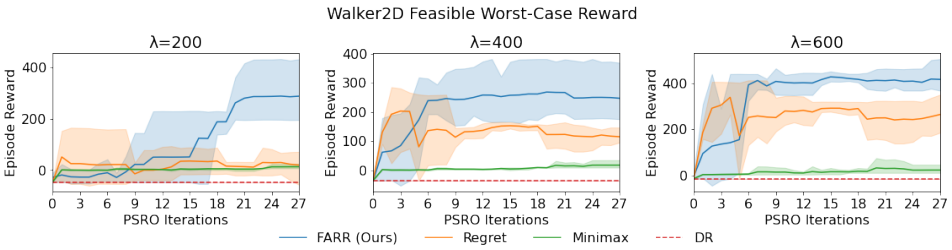
FEASIBLE ADVERSARIAL ROBUST REINFORCEMENT LEARNING FOR UNDERSPECIFIED ENVIRONMENTS SUPPLEMENTARY MATERIALS



(a)

Figure 1: Worst-case MuJoCo Walker2D reward among task parameterizations in the feasible set $\mathcal{F}^{\lambda}$ as a function of PSRO iterations for FARR and other baselines with multiple values of λ.

A MuJoCo Feasible Sets

For MuJoCo experiments, in order to measure each objective's worst-case average episode reward among feasible tasks, we evaluate on a discrete approximation of $\mathcal{F}^\lambda$. In these environments, Θ represents parameters of a beta distribution $\mathbf{B}(\alpha, \beta)$, $\alpha \in (0, 10], \beta \in (0, 10]$ sampled from each timestep to generate horizontal perturbing forces applied to the simulated robot. We consider a discretization of Θ with 11 different values in [0.01, 10] for both α and β. For each each combination $\theta = (\alpha, \beta)$, we train 7 seeds of a RL best-response $\mathbb{BR}(\theta)$ to completion using the same hyperparameters as the protagonist. The average final utility $U_p(\mathbb{BR}(\theta), \theta)$ across seeds is then used to calculate $\mathcal{F}^\lambda$ using equation (1). In Figure 2, for each environment, we show the 7-seed average $U_p(\mathbb{BR}(\theta), \theta)$ for every value of θ and the resulting feasible sets $\mathcal{F}^\lambda$ (shown in green) that we evaluate robustness to for each value of λ.

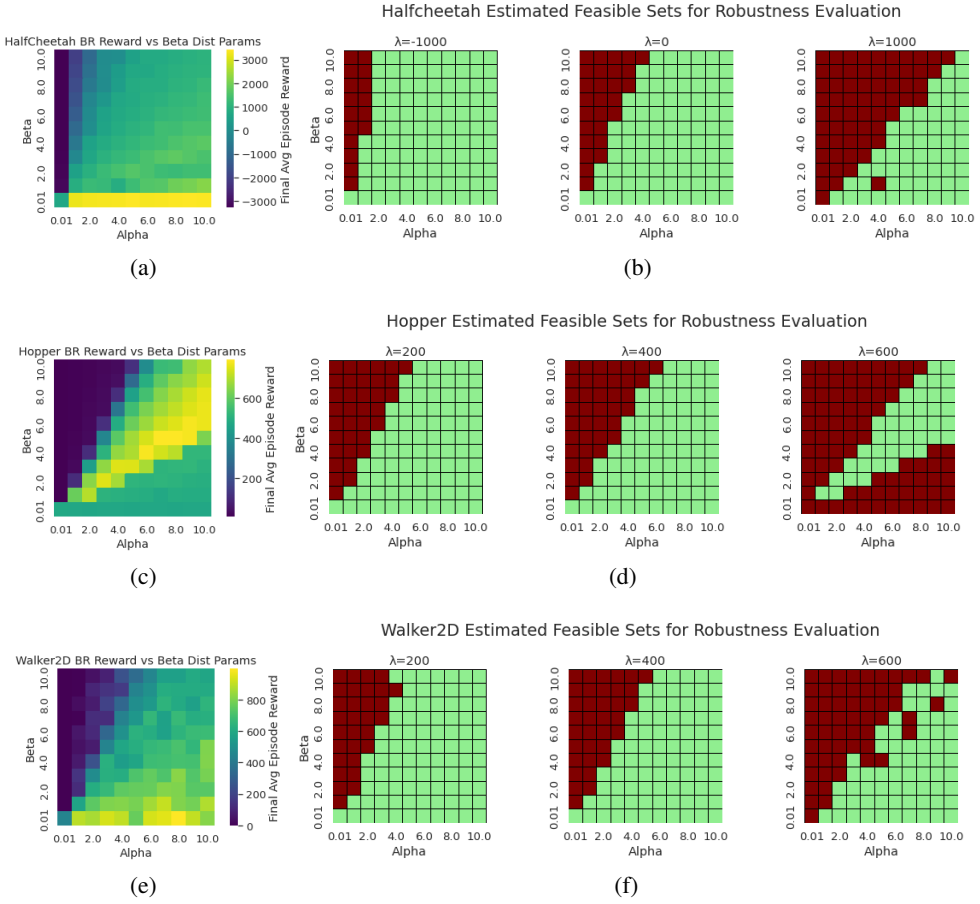


Figure 2: (a,c,e) Estimated values for $U_p(\mathbb{BR}(\theta), \theta)$ across the two parameters α and β that the adversary has control over. (b,d,f) The feasible sets $\mathcal{F}^\lambda$ used for evaluation marked in green for each value of λ.

B MuJoCo Learned Adversary Strategies

After running PSRO to completion, the output strategies are both a protagonist mixed strategy σ_p and an adversary mixed strategy σ_θ that should jointly approximate a Nash equilibrium to each of the two-player zero-sum game objectives we optimize. In figures 3, 4, and 5, we display the distribution of θ values induced by the final MuJoCo adversary mixed strategies σ_θ for the minimax, regret, and FARR objectives. For each λ value considered, we overlay in shades of green the number of $\mathbb{BR}(\theta)$ seeds used in measuring $\mathcal{F}^\lambda$ that achieved a final average episode reward greater than or equal for λ.

Across all environments, the minimax adversary consistently selects the most difficult θ values possible outside of any variations considered feasible with the λ values tested. In contrast, FARR mixes between θ values both well inside of the feasible regions and at the edges where task variations are as challenging as possible while remaining feasible.

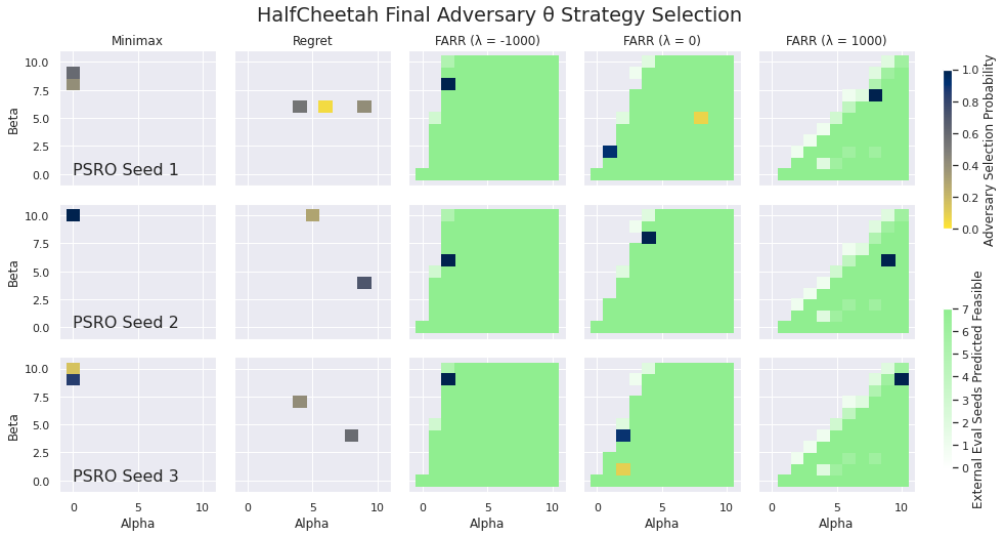


Figure 3: HalfCheetah θ distributions induced by the final adversary PSRO mixed strategy σ_θ for each objective.

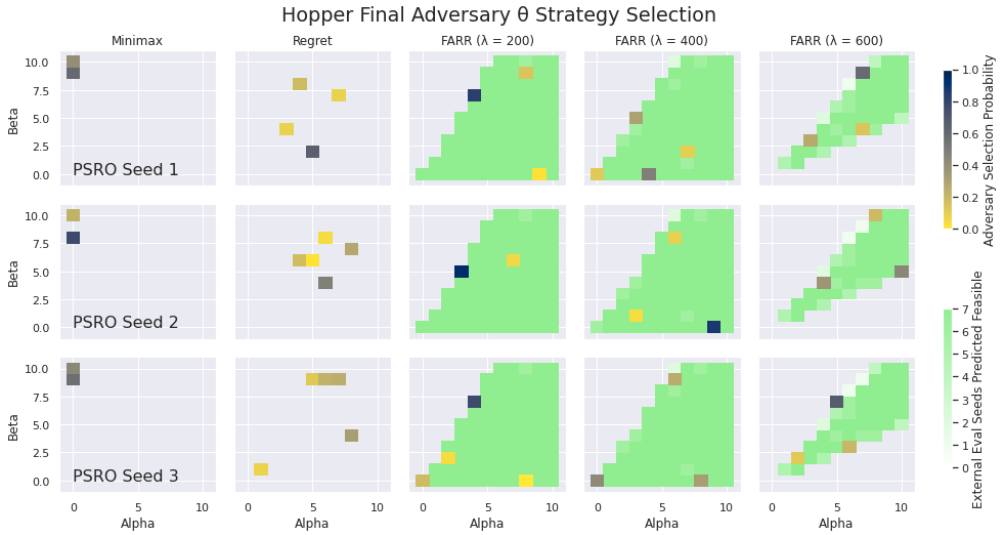


Figure 4: Hopper θ distributions induced by the final adversary PSRO mixed strategy σ_θ for each objective.

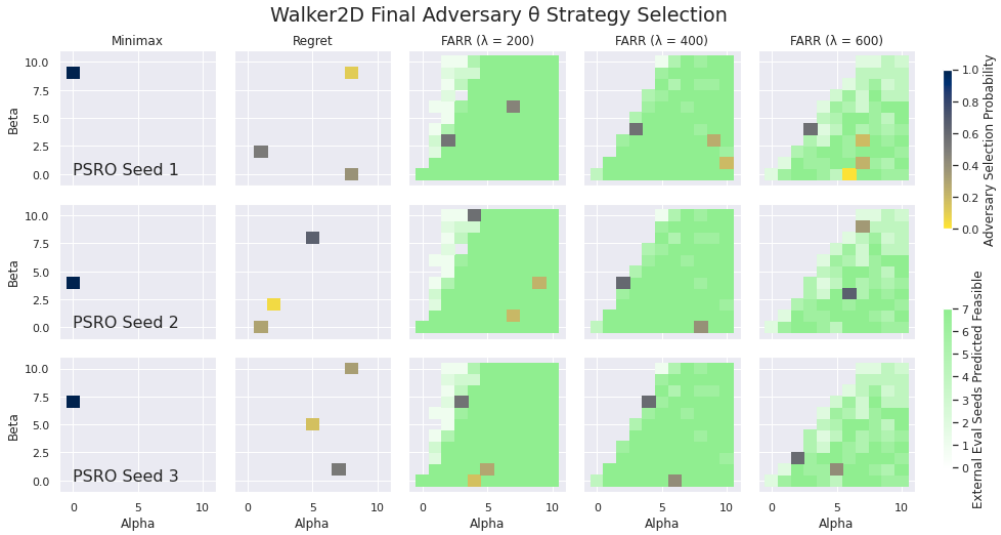


Figure 5: Walker2D θ distributions induced by the final adversary PSRO mixed strategy σ_θ for each objective.

C Properties of Nash Equilibria for the FARR Transformed Game

In this section we show that solving for Nash equilibria in the FARR transformed game will give the same result as solving for NE in a regular minimax robust RL game with the adversary strategy space already limited to only feasible strategies. The benefit of solving the FARR game is that the set of feasible adversary strategies does not need to be known a priori.

Define the set of λ-infeasible adversary strategies as $\mathcal{I}^\lambda = \Theta \setminus \mathcal{F}^\lambda$. Define the set of all possible protagonist strategies π_p as Π_p^*. For both protagonist utility functions U_p^λ and U_p, the adversary's utility function U_a^λ and U_a is the negative of the protagonist's, $U_a^\lambda(\pi_p, \theta) = -U_p^\lambda(\pi_p, \theta)$ and $U_a(\pi_p, \theta) = -U_p(\pi_p, \theta)$ for any π_p and θ.

Theorem 1. *For sufficiently large C, a Nash equilibrium joint strategy σ^*_{FARR} of a FARR transformed game G_{FARR} with utility function U_p^λ, protagonist strategies Π_p^* and all adversary strategies Θ is also a Nash equilibrium of a reduced game $G_{reduced}$ with utility function U_p, protagonist strategies Π_p^*, and only λ-feasible adversary strategies $\mathcal{F}^\lambda$.*

Proof. Let C take a sufficiently large value greater than any utility achievable by the protagonist $C > \max_{\pi_p,\theta} U_p(\pi_p, \theta)$. Since $U_a^\lambda(\pi_p, \theta') = -C$ when $\theta' \in \mathcal{I}^\lambda$, then for any $\theta' \in \mathcal{I}^\lambda$, any $\theta \in \mathcal{F}^\lambda$, and any $\pi_p \in \Pi_p^*$: $U_a^\lambda(\pi_p, \theta') < -U_p(\pi_p, \theta) = U_a^\lambda(\pi_p, \theta)$. It follows, as long as $\mathcal{F}^\lambda$ is nonempty, that for all $\theta' \in \mathcal{I}^\lambda$ there exists an adversary strategy $\theta \in \mathcal{F}^\lambda$ which achieves higher adversary utility against every $\pi_p \in \Pi_p^*$, thus all λ-infeasible adversary strategies $\theta' \in \mathcal{I}^\lambda$ are strictly dominated in the FARR transformed game.

If all $\theta' \in \mathcal{I}^\lambda$ are strictly dominated then they are not in the support of any Nash equilibrium for G_{FARR}. Furthermore, it is possible to remove strategies $\theta' \in \mathcal{I}^\lambda$ though iterated elimination of strictly dominated strategies (IESDS) to reduce G_{FARR} to G_{reduced} since the adversary strategy set for G_{reduced} is $\Theta \setminus \mathcal{I}^\lambda = \mathcal{F}^\lambda$ and for all $\theta \in \mathcal{F}^\lambda$ and $\pi_p \in \Pi_p^*$: $U_p^\lambda(\pi_p, \theta) = U_p(\pi_p, \theta)$. If G_{reduced} is an outcome of IESDS from G_{FARR}, then if σ^*_{FARR} is a NE of G_{FARR}, it is also an NE of G_{reduced}.

□

By employing a penalty C rather than directly pruning infeasible strategies from the PSRO restricted game, the FARR objective can be defined without consideration to the mechanics of any specific algorithm like PSRO. The FARR objective can potentially be optimized with two-player zero-sum game methods other than PSRO as well, though exploring the use of more optimization methods is left to future work.

D Selecting Values for λ

FARR is most applicable when the appropriate value for λ can be derived from problem requirements. For instance, λ would ideally be set to the lowest average return that an optimal agent would receive across task variations in deployment. This value could come from the environment definition, where doable tasks provide a minimum level of return if accomplished. Alternatively, λ could be chosen by anticipating the maximum difficulty of task variations that would be seen at test-time or on which robust performance is important to the practitioner.

In the case where an appropriate value of λ is completely unknown and cannot be derived from problem requirements, in a low-dimensional task variation space, manually tuning the adversary's capabilities without FARR may be appropriate. In a complex, high-dimensional task variation space, searching for a useful λ may be easier than a direct search over the space of adversary legal strategy sets because λ presents a single variable to tune, rather than a large number of legal parameter ranges or complex conditional constraints between adversary-specified parameters that may need to be defined. In this work, we consider the case where λ can be derived from problem requirements.

E PSRO Comparison with Self-Play

In Lava World, for each of the two-player zero-sum game objectives, we compare PSRO to self-play in which the protagonist, adversary, and evaluator π_e^θ continuously train together. For all self-play agents, we train with PPO to enable stochastic policies like PSRO is able to output. Regret self-play matches the original PAIRED algorithm from Dennis et al. (2020).

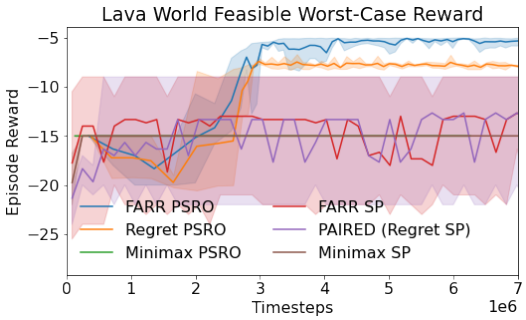


Figure 6: Worst-case average episode reward among goals in $\mathcal{F}^\lambda$ vs timesteps collected for each two-player zero-sum game objective optimized with both PSRO and PPO Self-Play.

Although self-play may potentially yield competitive performance in some scenarios, unlike PSRO, it lacks any guarantees of converging to an approximate Nash equilibrium in two-player zero-sum partially-observable Markov or extensive-form games. Seen in Figure 6, we see that self-play for both FARR and PAIRED fails to converge, reaching a maximum feasible-space worst-case average reward of -9 as agent policies cycle and learn to represent nearly deterministic strategies during most points in training. The NE for Lava World requires a mixed-strategy in which the adversary samples a high-entropy (non-uniform) distribution of hidden goals. The degenerate solution to Minimax is reached by both algorithms.

F Environment Details

F.1 Lava World

In the Lava World grid environment, the protagonist uses discrete actions to move in each of the 4 cardinal directions. For observations, the protagonist receives a one-hot encoding of its current location, and the protagonist does not observe the goal location. The adversary strategy space Θ is to define the hidden goal location and consists of every grid cell location in the environment's 5x5 grid with the exception of the protagonist's fixed starting location.

The protagonist receives a reward of -1 in each timestep that it does not reach the hidden goal location suggested by the adversary and -15 if it moves into a lava cell, even if the lava cell was a goal. An episode ends after either 20 timesteps elapse or the goal is reached.

F.2 MuJoCo Environments

The MuJoCo environments use the Mujoco physics engine (Todorov et al., 2012) and are modified versions of the perturbed robotic control environments originally presented in Pinto et al. (2017).

In each environment variation (HalfCheetah, Hopper, Walker2D), a max episode duration of 200 timesteps is imposed, and the proportion of time remaining in the range [0, 1] is appended to each task's original observation. We use the continuous action space variants of each task.

The adversary strategy space Θ consists of continuous α and β parameters in the range (0, 10] for a beta distribution $\mathbf{B}(\alpha, \beta)$ used to generate horizontal perturbing forces sampled and applied to the robot's torso every timestep. Each timestep, a new horizontal force $F \in [-F_{\text{max}}, F_{\text{max}}]$, $F = X(2F_{\text{max}}) - F_{\text{max}}$ is generated where $X \sim \mathbf{B}(\alpha, \beta)$ and $F_{\text{max}} = 100$.

When discretizing values of $\theta = (\alpha, \beta)$ for evaluation purposes to measure a policy's performance across values in Θ or $\mathcal{F}^{\lambda}$, we use $\alpha, \beta \in \{0.01, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0\}$.

G TRAINING DETAILS

Protagonist RL training details are provided below for each environment. Policies used to estimate $\mathbb{BR}(\theta)$ for a given θ use the same training procedure and parameters as the protagonist. Like Dennis et al. (2020), we train protagonist policies on the easier-to-learn unmodified environment reward U_p rather than our two-player game objective U_p^{λ} because the only component of the game utility that the protagonist can affect is U_p. A protagonist best-response that maximizes U_p also maximizes U_p^{λ}. Critically, we still calculate U_p^{λ} in the PSRO empirical payoff matrix U_{λ}^{Π} and use U_{λ}^{Π} to calculate the meta-game NE strategy $\sigma = (\sigma_p, \sigma_{\theta})$.

G.1 LAVA WORLD

We train Lava World protagonist RL policies using DDQN Van Hasselt et al. (2016). All Lava World protagonist RL policies are stopped training after either 150,000 timesteps are collected or once performance plateaus (average episode return doesn't improve by 0.5 over 20,000 timesteps and a minimum of 80,000 timesteps is collected). Lava World DDQN hyperparameters are presented below. Our RL code was built using the RLlib framework Liang et al. (2018), and any hyperparameters not specified are the version 1.0.1 defaults. We use an infeasibility penalty of $C = 50$.

algorithm	DDQN Van Hasselt et al. (2016)
circular replay buffer size	50,000
prioritized experience replay	No
total rollout experience gathered each iter	8 steps
learning rate	0.007
batch size	1024
optimizer	Adam (Kingma & Ba, 2014)
TD-error loss type	MSE
target network update frequency	every 4,000 steps
MLP layer sizes	[256, 256]
activation function	tanh
discount factor γ	1.0
exploration ϵ	Linearly annealed from 0.5 to 0.01 over 20,000 timesteps

Table 1: Lava World protagonist DDQN hyperparameters

algorithm	PPO Schulman et al. (2017)
GAE λ	0.9
entropy coeff	0.007
clip param	0.276
KL target	3e-4
KL coeff	0.0016
learning rate	5e-4
train batch size	8192
SGD minibatch size	64
num SGD epochs on each train batch	40
shared policy and value networks	No
value function clip param	10
MLP layer sizes	[256, 256]
activation function	Tanh
discount factor γ	1.0

Table 2: Lava World PPO self-play protagonist hyperparameters

algorithm	PPO Schulman et al. (2017)
GAE λ	0.95
entropy coeff	0.006
clip param	0.292
KL target	0.092
KL coeff	0.168
learning rate	3e-4
train batch size	8192
SGD minibatch size	64
num SGD epochs on each train batch	30
shared policy and value networks	No
value function clip param	100
MLP layer sizes	[256, 256]
activation function	Tanh
discount factor γ	1.0

Table 3: Lava World PPO self-play adversary hyperparameters

In PSRO, to calculate the payoff matrix U_λ^Π, we estimate $U_p(\pi_p, \theta)$ for each pairing of player policies $\pi_p \in \Pi_p$ and $\theta \in \Pi_\theta$ using a single rollout because both Lava World environment dynamics and evaluation DDQN policies are deterministic. The normal-form meta-game Nash Equilibrium over U_λ^Π is calculated using 2000 iterations of Fictitious Play (Brown, 1951). Calculating the meta-game NE typically takes a second or less of wall-time compute.

During PSRO evaluation, we measure the performance of the protagonist meta-game mixed strategy σ_p, in which a new protagonist policy $\pi_p \sim \sigma_p$ is sampled at the begining of each episode.

In self-play, the adversary is trained as a single-step agent via PPO. For simplicity, we keep the same network architecture for all self-play agents, and the adversary observes a constant vector of zeros.

G.2 MuJoCo

We train MuJoCo environment protagonist RL policies using PPO (Schulman et al., 2017). Each PPO model consists of an MLP followed by an LSTM with shared weights between the policy and value function, branching into final output layers after the LSTM. PPO hyperparameters for each MuJoCo environment are presented below. Any hyperparameters not specified are the RLlib version 1.0.1 defaults. We use an infeasibility penalty of $C = 1e6$.

algorithm	PPO Schulman et al. (2017)
GAE λ	0.9
entropy coeff	0.01
clip param	0.001
KL target	0.004
KL coeff	0.522
learning rate	5e-4
train batch size	4096
SGD minibatch size	64
num SGD epochs on each train batch	5
shared policy and value networks	Yes
value function loss coeff	0.001
value function clip param	100
continuous action range	[-1.0, 1.0] for each dim
MLP layer sizes	[32]
activation function	Tanh
LSTM cell size	32
LSTM max sequence length	20
discount factor γ	0.99
RL policy training stopping condition	7e6 timesteps

Table 4: HalfCheetah PPO hyperparameters

algorithm	PPO Schulman et al. (2017)
GAE λ	0.9
entropy coeff	0.001
clip param	0.002
KL target	0.036
KL coeff	0.013
learning rate	7e-4
train batch size	4096
SGD minibatch size	32
num SGD epochs on each train batch	5
shared policy and value networks	Yes
value function loss coeff	0.001
value function clip param	10
continuous action range	[-1.0, 1.0] for each dim
MLP layer sizes	[64, 64]
activation function	Tanh
LSTM cell size	32
LSTM max sequence length	20
discount factor γ	0.99
RL policy training stopping condition	6e6 timesteps

Table 5: Hopper PPO hyperparameters

algorithm	PPO Schulman et al. (2017)
GAE λ	0.95
entropy coeff	0.0
clip param	0.014
KL target	0.005
KL coeff	0.007
learning rate	9e-4
train batch size	1024
SGD minibatch size	64
num SGD epochs on each train batch	10
shared policy and value networks	Yes
value function loss coeff	1e-4
value function clip param	1000
continuous action range	[-1.0, 1.0] for each dim
MLP layer sizes	[32]
activation function	Tanh
LSTM cell size	32
LSTM max sequence length	20
discount factor γ	1.0
RL policy training stopping condition	6e6 timesteps

Table 6: Walker2D PPO hyperparameters

In PSRO, to calculate the payoff matrix U_λ^Π, we estimate $U_p(\pi_p, \theta)$ for each pairing of player policies $\pi_p \in \Pi_p$ and $\theta \in \Pi_\theta$ using 100 rollouts as perturbed MuJoCo environment transition dynamics are stochastic. The normal-form meta-game Nash Equilibrium over U_λ^Π is also calculated using 2000 iterations of Fictitious Play.

Although PSRO convergence guarantees are not provided for continuous-action environments, in McAleer et al. (2021), McAleer et al. (2022), and our own experiments, PSRO reliably produces meta-game NE mixed strategies that are empirically difficult for an opponent to exploit.

H Computational Costs

Experiments were performed on a local computer with 128 logical CPU-cores, 4 RTX 3090 GPUs, and 512GB of RAM. Due to small network sizes and comparably high overhead of CPU-based environments, logging, and other tasks, most experiments were performed without GPU acceleration. All individual training runs for a given player against a fixed opponent took 5 CPU-cores each. Lava world experiments individually ran for roughly 8 to 24 hours each, while MuJoCo experiments individually ran for roughly 72 hours each.

I Code

A GitHub link for our experiment code will be provided under the MIT license in an updated version of this work.

Our code is written on top of the RLlib framework (Liang et al., 2018) and uses environments built using the MuJoCo physics engine (Todorov et al., 2012), both of which are open-source and available under the Apache-2.0 Licence.

References

George W. Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, pp. 374–376, 1951.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *arXiv preprint arXiv:2012.02096*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pp. 3053–3062. PMLR, 2018.

Stephen McAleer, John Banister Lanier, Kevin A Wang, Pierre Baldi, and Roy Fox. Xdo: A double oracle algorithm for extensive-form games. *Advances in Neural Information Processing Systems*, 34:23128–23139, 2021.

Stephen McAleer, Kevin Wang, Marc Lanctot, John Lanier, Pierre Baldi, and Roy Fox. Anytime optimal psro for two-player zero-sum games. *arXiv preprint arXiv:2201.07700*, 2022.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pp. 2817–2826. PMLR, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI conference on artificial intelligence*, volume 30, 2016.