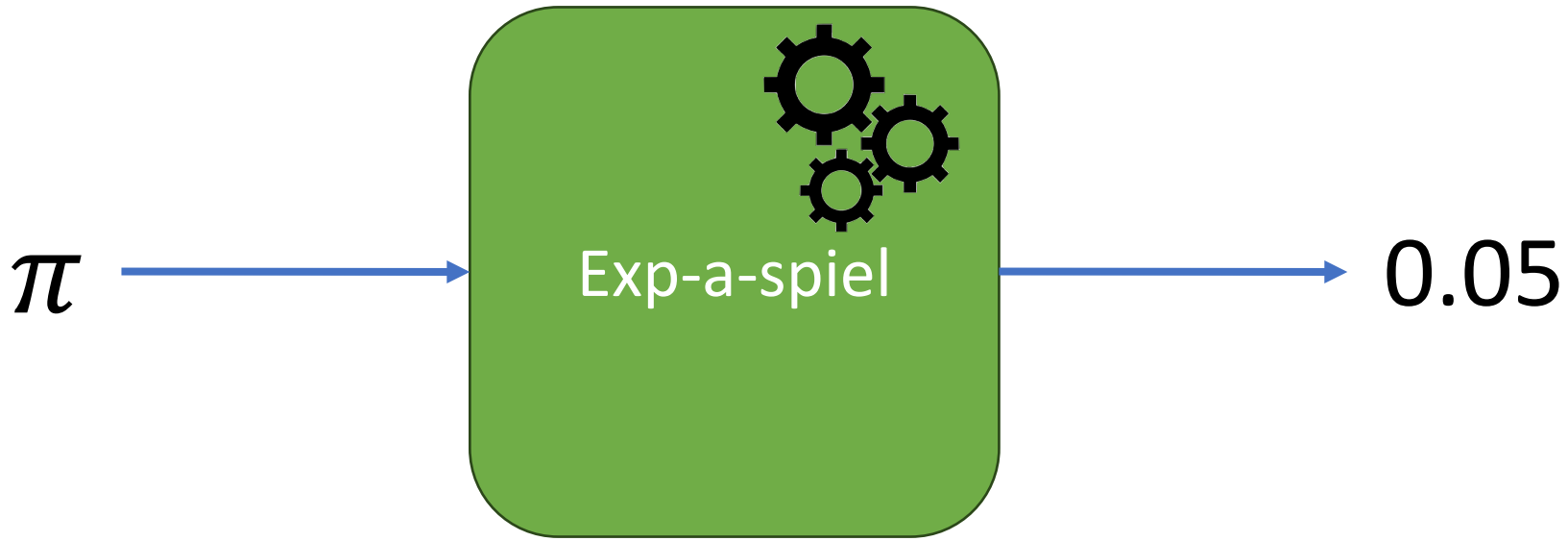
The games themselves



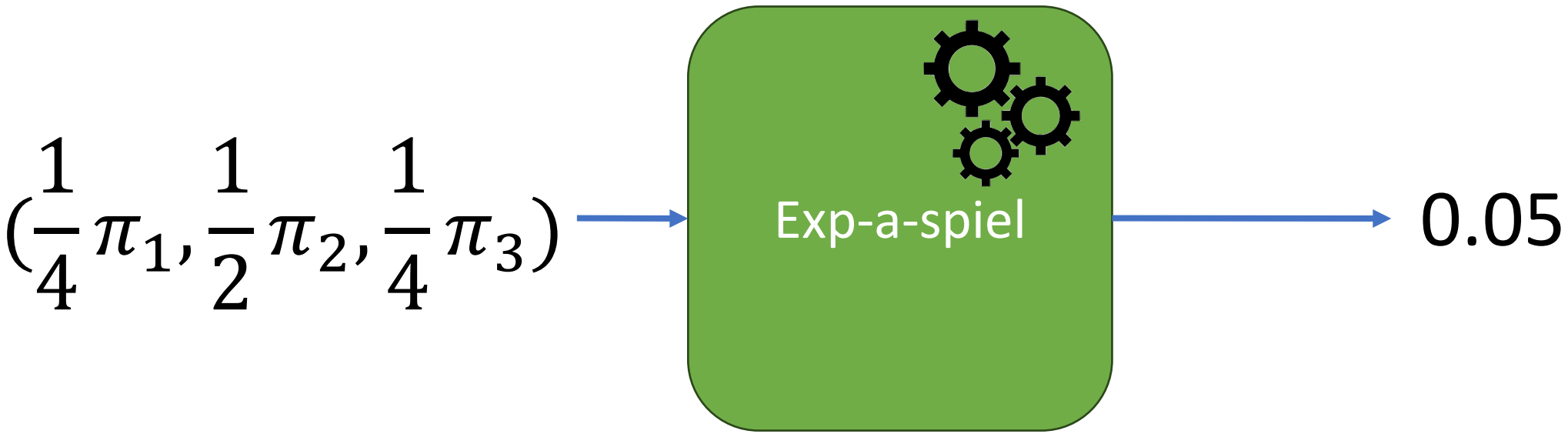
The games themselves

Playable agents

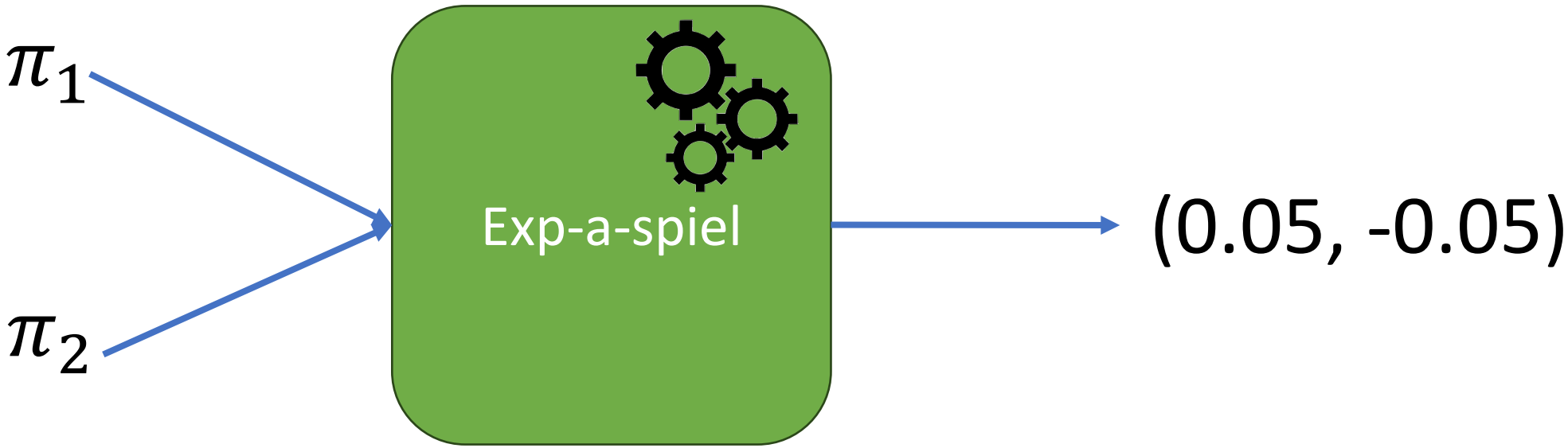
The exp-a-spiel interface



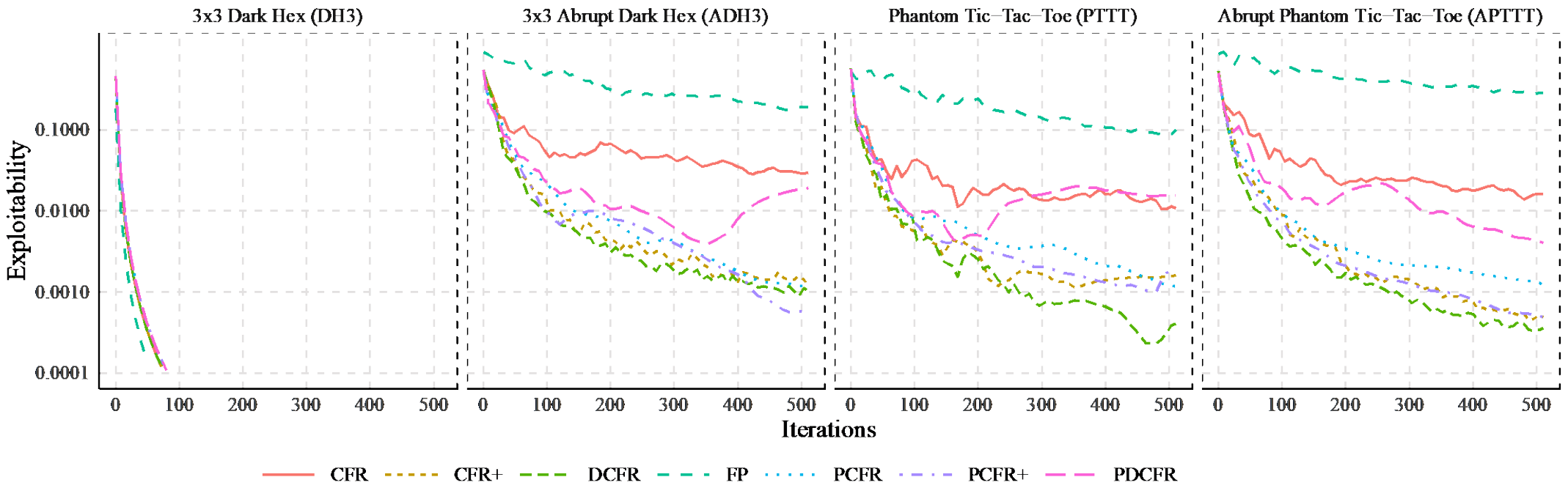
The exp-a-spiel interface



The exp-a-spiel interface

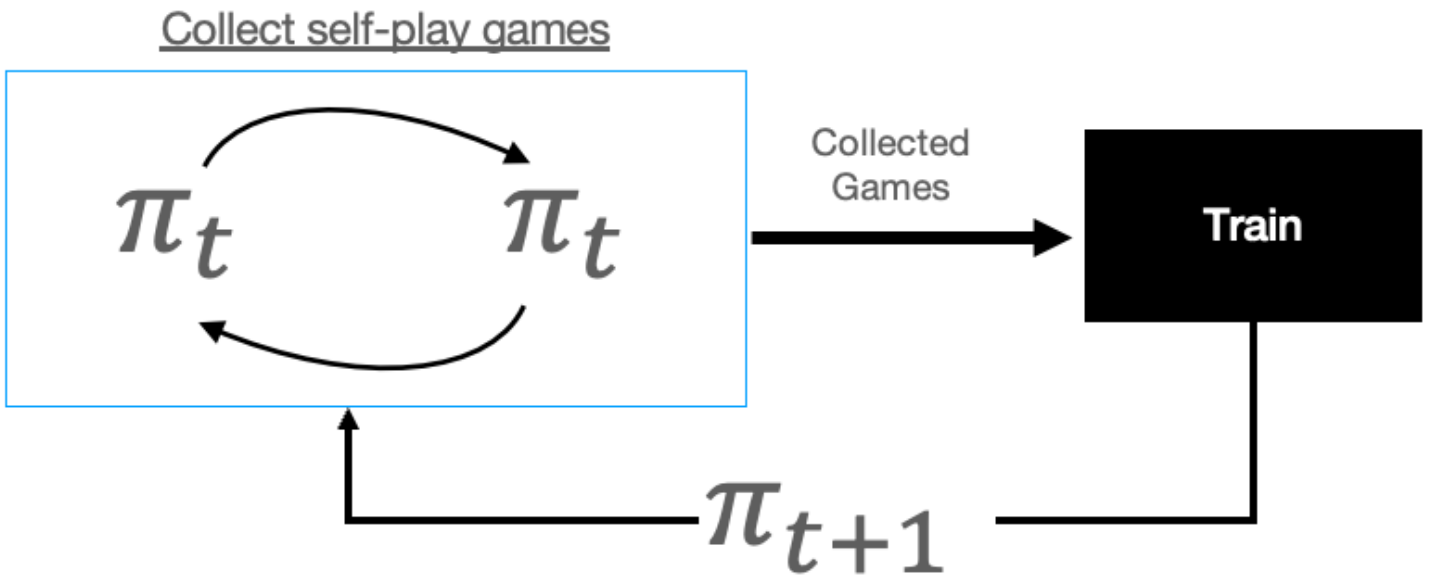


Pre-computed near-Nash policies

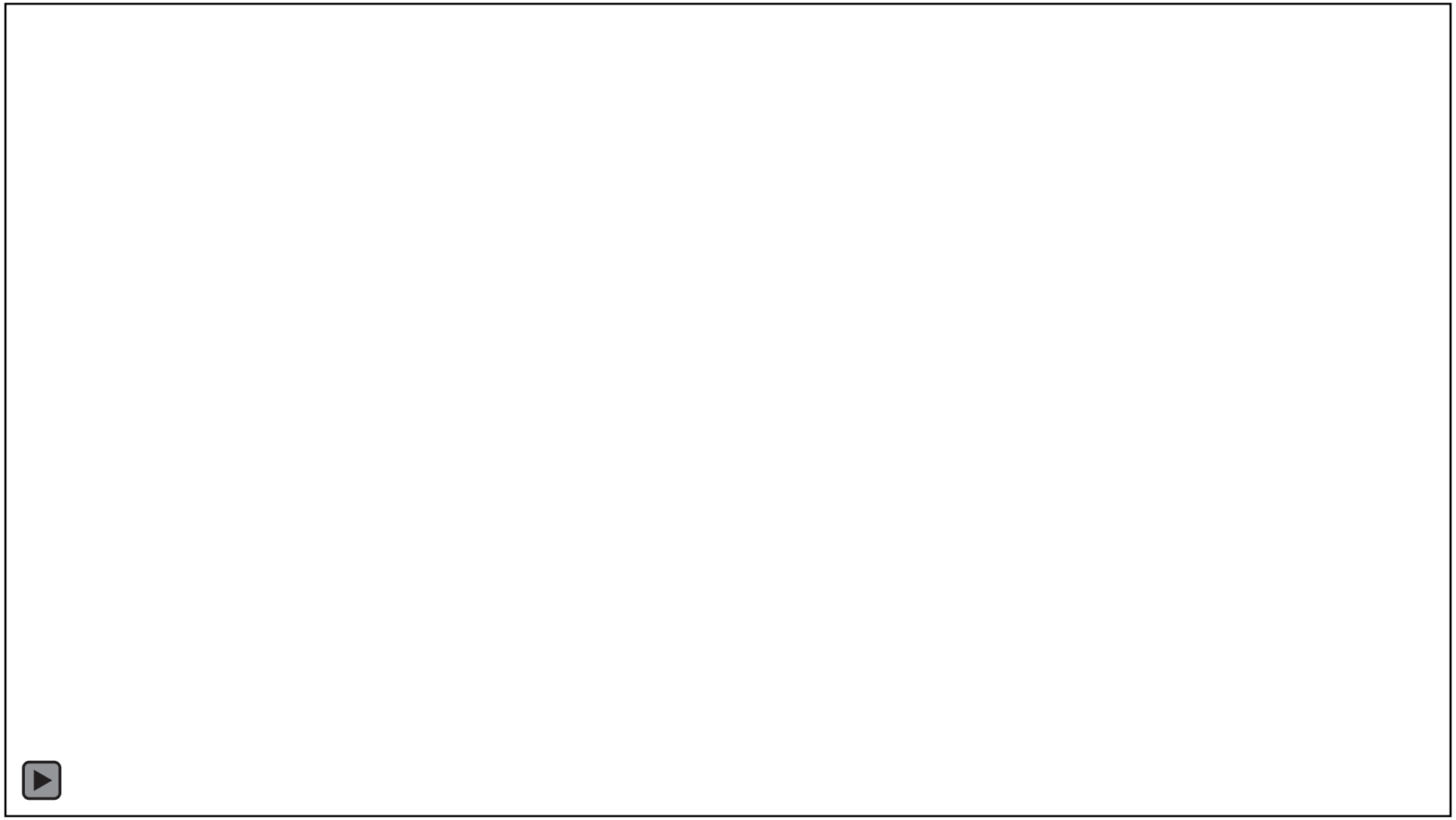


Using exp-a-spiel as an investigative tool: are regularized policy gradient methods competitive?

Tackling IIGs with self-play RL



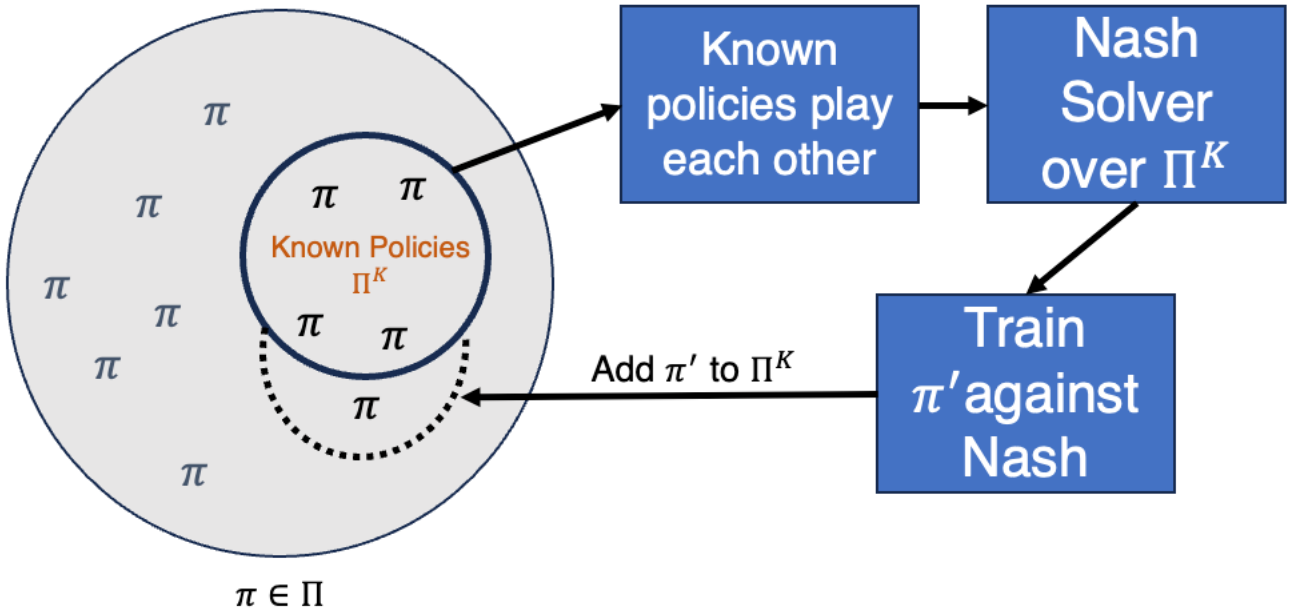
Naïve self-play RL doesn’t work in IIGs



What is done instead? Make custom algorithms

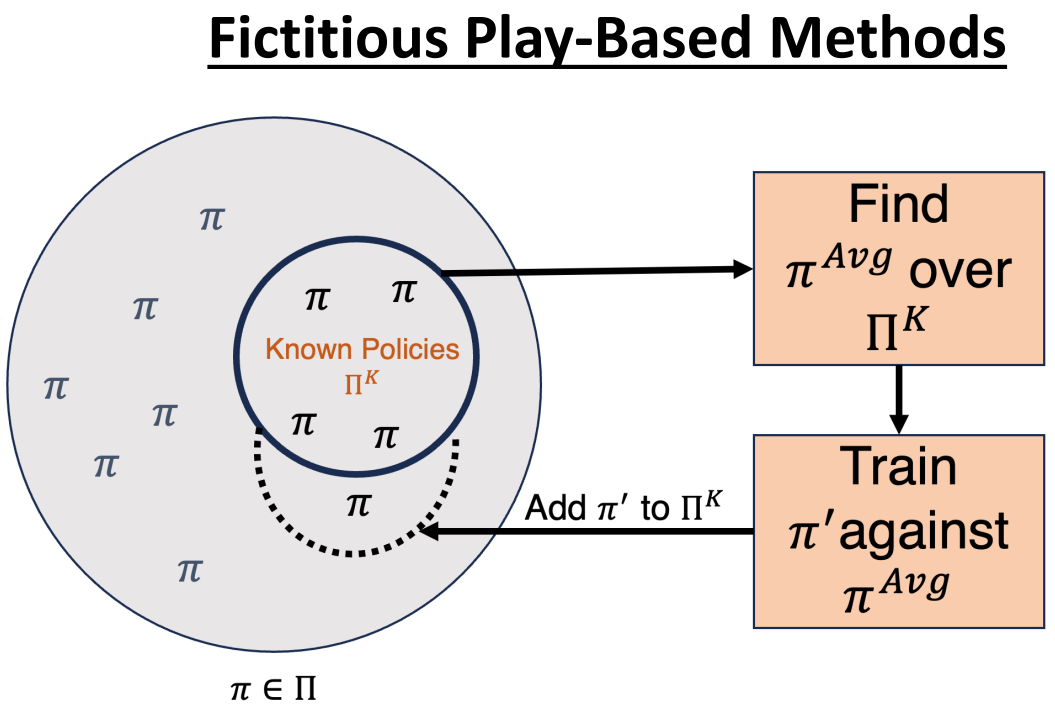
- Derive RL algorithms from tabularly convergent algorithms

Double Oracle-Based Methods



What is done instead? Make custom algorithms

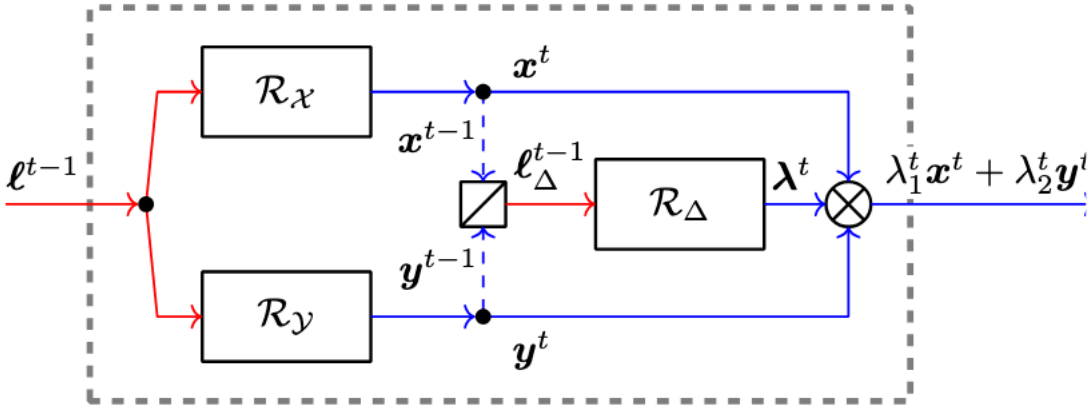
- Derive RL algorithms from tabularly convergent algorithms



What is done instead? Make custom algorithms

- Derive RL algorithms from tabularly convergent algorithms

Counterfactual Regret Minimization-Based Methods



Source: Computational Game Solving, CMU, Gabriele Farina

Weaknesses of these algorithms

Slow convergence

Converge in time average

Not always plug-and-play

$$\pi(T) \neq \pi_{NASH} = \frac{1}{T}\sum_{i} \pi(t)$$

An alternative: regularization

Magnetic Mirror Descent[1]:

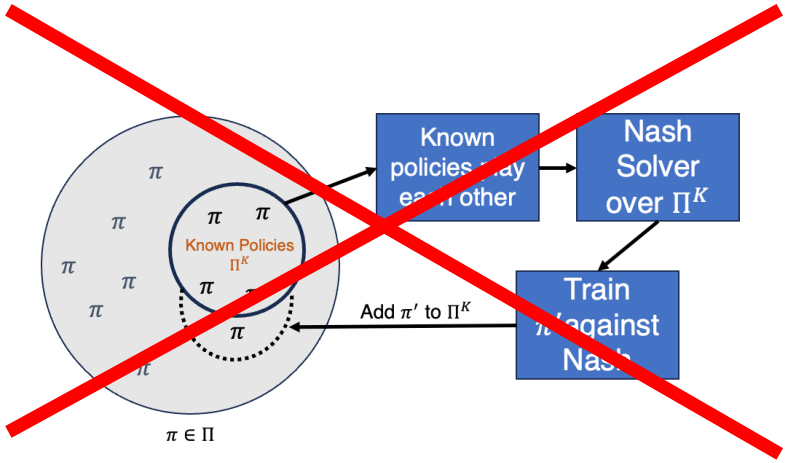
- Competitive with CFR in tabular settings
- Maximize the reward
- Control update size
- An entropy bonus
- Looks like your good old regularized policy gradient algorithm!

1: “A Unified Approach to Reinforcement Learning, Quantal Response Equilibria, and Two Player Zero-Sum Games” by Sokota, D’Orazio et. al.

The Policy Gradient Hypothesis. Appropriately tuned policy gradient methods that share an ethos with magnetic mirror descent are competitive with or superior to model-free deep reinforcement learning approaches based on fictitious play, double oracle, or counterfactual regret minimization in two-player zero-sum imperfect-information games.

Also, if the PG hypothesis was true…

Single policy



Converge in last iterate

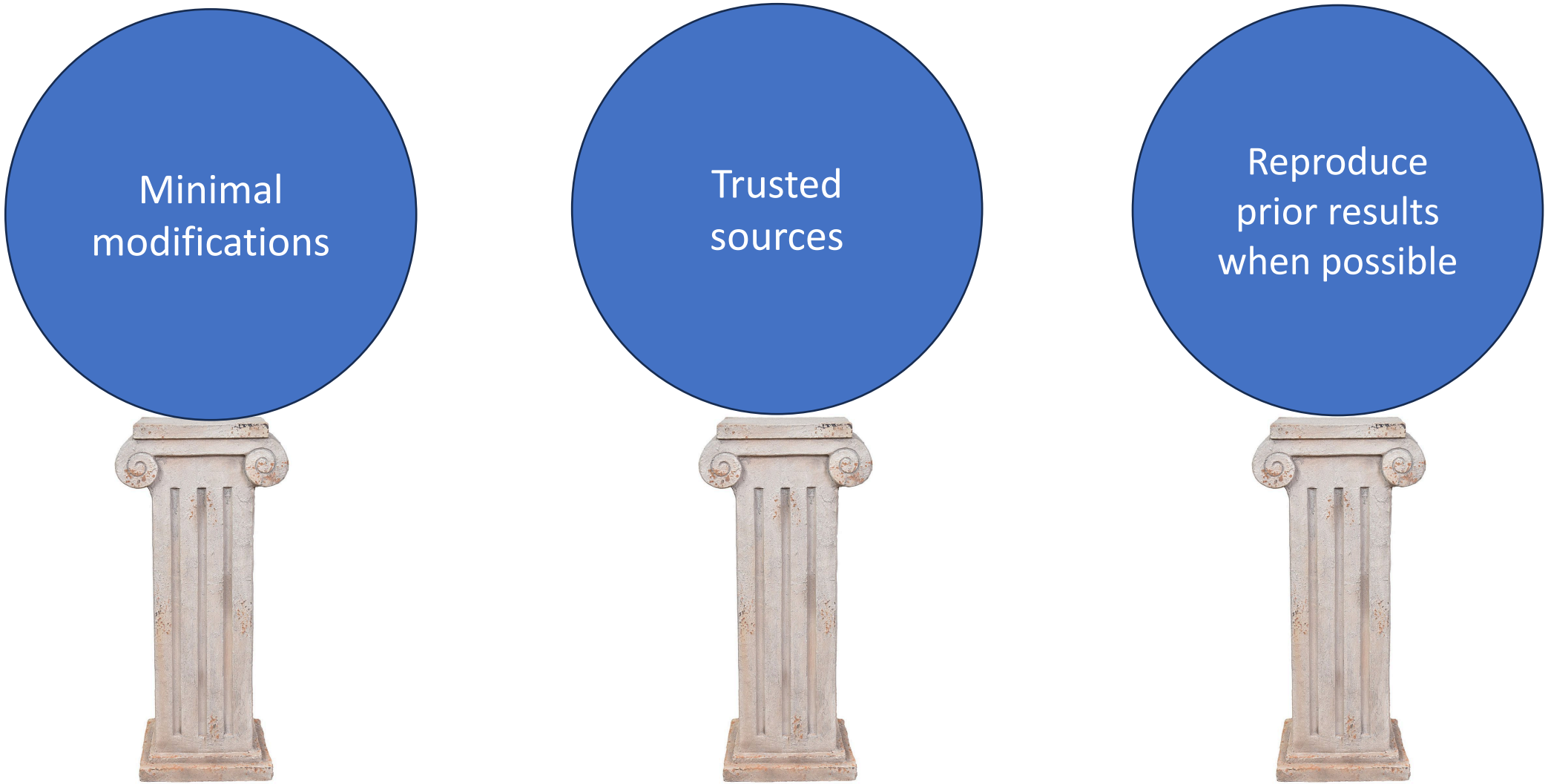
$$\pi(T) \approx \pi_{NASH}$$

Simple single-agent RL

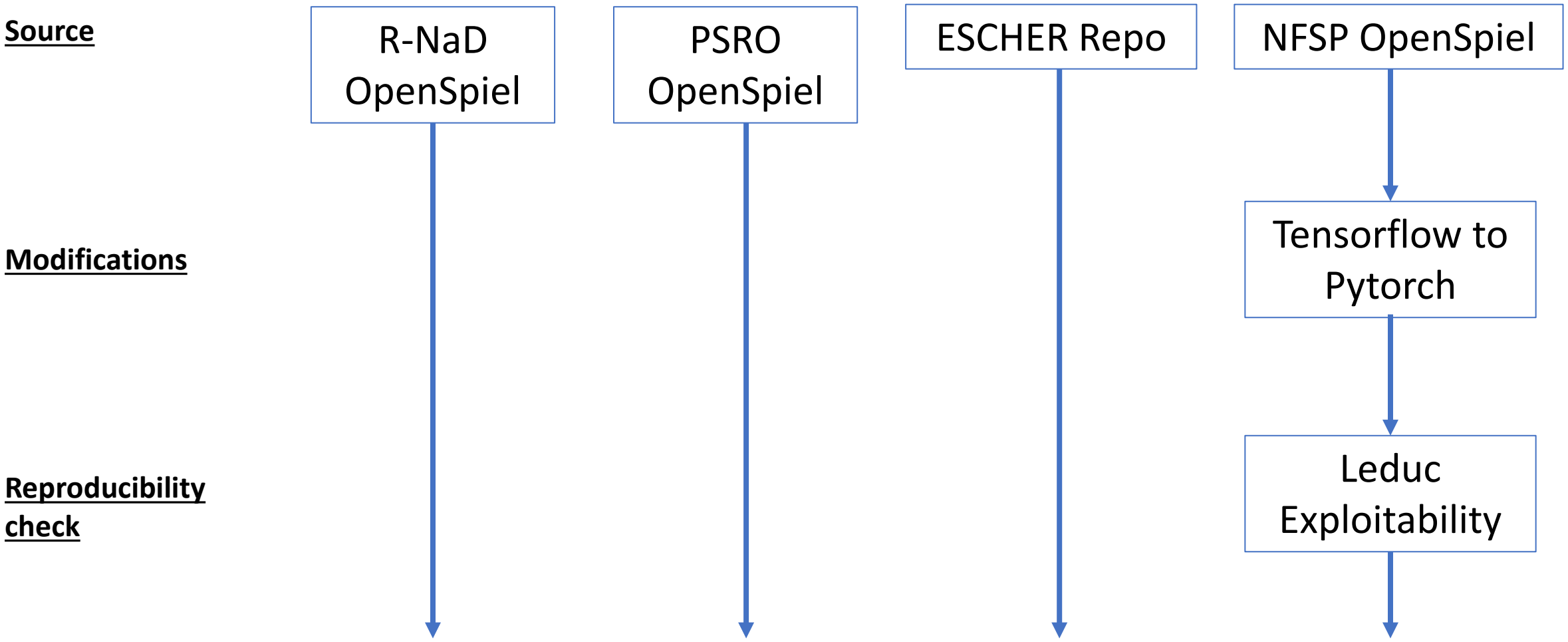
Algorithms we test

- One standard algorithm for each “competitor” class:
  - NFSP – fictitious play
  - PSRO – double oracle
  - ESCHER – counterfactual regret-type
  - R-NaD – regularized algorithm
- Policy gradients
  - PPO
  - MMD
  - PPG

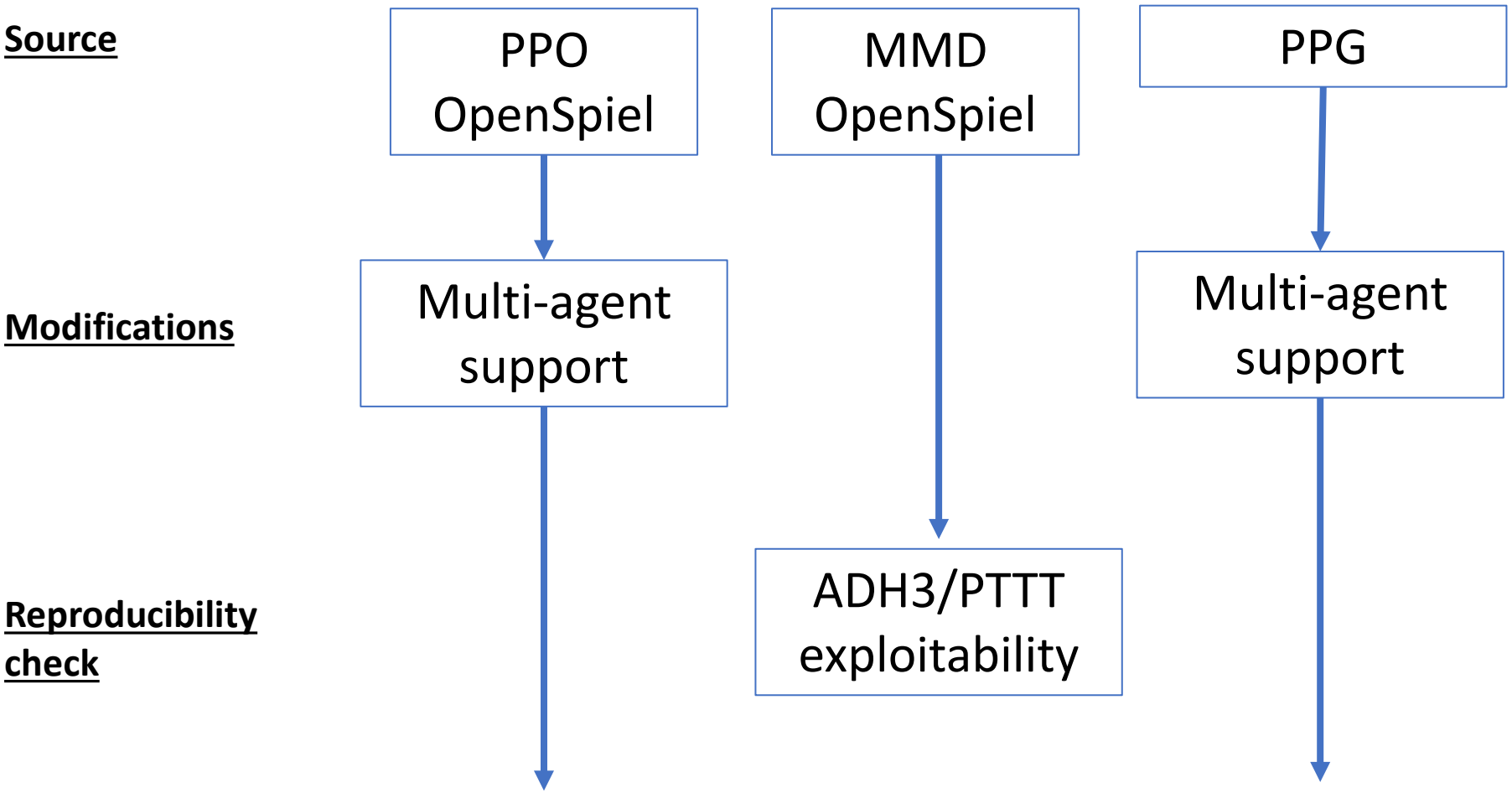
Sourcing algorithm considerations



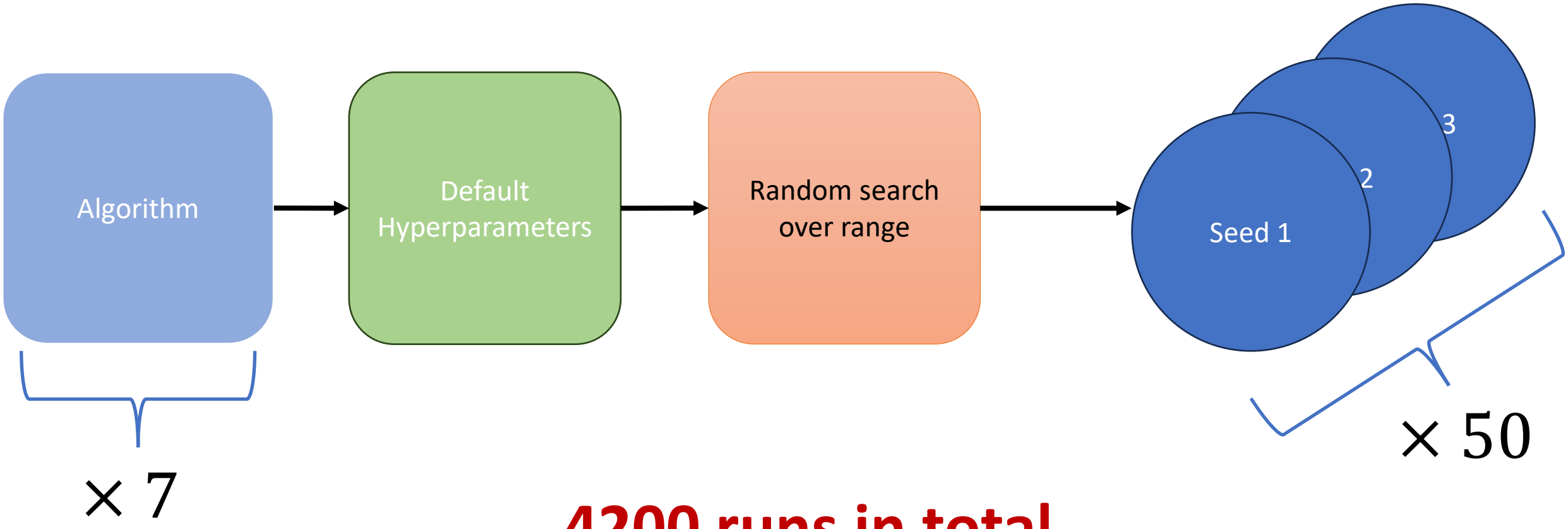
Implementation safety



Implementation safety



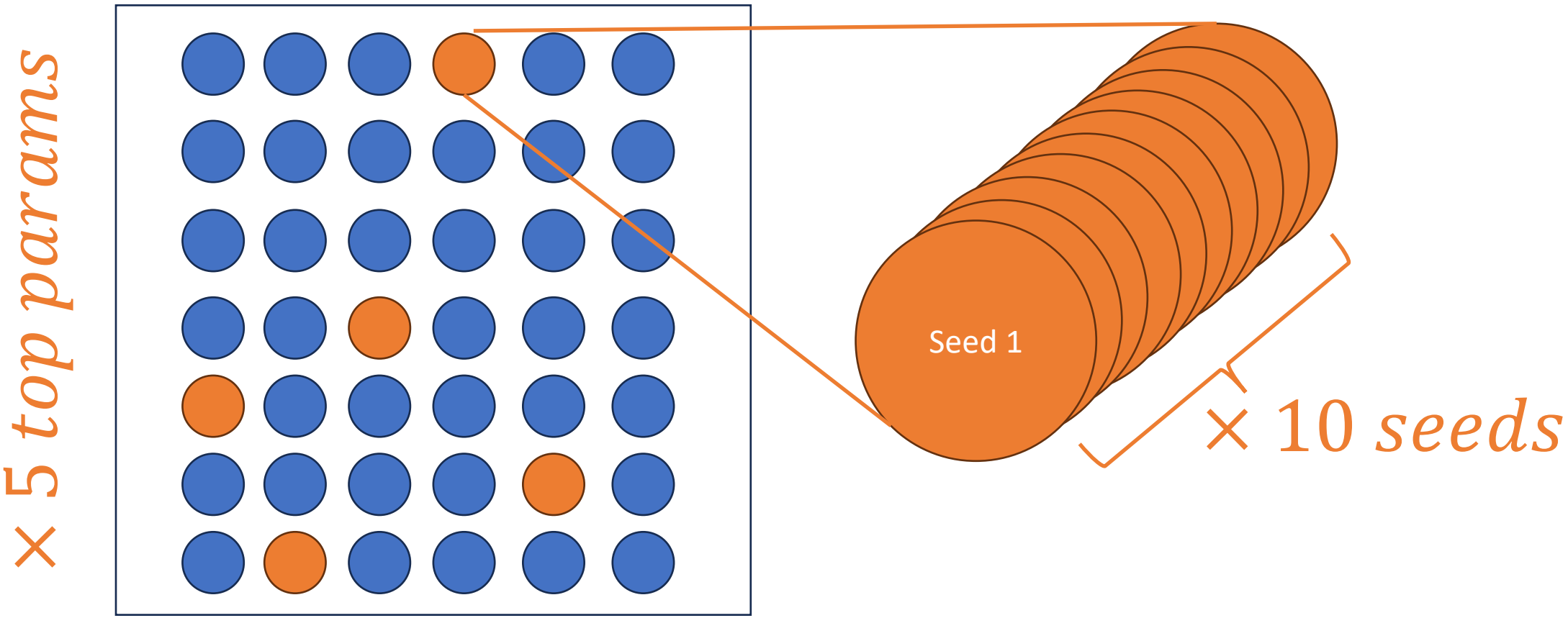
Tuning Procedure – Round 1 (4 games)



4200 runs in total

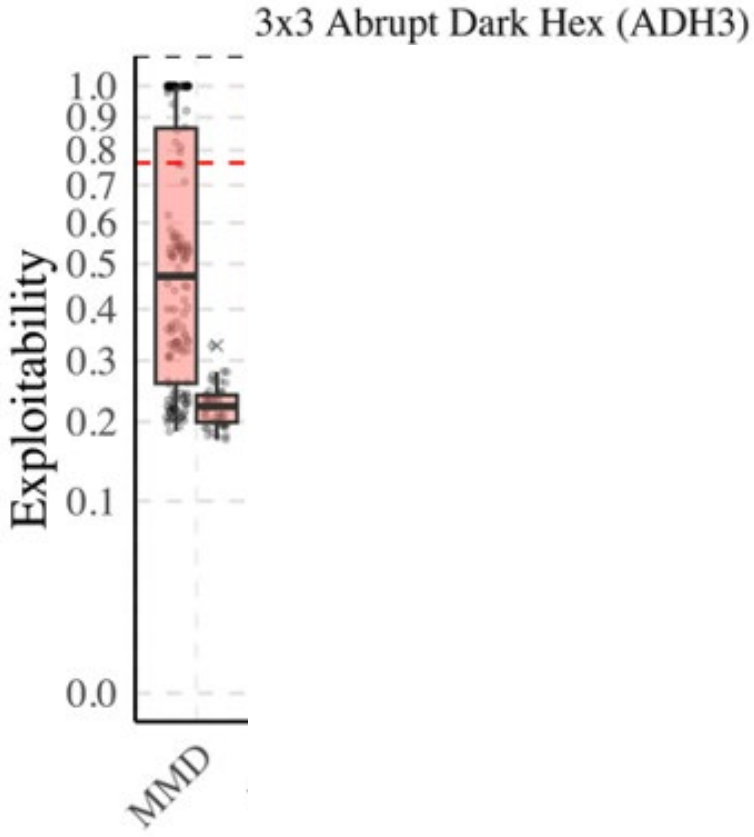
Tuning Procedure – Round 2

1400 runs in total

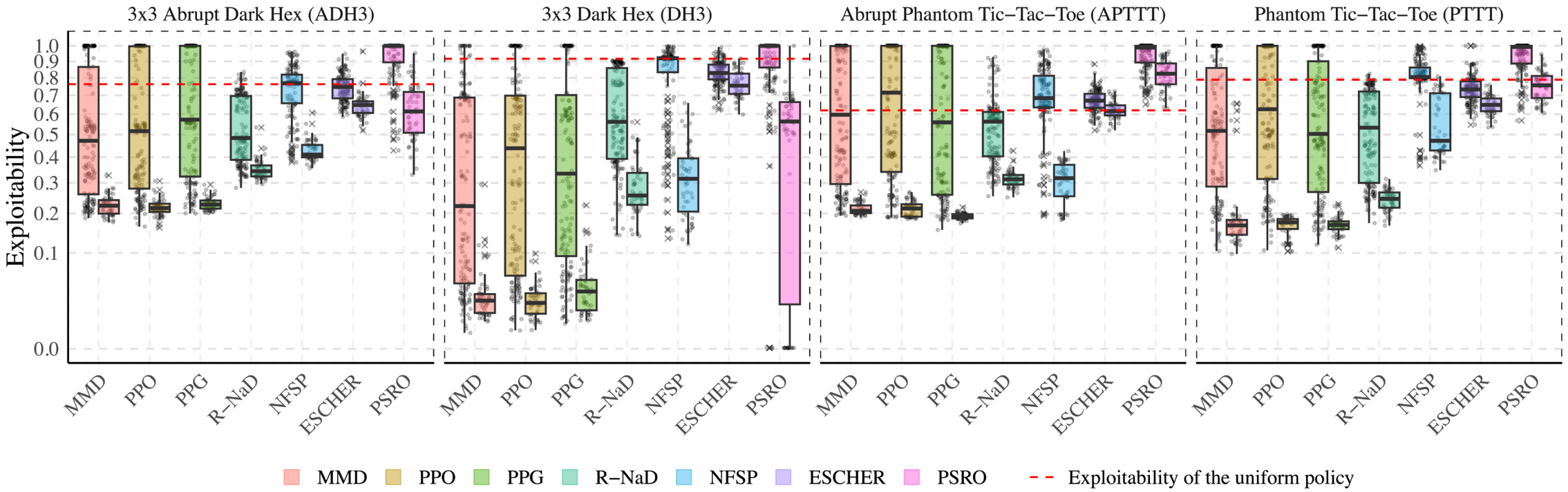


So how’s it shake out?

Exploitability



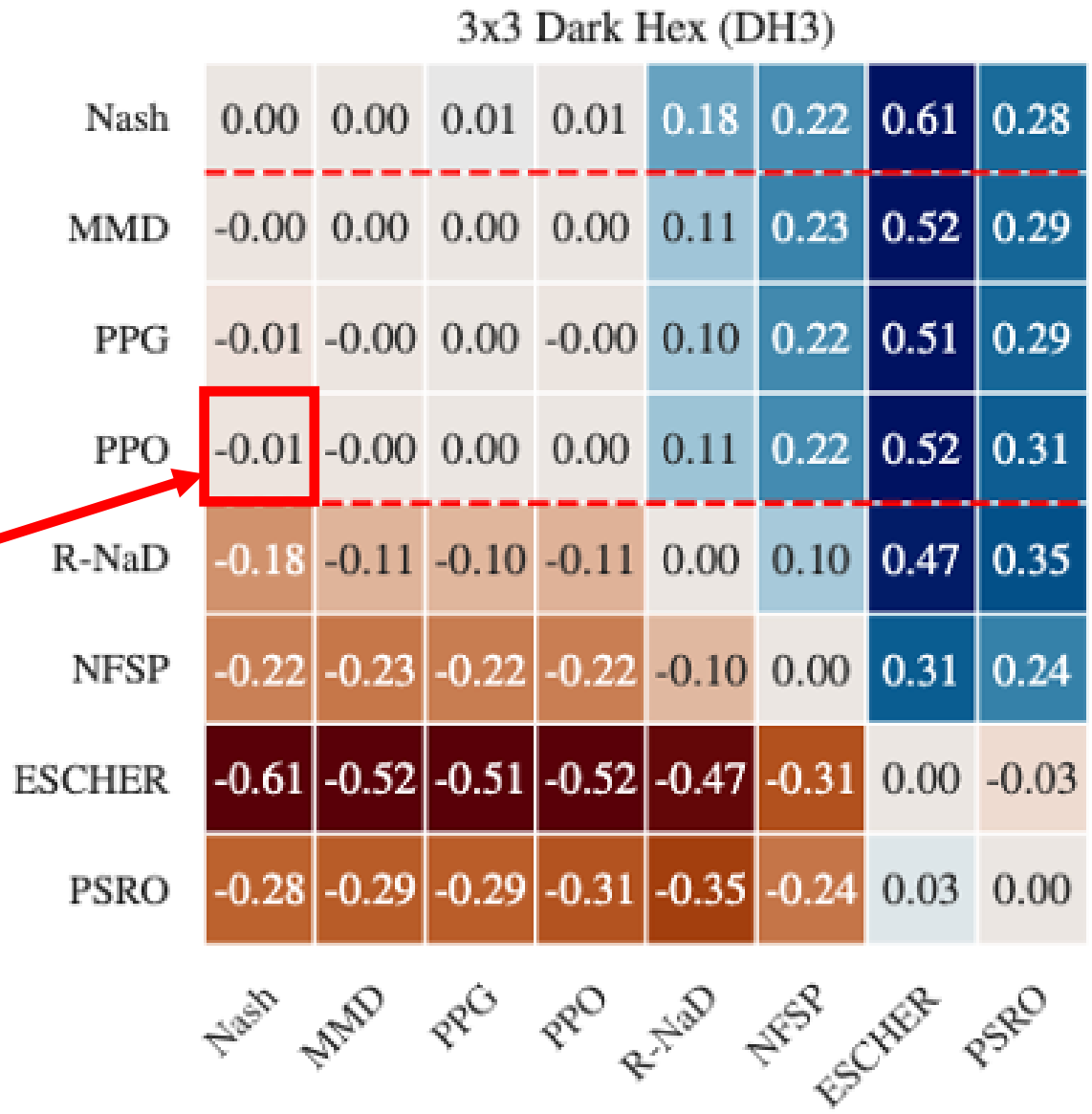
Exploitability



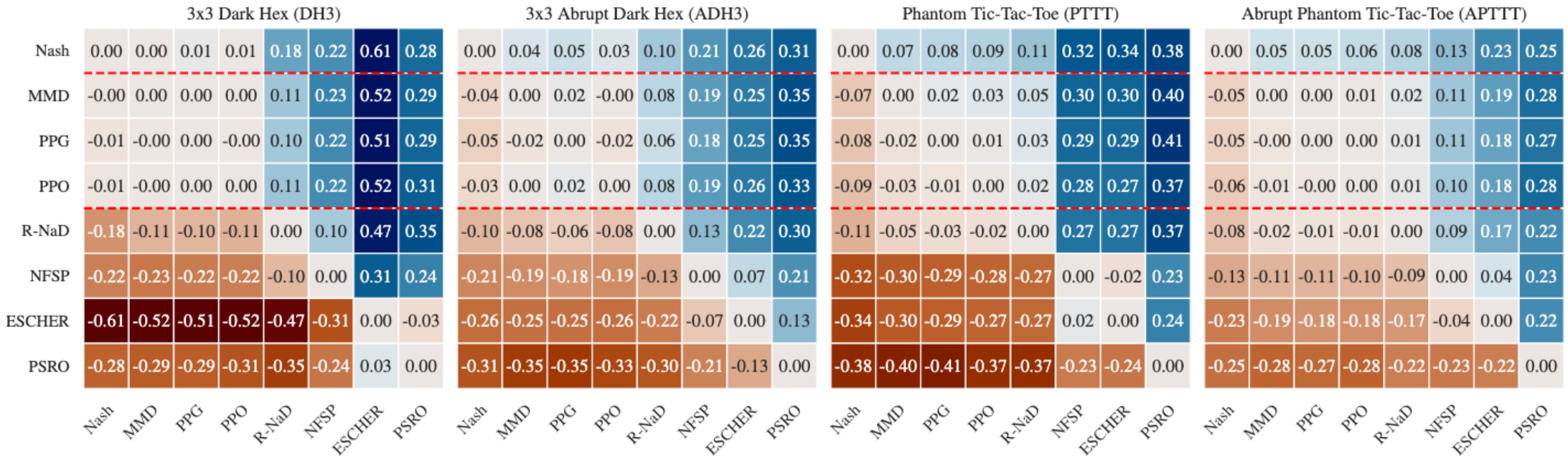
Head-to-head

PPO vs. Nash

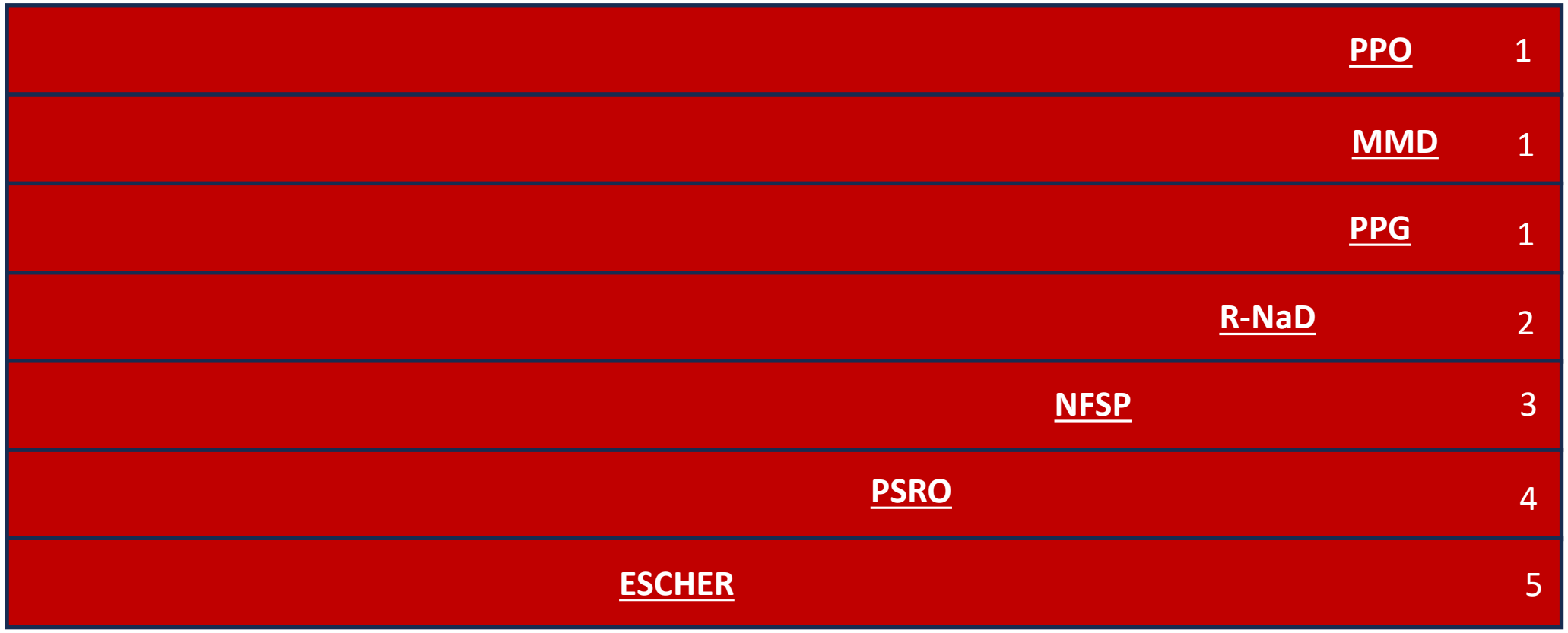
Close to 0 is good



Head-to-head

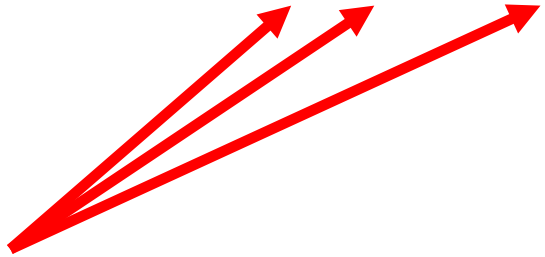

3x3 Dark Hex (DH3)
Nash 0.00 0.00 0.01 0.01 0.18 0.22 0.61 0.28
MMD -0.00 0.00 0.00 0.00 0.11 0.23 0.52 0.29
PPG -0.01 -0.00 0.00 -0.00 0.10 0.22 0.51 0.29
PPO -0.01 -0.00 0.00 0.00 0.11 0.22 0.52 0.31
R-NaD -0.18 -0.11 -0.10 -0.11 0.00 0.10 0.47 0.35
NFSP -0.22 -0.23 -0.22 -0.22 -0.10 0.00 0.31 0.24
ESCHER -0.61 -0.52 -0.51 -0.52 -0.47 -0.31 0.00 -0.03
PSRO -0.28 -0.29 -0.29 -0.31 -0.35 -0.24 0.03 0.00
Nash MMD PPG PPO R-NaD NFSP ESCHER PSRO
3x3 Abrupt Dark Hex (ADH3)
Nash 0.00 0.04 0.05 0.03 0.10 0.21 0.26 0.31
MMD -0.04 0.00 0.02 -0.00 0.08 0.19 0.25 0.35
PPG -0.05 -0.02 0.00 -0.02 0.06 0.18 0.25 0.35
PPO -0.03 0.00 0.02 0.00 0.08 0.19 0.26 0.33
R-NaD -0.10 -0.08 -0.06 -0.08 0.00 0.13 0.22 0.30
NFSP -0.21 -0.19 -0.18 -0.19 -0.13 0.00 0.07 0.21
ESCHER -0.26 -0.25 -0.25 -0.26 -0.22 -0.07 0.00 0.13
PSRO -0.31 -0.35 -0.35 -0.33 -0.30 -0.21 -0.13 0.00
Nash MMD PPG PPO R-NaD NFSP ESCHER PSRO
Phantom Tic-Tac-Toe (PTTT)
Nash 0.00 0.07 0.08 0.09 0.11 0.32 0.34 0.38
MMD -0.07 0.00 0.02 0.03 0.05 0.30 0.30 0.40
PPG -0.08 -0.02 0.00 0.01 0.03 0.29 0.29 0.41
PPO -0.09 -0.03 -0.01 0.00 0.02 0.28 0.27 0.37
R-NaD -0.11 -0.05 -0.03 -0.02 0.00 0.27 0.27 0.37
NFSP -0.32 -0.30 -0.29 -0.28 -0.27 0.00 -0.02 0.23
ESCHER -0.34 -0.30 -0.29 -0.27 -0.27 0.02 0.00 0.24
PSRO -0.38 -0.40 -0.41 -0.37 -0.37 -0.23 -0.24 0.00
Nash MMD PPG PPO R-NaD NFSP ESCHER PSRO
Abrupt Phantom Tic-Tac-Toe (APTTT)
Nash 0.00 0.05 0.05 0.06 0.08 0.13 0.23 0.25
MMD -0.05 0.00 0.00 0.01 0.02 0.11 0.19 0.28
PPG -0.05 -0.00 0.00 0.00 0.01 0.11 0.18 0.27
PPO -0.06 -0.01 -0.00 0.00 0.01 0.10 0.18 0.28
R-NaD -0.08 -0.02 -0.01 -0.01 0.00 0.09 0.17 0.22
NFSP -0.13 -0.11 -0.11 -0.10 -0.09 0.00 0.04 0.23
ESCHER -0.23 -0.19 -0.18 -0.18 -0.17 -0.04 0.00 0.22
PSRO -0.25 -0.28 -0.27 -0.28 -0.22 -0.23 -0.22 0.00
Nash MMD PPG PPO R-NaD NFSP ESCHER PSRO

A consistent trend



Why hasn’t this been widely observed? Some hypotheses.

Why hasn’t this been widely observed? Some hypotheses.



The 3 most common RL libraries

Why hasn’t this been widely observed? Some hypotheses.

- **Data sharing 🔴 – bad baselines**
  - Good policies aren’t shared across papers
  - Per-paper baseline tuning is hard

Why hasn't this been widely observed? Some hypotheses.

- Standard benchmark scale too small

Game	Number of Infosets	Number of States (including terminal)
ADH2	94	471
Leduc	936	9457
LD1D4F	1024	8181
2x2 Battle Ship!	3286	10069
LD1D6F	24576	294883

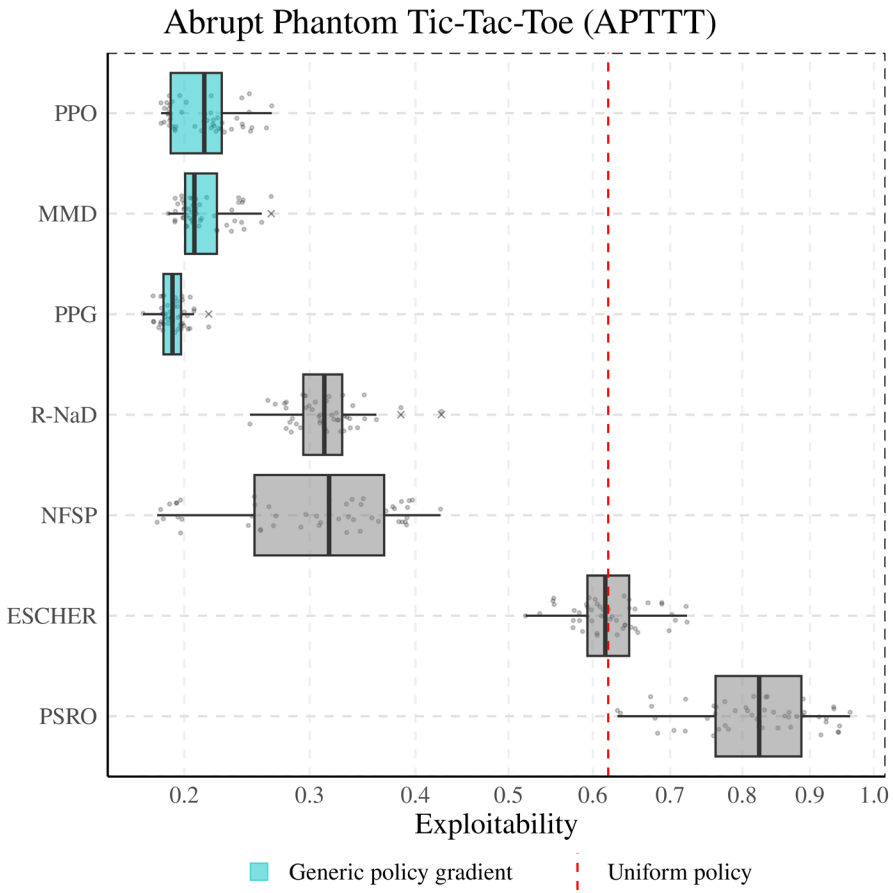
Paper	Exact Exploitability	Approx. Exploitability or Head-to-Head
NFSP (Heinrich and Silver, 2016)	Leduc	
PSRO (Lanctot et al., 2017)	Leduc	
ESCHER (McAleer et al., 2023)	2x2 Battle Ship!, Leduc	Dark Chess, DH5, PTTT
RNaD (Perolat et al., 2022)		Stratego
MMD (Sokota et al., 2023)	LD1D4F, Leduc, ADH2	ADH3, PTTT
Qin et al. (2019)	Kuhn Poker	
AdaptFSP (Goldstein and Brown, 2023)	Leduc	
Hu et al. (2023)	Leduc	
NSFP-PLT (Li et al., 2023a)		6-, 3-, 2-player poker, Leduc
Pipeline PSRO (McAleer et al., 2020)	Leduc	Barrage
XDO (McAleer et al., 2021)	Leduc	
Smith et al. (2021)	Leduc	
Anytime PSRO(McAleer et al., 2022a)		Leduc
Online DO (Dinh et al., 2022)		
Zhou et al. (2022)		Leduc
PDO (Yao et al., 2023)	Leduc	
SP-PSRO (McAleer et al., 2022b)	LD1D6F, 2x2 Battle Ship!, Leduc	
Self-Adaptive PSRO (Li et al., 2024a)	LD1D6F, Leduc	
Fusion PSRO (Lian, 2024)	LD1D6F, Leduc	
Dream (Steinberger et al., 2020)	Leduc	Flop Hold'em
ARMAC (Gruslys et al., 2020)	LD1D6F, Leduc	Head's up
FoReL (Perolat et al., 2021)	LD1D6F, Leduc	
Meng et al. (2023)	Leduc	PTTT, DH5, DH4
Liu et al. (2024)	Leduc	

Cautions and limitations

- Wrong hyperparameter regime?
- Our games are homogeneous
  - Only sparse reward at game end
  - Deterministic dynamics
  - All partial observability from actions
- So many algorithms we haven’t tried!
- Test out hypothesis on our code!

Quick takeaways

- We find regularized policy gradient algorithms to outperform every multi-agent specific algorithm we test



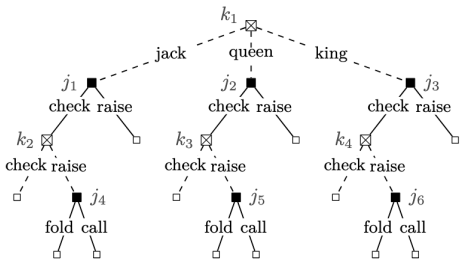
Quick takeaways

- We find regularized policy gradient algorithms to outperform every multi-agent specific algorithm we test
- New benchmarks that let us exactly measure algorithm performance

Game	States	Info. states	Nash value ± error
DH3	19.12B	6.07M	1.00000 ± 0.00000
ADH3	29.31B	27.33M	0.38443 ± 0.00021
PTTT	19.93B	5.99M	0.66665 ± 0.00026
APTTT	27.12B	23.31M	0.55017 ± 0.00014

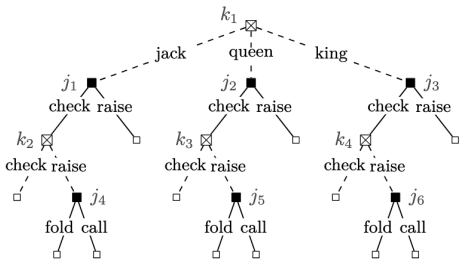
Future work and speculation

More games!



A missing middle

More games!

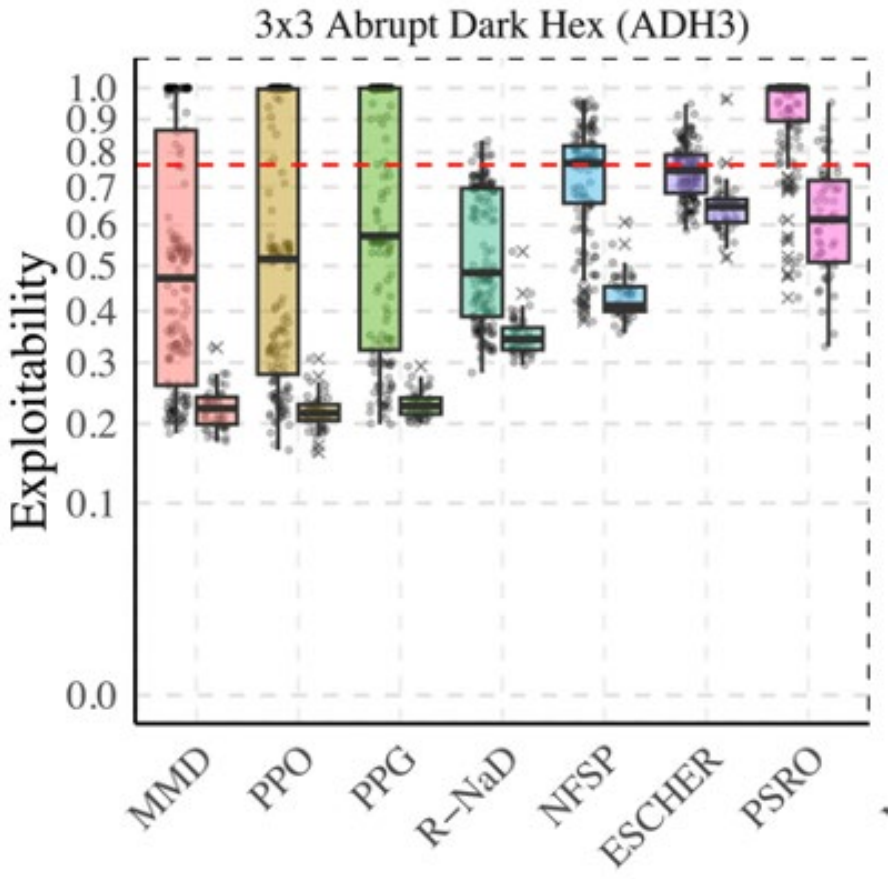


Dense reward games

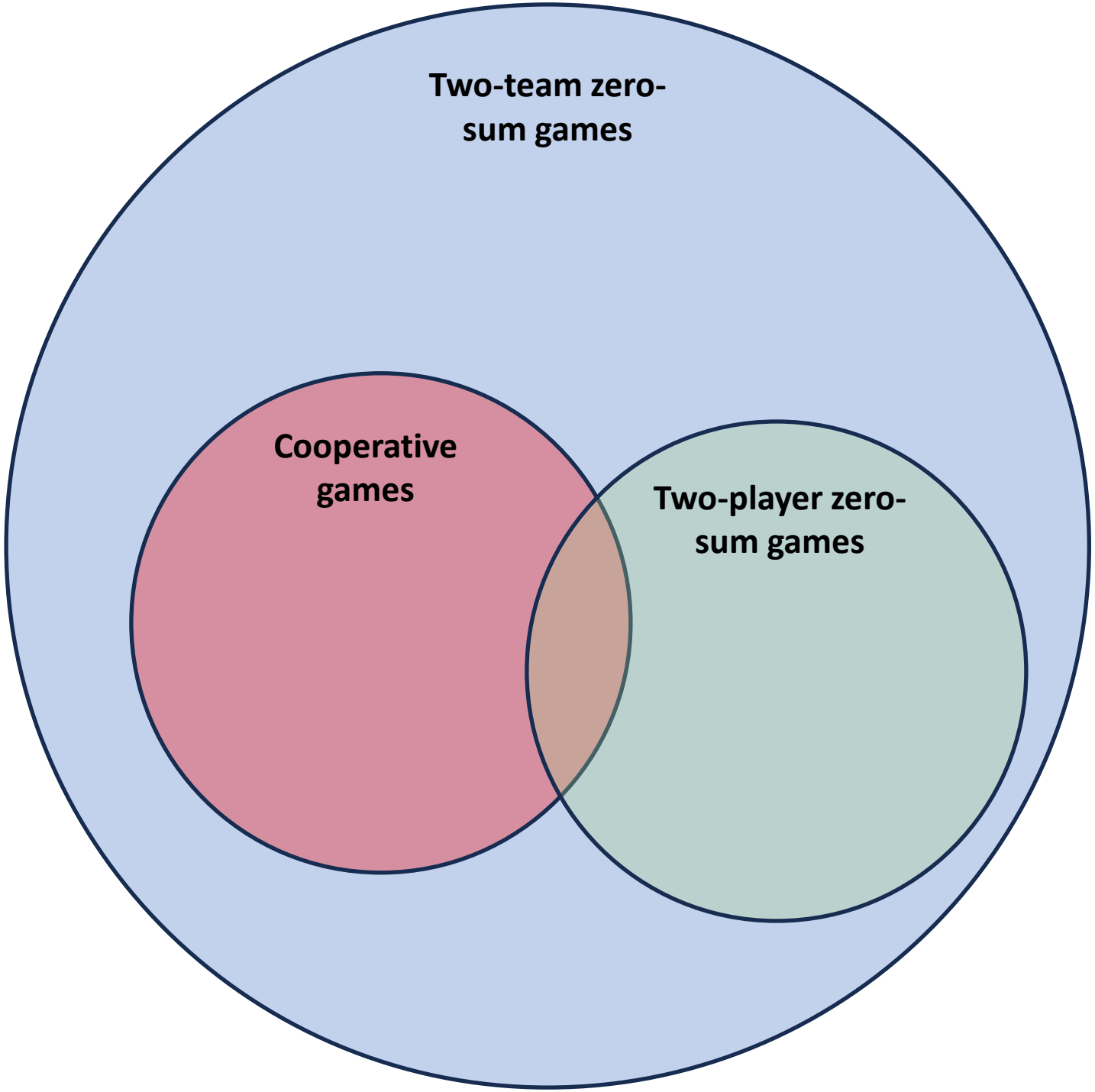
Stochastic games

Varied amounts of memory

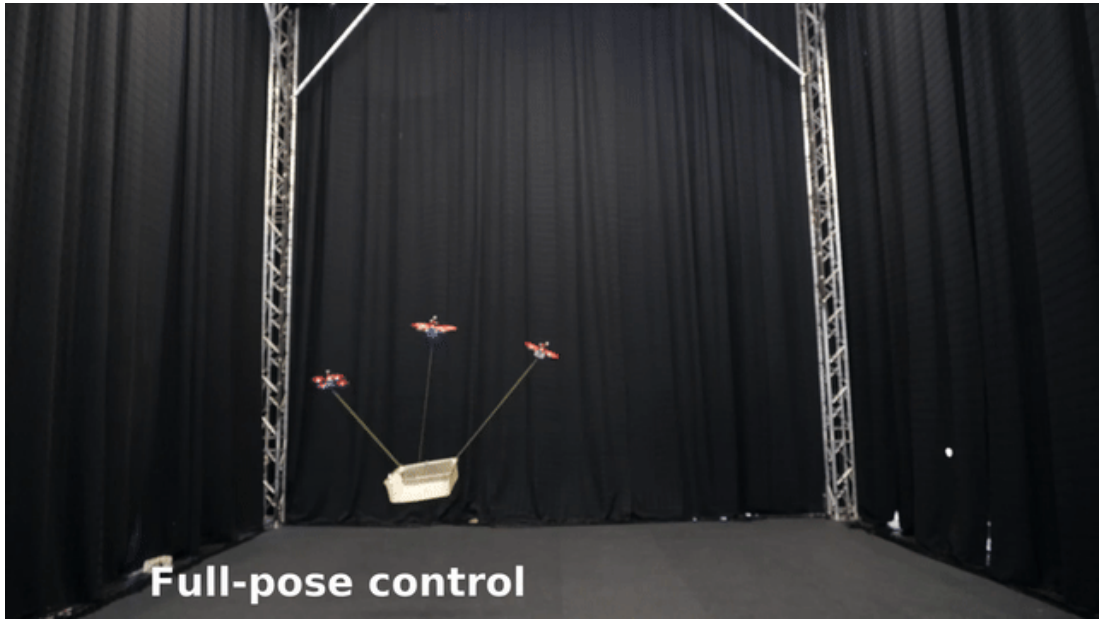
Closing the gap!



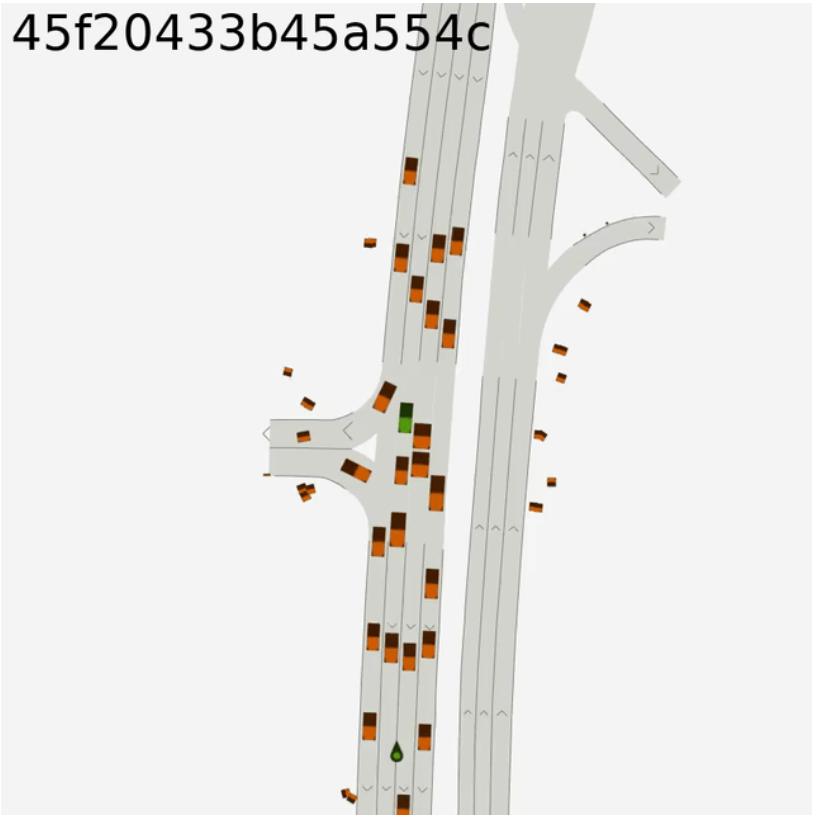
Building toward more unified algorithms!



Stable algorithms enable confidence: multi-agent planning through RL



Zeng, Jack, Andreu Matoses Gimenez, Eugene Vinitsky, Javier Alonso-Mora, and Sihao Sun. "Decentralized aerial manipulation of a cable-suspended load using multi-agent reinforcement learning." *CORL* (2025).



Cusumano-Towner, Marco, David Hafner, Alex Hertzberg, Brody Huval, Aleksei Petrenko, Eugene Vinitsky, Erik Wijmans et al. "Robust autonomy emerges from self-play." ICML (2025).

Better evaluation enables us to observe the strength of regularized policy gradient methods