# NEUPL: NEURAL POPULATION LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Learning in strategy games (e.g. StarCraft, poker) requires the discovery of diverse policies. This is often achieved by training new policies against existing ones, growing a population of policies that collectively become robust to exploit. Unfortunately, this approach suffers from two fundamental issues in *real-world* games: a) under finite compute budget, approximate best-response procedure often needs truncating, resulting in *good*-responses populating the policy population; b) in skill-based games, *tabula rasa* learning of best-responses is wasteful and quickly become computationally intractable, facing increasingly skillful opponents. In this work, we propose Neural Population Learning (NeuPL) as a solution to both issues. NeuPL offers convergence guarantees to a population of *best*-responses under mild conditions. By representing an entire population of policies within a single conditional model, NeuPL enables skill transfer across policies by construction. Empirically, we show the generality, improved performance and efficiency of NeuPL across several test domains[1]. Most interestingly, we show that novel strategies become more accessible, not less, as the neural population expands.

The need for learning not one, but a population of strategies is rooted in classical game theory. Consider the purely cyclical game of *rock-paper-scissors*, the performance of individual strategies is meaningless as improving against one entails losing to another. By contrast, performance can be meaningfully examined *between* populations. For instance, a population consisting of pure strategies $\{rock, paper\}$ is said to do well against a singleton population of $\{scissors\}$ because in the meta-game where both populations are known, a player picking strategies from the former can always beat a player choosing from the latter[2]. This observation underpins recent successes combining population learning and deep reinforcement learning (deep RL) in complex multi-agent domains (Jaderberg et al., 2019; Liu et al., 2021). Most impressively, Vinyals et al. (2019) explored the strategy game of StarCraft where exploiters to prioritized mixtures of opponents are introduced to a growing league of policies. The league counted close to a thousand sophisticated deep RL agents as the population collectively became robust to exploits.

Unfortunately, such empirical successes often come at considerable costs. Population learning algorithms with theoretical guarantees are often studied in normal-form games (Brown, 1951; McMahan et al., 2003) where best-responses can be solved exactly and efficiently. This is, however, in stark contrast to real-world games (Czarnecki et al., 2020) — such games are often temporal in nature, yielding exponentially large policy spaces where best-responses can only be approximated with computationally intensive methods (e.g. deep RL). This has two implications. First, one cannot efficiently tell apart *good*-responses from best-responses. As a result, approximate best-response procedures are often truncated prematurely, according to hand-crafted schedules (Lanctot et al., 2017). Second, real-world games often afford transitive skill dimensions where the most skilled policy prevails. In this setting, *tabula rasa* learning in the presence of skillful opponents quickly becomes wasteful and challenging, rendering population learning intractable beyond a few iterations.

This approach is in stark contrast to highly efficient and versatile human learning. Mastering diverse policies in a game often facilitates incremental strategic innovation and learning about new strategies does not stop us from revisiting and improving upon old ones (Caruana, 1997; Krakauer et al., 2006). In this work, we make progress towards endowing artificial agents with similar capacity by extending efficient population learning to real-world games. Specifically, we propose NeuPL, an efficient population learning framework that learns and represents diverse policies in symmetric zero-sum

[1] See `https://neupl.github.io/demo/` for supplementary illustrations.

[2] This is formally quantified by Relative Population Performance, see Definition A.1 (Balduzzi et al., 2019).

games within a single conditional network via self-play (**Section 1.2**). Theoretically, we show that NeuPL converges to a sequence of N-step best-responses under certain conditions (**Section 1.4**). Empirically, we illustrate the generality of NeuPL by replicating known effects of population learning algorithms on the classical domain of *rock-paper-scissors* as well as its partially-observed, spatiotemporal counterpart *running-with-scissors* (Vezhnevets et al., 2020) (**Section 2.1**). Most interestingly, we show that NeuPL by construction offers skill transfers across policies, discovering best responses to strong opponents that would have been inaccessible to comparable baselines (**Section 2.2**). Finally, we show the appeal of NeuPL in the skill-based challenge domain of MuJoCo Football (Liu et al., 2019) where players must integrate low-level actions in order to coordinate as a team. In this transitive game-of-skills, NeuPL succinctly represents a short sequence of best-responses, as opposed to an ever-growing population of *good*-responses if population learning were applied naively (**Section 2.4**).

# 1 METHODS

Our method is designed with two desiderata in mind. First, at convergence, the resulting population of policies should represent a sequence of iterative N-step best-responses under reasonable conditions. Second, skill transfer can occur across policies throughout training. In this section, we first define the problem setting of interests as well as necessary terminologies. We then describe NeuPL, our main conceptual algorithm as well as its theoretical properties. To make it concrete, we further consider *deep RL* specifically and offer two practical implementations of NeuPL in real-world games.

## 1.1 PRELIMINARIES

**Approximate Best-Response (ABR) in Stochastic Games** We consider a symmetric zero-sum Stochastic Game (Shapley, 1953) defined by $(\mathcal{S}, \mathcal{O}, \mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, p_0)$ with $\mathcal{S}$ the state space, $\mathcal{O}$ the observation space and $\mathcal{X} : \mathcal{S} \rightarrow \mathcal{O} \times \mathcal{O}$ the observation function defining the (partial) views of the state for both players. Given joint actions $(a_t, a'_t) \in \mathcal{A} \times \mathcal{A}$, the state follows the transition distribution $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{A} \rightarrow \Pr(\mathcal{S})$. The reward function $\mathcal{R} : \mathcal{S} \rightarrow \mathbb{R} \times \mathbb{R}$ defines the rewards for both players in state $s_t$, denoted $\mathcal{R}(s_t) = (r_t, -r_t)$. The initial state of the environment follows the distribution $p_0$. In a given state $s_t$, players act according to policies $(\pi(\cdot|o_{\leq t}), \pi'(\cdot|o'_{\leq t}))$. Player $\pi$ achieves an expected return of $J(\pi, \pi') = \mathbb{E}_{\pi,\pi'}[\sum_t r_t]$ against $\pi'$. Policy $\pi^*$ is a *best response* to $\pi'$ if $\forall \pi, J(\pi^*, \pi') \geq J(\pi, \pi')$. We define $\hat{\pi} \leftarrow \text{ABR}(\pi, \pi')$ with $J(\hat{\pi}, \pi') \geq J(\pi, \pi')$. In other words, an ABR procedure yields a policy $\hat{\pi}$ that does no worse than $\pi$, in the presence of an opponent $\pi'$.

**Meta-strategies in Population Learning** Given a symmetric zero-sum game and a set of $N$ policies $\Pi := \{\pi_i\}_{i=1}^N$, we define a normal-form meta-game where players' $i$-th action corresponds to executing policy $\pi_i$ for one episode. A meta-strategy $\sigma$ thus defines a probability assignment, or an action profile, over $\Pi$. Within $\Pi$, we define $\mathcal{U} \in \mathbb{R}^{N \times N} \leftarrow \text{EVAL}(\Pi)$ to be the expected payoffs between pure strategies of this meta-game or equivalently, $\mathcal{U}_{ij} := J(\pi_i, \pi_j)$ in the underlying game. We further extend the ABR procedure of the underlying game to mixture policies represented by $\sigma$, such that $\hat{\pi} \leftarrow \text{ABR}(\pi, \sigma, \Pi)$ with $\mathbb{E}_{\pi' \sim P(\sigma)}[J(\hat{\pi}, \pi')] \geq \mathbb{E}_{\pi' \sim P(\sigma)}[J(\pi, \pi')]$. Finally, we define $f : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^N$ to be a meta-game solver with $\sigma \leftarrow f(\mathcal{U})$ and $\mathcal{F} : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}^{N \times N}$ a meta-graph solver with $\Sigma \leftarrow \mathcal{F}(\mathcal{U})$. The former formulation is designed for iterative optimization of approximate best-responses as in Lanctot et al. (2017) whereas the latter is motivated by concurrent optimization over a set of population-level objectives as in Garnelo et al. (2021). In particular, $\Sigma \in \mathbb{R}^{N \times N} := \{\sigma_i\}_{i=1}^N$ defines $N$ population-level objectives, with $\pi_i$ optimized against the mixture policy represented by $\sigma_i$ and $\Pi$. As such, $\Sigma \in \mathbb{R}^{N \times N}$ corresponds to an adjacency matrix of an interaction graph. Appendix A.1 illustrates several structured population learning objectives defined by $\Sigma$ and equivalently, by their interaction graphs.

## 1.2 NEURAL POPULATION LEARNING

We now present NeuPL and contrast it with Policy-Space Response Oracles (PSRO, Lanctot et al. (2017)) [3] which is similarly focused on population learning with approximate best-responses by RL. It's worth noting that NeuPL implements a more general class of population learning algorithms than

[3] We present PSRO specialised to N-step approximate best-responses instead of the generalised formulation.

PSRO. In particular, NeuPL supports static meta-graph solvers $\mathcal{F}$ with a constant number of policies (e.g. Section 2.1).

**Algorithm 1** Neural Population Learning (Ours)

$\mathcal{F} : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ ▷ Meta-graph solver.
$\Pi_\theta(\cdot|s,\sigma)$ ▷ Conditional neural population net.
$\Sigma := \{\sigma_i\}_{i=1}^N$ ▷ Initial interaction graph.
**while** true **do**
  $\Pi_\theta^\Sigma \leftarrow \{\Pi_\theta(\cdot|s,\sigma_i)\}_{i=1}^N$ ▷ Neural population.
  **for** $\sigma_i \in \text{UNIQUE}(\Sigma)$ **do**
    $\Pi_\theta^{\sigma_i} \leftarrow \Pi_\theta(\cdot|s,\sigma_i)$
    $\Pi_\theta^{\sigma_i} \leftarrow \text{ABR}(\Pi_\theta^{\sigma_i}, \sigma_i, \Pi_\theta^\Sigma)$ ▷ Self-play.
  $\mathcal{U} \leftarrow \text{EVAL}(\Pi_\theta^\Sigma)$ ▷ (Optional) if $\mathcal{F}$ adaptive.
  $\Sigma \leftarrow \mathcal{F}(\mathcal{U})$ ▷ (Optional) if $\mathcal{F}$ adaptive.
**return** $\Pi_\theta, \Sigma$

**Algorithm 2** PSRO (Lanctot et al., 2017)

$f : \mathbb{R}^{M\times M} \to \mathbb{R}^M$ ▷ Meta-game solver.
$\Pi := \{\pi_0\}$ ▷ Initial policy population.
$\sigma \leftarrow \text{UNIF}(\Pi)$ ▷ Initial meta-strategy.

**for** $i \in [[N]]$ **do** ▷ N-step ABR.
  Initialize $\pi_{\theta_i}$.
  $\pi_{\theta_i} \leftarrow \text{ABR}(\pi_{\theta_i}, \sigma, \Pi)$
  $\Pi \leftarrow \Pi \cup \{\pi_{\theta_i}\}$
  $\mathcal{U} \leftarrow \text{EVAL}(\Pi)$ ▷ Exp. payoffs.
  $\sigma \leftarrow f(\mathcal{U})$
**return** $\Pi$

Comparing the two algorithms, NeuPL deviates from PSRO in two important ways. First, NeuPL suggests concurrent and continued training of all *unique* policies such that no good-response features in the population prematurely due to early truncation. Second, NeuPL represents an entire population of policies via a shared conditional network $\Pi_\theta(\cdot|s,\sigma)$ with each policy $\Pi_\theta(\cdot|s,\sigma_i)$ conditioned on and optimised against its opponent's meta-game strategy $\sigma_i$, enabling skill-transfers across policies. This representation also makes NeuPL general, as it delegates the choice of *effective* population sizes $|\text{UNIQUE}(\Sigma)| \leq |\Sigma| = N$ to the meta-graph solver $\mathcal{F}$ as $\sigma_i = \sigma_j \implies \Pi_\theta(\cdot|s,\sigma_i) \equiv \Pi_\theta(\cdot|s,\sigma_j)$ (cf. examples in Section 2.1).

**N-step Best-Responses via Lower-Triangular Interaction Graphs** A popular class of population learning algorithms aims for convergence to a sequence of iterative best-responses where each policy $\pi_i$ seeks to best-respond to an opponent meta-game strategy $\sigma_i$ with support over a subset of the policy population $\Pi_{<i} = \{\pi_j\}_{j<i}$. In NeuPL, this class of algorithms are implemented with meta-graph solvers that return *lower-triangular* adjacency matrices $\Sigma$ with $\Sigma_{i\leq j} = 0$. Under this constraint, $\sigma_0$ becomes a zero vector, implying that $\Pi_\theta(\cdot|s,\sigma_0)$ does not seek to best-respond to any policies. Similar to the role of initial policies $\{\pi_0\}$ in PSRO (Algorithm 2), $\Pi_\theta(\cdot|s,\sigma_0)$ serves as the *sink* policy for the sequence of N-step best-responses. In practice, any fixed policy can be used and we typically use a random or zero policy depending on the underlying game.

One prominent example is PSRO-NASH, where $\pi_i$ is optimized to best-respond to the Nash mixture policy over $\Pi_{<i}$. This particular meta-game solver is implemented in Algorithm 3.

**Algorithm 3** A meta-graph solver implementing PSRO-NASH.

**function** $\mathcal{F}_{\text{PSRO-N}}(\mathcal{U})$ ▷ $\mathcal{U} \in \mathbb{R}^{N\times N}$ the empirical payoff matrix.
  Initialize meta-game strategies $\Sigma \in \mathbb{R}^{N\times N}$ with zeros.
  **for** $i \in \{1,\ldots,N-1\}$ **do**
    $\Sigma_{i+1,1:i} \leftarrow \text{SOLVE-NASH}(\mathcal{U}_{1:i,1:i})$ ▷ LP Nash solver, see Shoham & Leyton-Brown (2008).
  **return** $\Sigma$

### 1.3 NEURAL POPULATION LEARNING BY RL

We now define the discounted return maximized by $\Pi_\theta(\cdot|o_{\leq t},\sigma_i)$ in Equation 1. We denote $P(\sigma_i)$ as the probability distribution over policy $i$'s opponent identities $\sigma_j \in \{\sigma_1,\ldots,\sigma_N\}$. Intuitively, each policy is maximizing its expected returns in the underlying game under a double expectation: the first is taken over its opponent distribution, with $\sigma_j \sim P(\sigma_i)$ and the second taken under the game dynamics partly defined by the pair of policies $(\Pi_\theta(\cdot|o_{\leq t},\sigma_i), \Pi_\theta(\cdot|o'_{\leq t},\sigma_j))$.

$$J_{\sigma_i} = \mathop{\mathbb{E}}_{\sigma_j\sim P(\sigma_i)} \Big[ \mathop{\mathbb{E}}_{a\sim\Pi_\theta(\cdot|o_{\leq t},\sigma_i), a'\sim\Pi_\theta(\cdot|o'_{\leq t},\sigma_j)} \big[ \sum_t r_t\gamma^t \big] \Big] \tag{1}$$

To optimize $\Pi_\theta^\Sigma$ by RL, we jointly train an opponent-conditioned action-value[4] function, approximating the expected return of choosing an action $a_t$ given an observation history $o_{\leq t}$, following $\Pi_\theta(\cdot|o_{\leq t}, \sigma_i)$ thereafter in the presence of $\Pi_\theta(\cdot|o'_{\leq t}, \sigma_j)$, denoted by $Q_(o_{\leq t}, a_t, \sigma_i, \sigma_j) = \mathbb{E}_{\sigma_i,\sigma_j}[\sum_{\tau=t}^{t+T} \gamma^{\tau-t} r_\tau | o_{\leq t}, a_t]$ with $\gamma$ the discount factor. In the case of ABR by deep RL, we could additionally approximate the expected payoffs matrix $\mathcal{U}$ by learning a payoff estimator $\phi_\omega(\sigma_i, \sigma_j)$ minimizing the loss $\mathcal{L}_{ij} = \mathbb{E}_{o\sim\mathcal{D}}\left[(\phi_\omega(\sigma_i,\sigma_j) - \mathbb{E}_{a\sim\Pi_\theta(\cdot|o,\sigma_i)}[Q_\theta(o,a,\sigma_i,\sigma_j)])^2\right]$ where the expectation is taken over the state visitation distribution $\mathcal{D}$ defined by the pair of policies and the environment dynamics $\mathcal{P}$. In other words, $\phi_\omega(\sigma_i, \sigma_j)$ approximates the expected return of $\Pi_\theta(\cdot|o_{\leq}, \sigma_i)$ playing against $\Pi_\theta(\cdot|o'_{\leq}, \sigma_j)$. By connecting payoff matrix $\mathcal{U}$ to the learned $Q$ function, we can evaluate $\mathcal{U}$ efficiently, without explicitly evaluating all policies at each iteration.

Finally, we propose Algorithm 4 in the setting where the meta-graph solver $\Sigma^{\text{const}} \leftarrow \mathcal{F}(\mathcal{U}), \forall \mathcal{U}$ is a *constant* function and extends it to Algorithm 5 where the meta-graph solver is adaptive in $\mathcal{U}$. For instance, population learning algorithms such as Fictitious Play (Brown, 1951) implement static interaction graphs while algorithms such as PSRO-NASH (Lanctot et al., 2017) rely on adaptive meta-graph solvers.

### 1.4 CONVERGENCE TO N-STEP BEST-RESPONSES VIA NEUPL

Under certain assumptions on the best respond operator, interaction graph, and meta-graph solver we can construct proofs that NeuPL converges to an $N$-step best response. We introduce the term *grounded* (Section C) to refer to interaction graphs and meta-graph solvers that have a structure that imposes convergence to a unique set of policies. Certain interaction graphs are grounded, in particular, lower-triangular graphs are one such class which describe an $N$-step best response. In addition, certain meta-graph solvers are grounded, in particular, ones that operate on the sub-payoff and output a lower-triangular interaction graph, $\mathcal{F} : \mathcal{U}_{<i,<i} \to \Sigma_{i,<i}$. The lower-triangular maximum entropy Nash equilibrium (MENE) is one such grounded meta-graph solver. Therefore with sufficiently large $N$ NeuPL will converge to a normal-form Nash equilibrium. See Section C for the full definitions, theorems and proofs.

## 2 EXPERIMENTS

In this section, we validate different contributions of NeuPL across several domains of varying nature. First, we verify the generality of NeuPL from two aspects: **a)** NeuPL recovers expected results of existing population learning algorithms (Brown, 1951; Heinrich et al., 2015; Lanctot et al., 2017) on the classical game of *rock-paper-scissors* where we can visualize the learned policy population over time and; **b)** NeuPL generalises to the spatiotemporal, partially observed strategy game of *running-with-scissors* (Vezhnevets et al., 2020), where players must infer opponent behaviours through tactical and skillful interactions. Second, we show that NeuPL induces skill transfer across policies, enabling the discovery of exploiter policies to strong opponents that would have been out-of-reach otherwise. This property translates to improved efficiency and performance compared to PSRO-NASH baselines, even under favorable conditions. Lastly, we show that NeuPL scales to the large-scale game-of-skill of MuJoCo Football (Liu et al., 2019) where a concise sequence of best-responses are learned.

In all experiments, we use Maximum A Posterior Optimization (MPO, Abdolmaleki et al. (2018)) as the underlying RL algorithm, though any alternative can be used instead. Further details of our experimental setup are described in Appendix B.2.

### 2.1 IS NEURAL POPULATION LEARNING GENERAL?

**Static Interaction Graphs** Figure 1 illustrates the effect of NEUPL-RL-STATIC implementing three population learning algorithms in the purely cyclical domain of *rock-paper-scissors*. Figure 1 (Left) shows that learning from self-play leads to the policy transitioning through the strategy space indefinitely. This is a well-known result: an agent learning by self-play fails to converge in a purely cyclical game (Balduzzi et al., 2019). By contrast, Figure 1 (Middle) shows the effect of a specialized interaction graph. This graph explicitly encourages the discovery of one strategic cycle (formed by the first three nodes), as well as a final node that seeks to beat the others equally. As a result,

[4]We present the case for an action-value function but a value function could be used instead.

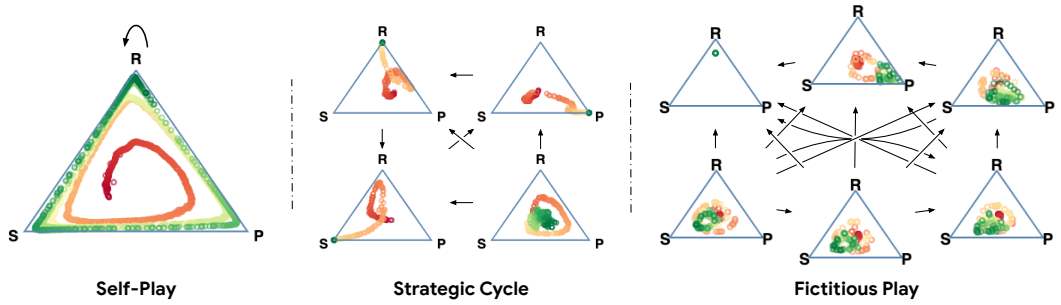


Figure 1: Neural Population Learning in *rock-paper-scissors* induced by static interaction graphs. Policy distributions are colored by training iteration from red (earliest) to green (latest). **(Left)** learning by self-play. **(Middle)** a neural population of 4 strategies exploring a strategic cycle. **(Right)** a neural population of 6 strategies iteratively best respond to the average previous strategies.

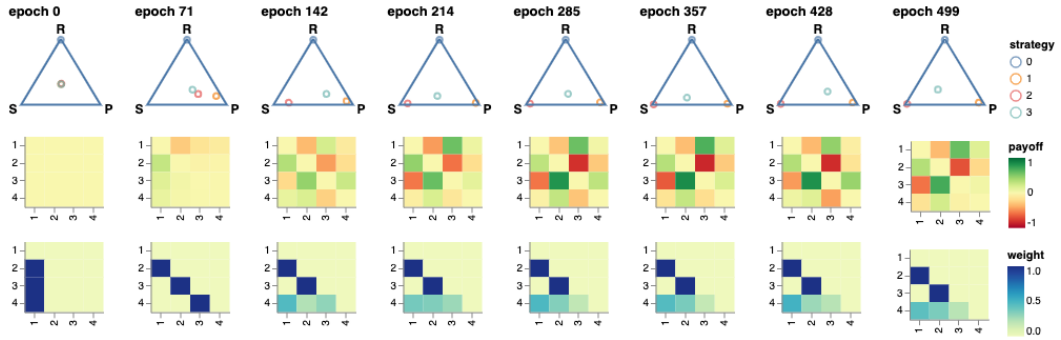


Figure 2: NeuPL in *rock-paper-scissors* induced by an adaptive interaction graph implementing PSRO-NASH. An epoch lasts 10 iterations. **(Top)** strategy space explored by the neural population of strategies over time. **(Middle)** the learned payoff estimates between the population of strategies. **(Bottom)** Adjacency matrices representing interaction graphs as responses to payoff matrices.

we obtain a population of strategies implementing the pure strategies of the game as well as a mixture strategy[5]. The design of such specific interaction graphs is not always possible as it requires domain knowledge and is only feasible in the simplest games. Figure 1 (Right) illustrates the general population learning algorithm of Fictitious Play (Brown, 1951; Heinrich et al., 2015) where players at each iteration optimize against their "average" previous players. As the sink policy of the N-step best-response sequence, the first strategy could execute any policy and we initialize it to an exploitable policy heavily biased towards playing rock[6]. We observe that the resulting population, captured by $\Pi_\theta(a|o_{\leq}, \sigma)$, learns to execute 6 different strategies, with the first being "rock", followed by a second learning to play pure "paper" to exploit the first and subsequent strategies gravitating towards the Nash equilibrium that cannot be exploited in this game.

**Adaptive Interaction Graphs** Figure 2 explores the effect of NEUPL-RL-ADAPTIVE implementing $\mathcal{F}_{\text{PSRO-N}}$ in *rock-paper-scissors*. We make the following observations. First, the first three strategies form a strategic cycle and converge to the pure strategies, while the last strategy optimizes against a mixture of the pure strategies, as expected of PSRO-NASH. We note that as in Figure 1 (Strategic Cycle), the last strategy needs *not* converge to a Nash strategy. Second, the number of distinct strategies represented by the neural population grows dynamically (comparing $\Sigma$ at epoch 0 and 71) — a distinct training objective for strategy $i+1$ is introduced if and only if $\Pi_\theta(\cdot|o_{\leq}, \sigma_i)$ successfully beats the Nash mixture strategy over $\Pi_\theta^{\Sigma<i}$. Unlike Lanctot et al. (2017), the effective size of the population is driven by the underlying meta-game solver, rather than a handcrafted schedule.

[5]Note that it needs not be the Nash strategy of the game as any strategy would achieve a return of zero.

[6]This choice is important: had the sink strategy been initialized to be the Nash mixture strategy, subsequent strategies would be "uninteresting" as no strategy can improve upon the sink strategy.

This is particularly appealing in real-world games, where we cannot efficiently determine if a policy has converged to a best-response. The size of the neural population limits the *maximum* number of strategies represented and optimized concurrently — a size of $N$ does not imply that the population would optimize $N$ distinct objectives at the start, nor does it enforce the number of distinct policies to reach $N$ at convergence. The number of distinct strategies represented by a neural population is primarily governed by the dynamics of the underlying game, the generalisation capacity of the neural network, the effectiveness of the RL algorithm and the meta-game solver $\mathcal{F}$. We later recall this property in *running-with-scissors* and in MuJoCo Football, where the underlying game affords varying degrees of strategic complexities.

**Stochastic Games** The rich strategic domain of *running-with-scissors* (Vezhnevets et al., 2020) extends the game of *rock-paper-scissors* to the spatiotemporal and partially observable setting. Using first-person partial observations of the game (a small window in front of the agent), each player collects resources representing "rock", "paper" and "scissors" so as to counter its opponent's unobserved inventory. At the end of an episode or when players confront each other through "tagging", players compare their inventories and receive rewards according to the *rock-paper-scissors* payoff table. To do well, an agent must learn to infer its opponent's behaviours from partial observation history $o_{\leq t}$. For instance, if "rock"s are missing, then the opponent may be collecting them; if the opponent is seen running past a "scissors", then it may not be interested in "scissors". We describe the environment in details in Appendix B.1. We show in Figure 3 that NeuPL with an adaptive meta-graph solver $\mathcal{F}_{\text{PSRO-N}}$ results in a population of sophisticated policies. As before, we set the first policy (sink policy) to be heavily exploitable and biased towards picking up "rock"s. Early in training, we note that the first three policies of the neural population implement the pure-resource policies of "rock", "paper" and "scissors" respectively, as evidenced by their relative payoffs. In contrast to *rock-paper-scissors*, the mixture of pure-resource policies is exploitable in the sequential setting, where the player can observe its opponent before implementing a counter strategy. Indeed, policy $\Pi_\theta(\cdot|o_{\leq}, \sigma_4)$ employs the tactic of "observe-and-exploit", successfully beating the mixture policies at epoch 680. Following $\mathcal{F}_{\text{PSRO-N}}$, $\Pi_\theta(\cdot|o_{\leq}, \sigma_5)$ updates its objective to focus solely on this newfound Nash equilibrium over $\Pi_\theta^{\Sigma<5}$, eventually developing a deceptive counter strategy. Example episodes between all 8 policies are visualized on the project website.

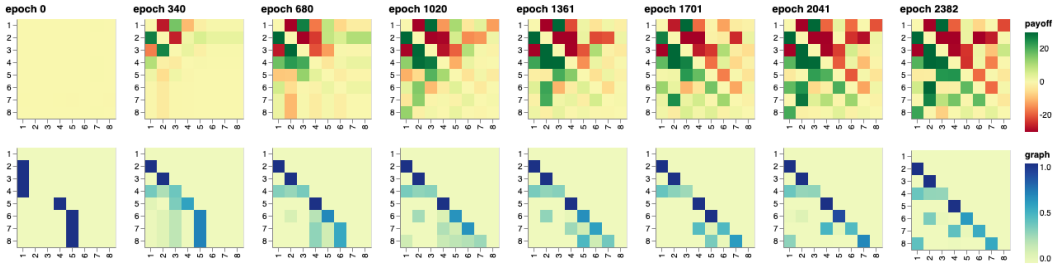


Figure 3: A NeuPL population developing an increasingly sophisticated set of diverse policies in *running-with-scissors*. The interaction graph is updated every 1,000 gradient updates (an epoch). **(Top)** the learned payoff estimates $\mathcal{U}$ between the neural population of policies as training progresses. **(Bottom)** interaction graphs $\Sigma \leftarrow \mathcal{F}_{\text{PSRO-N}}(\mathcal{U})$ as a response to the corresponding payoff matrix.

We verify quantitatively that NeuPL implementing $\mathcal{F}_{\text{PSRO-N}}$ similarly induces a policy population that becomes robust to adversarial exploits, as expected of PSRO-NASH (Lanctot et al., 2017). We compare independent policy populations and measure their Relative Population Performance (RPP, Appendix A.1). Intuitively, a negative RPP between populations $(\mathfrak{B}, \mathfrak{D})$ implies that there exists a mixture policy in $\mathfrak{D}$ that exploits all mixture policies over $\mathfrak{B}$. Figure 4 (Left) shows this metric measured at different stages of learning across 4 independent NeuPL experiments with different maximum population sizes. First, we observe that a neural population of maximum size two is easily exploited by all other populations and increasingly so as training progresses. This is expected, as the population converges to playing the two pure-resource policies ("rock" and "paper"). A larger population of size four is more robust, covering the single resource policies in addition to the policy that "observe-and-exploit". This population however becomes exploitable to the larger neural populations eventually as they discover exploiters that intentionally "mislead" its opponents. Finally, we note that a neural population of size greater than eight has limited impact on learning, both in terms

of additional robustness benefits by observing that relative population performance tends towards zero and their comparable rate of improvement against smaller populations (shown in blue and orange). In fact, Figure 4 (Right) shows that the effective population size $|\text{UNIQUE}(\Sigma)|$ represented by each neural population plateaus at 12, regardless of the specified maximum capacities of the neural populations. We hypothesise that this is due to the increased difficulty in discovering reliable exploiter policies in this domain. As noted in Section 1.2, $\mathcal{F}_{\text{PSRO-N}}$ forms an efficient curriculum where new objectives are introduced when the population discovers new meta-strategies worth exploiting.

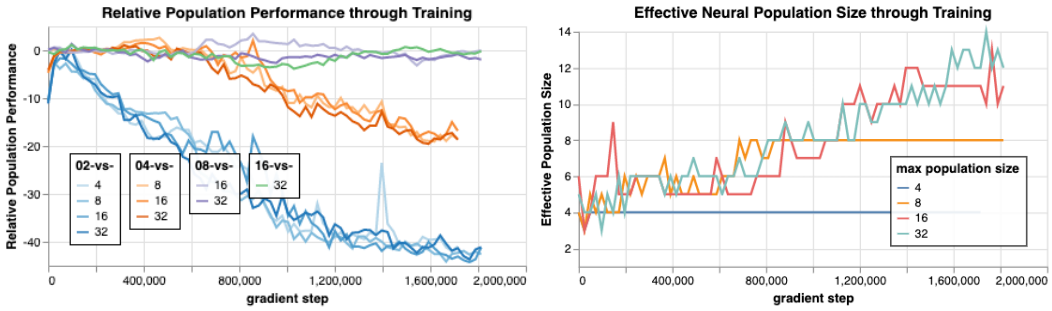


Figure 4: **(Left)** Relative Population Performance comparing independent NeuPL runs of different maximum population sizes. **(Right)** Effective population size over time, driven by $\mathcal{F}_{\text{PSRO-N}}$.

## 2.2 Does NeuPL enable Skill Transfer across Policies?

When coupled with deep RL, population learning algorithms such as Fictitious Self-Play and PSRO are often implemented as learning an approximate best response at each iteration *from scratch* (Heinrich et al., 2015; Lanctot et al., 2017; Mcaleer et al., 2020). In comparison, a NeuPL population represents different policies with a shared understanding of the game through parameter sharing. Figure 5 compares the two approaches and illustrates the important role of skill transfer offered by NeuPL. Specifically, we verify that the shared representation learned by training against fewer, weaker policies early in training, facilitates the learning of exploiters to stronger, previously unseen opponents. To verify this statement, a set of randomly initialized MPO agents with partially transferred parameters from different epochs shown in Figure 3 are optimized against the opponent mixtures defined by $\Pi_{\theta^*}^{\Sigma^*}$, where $\theta^*$ and $\Sigma^* = \mathcal{F}_{\text{PSRO-N}}(\mathcal{U}^*)$ are obtained at epoch 1,200 from the same NeuPL experiment. In other words, the objective for each MPO agent is to perform well against a fixed Nash mixture over pre-trained policies $\{\Pi_{\theta^*}(a|o_{\leq}, \sigma_k^*)\}_{k=1}^{n}$, with a specific $n$. Figure 5 shows the learning progressions of the MPO agents for three objectives, $n \in \{2, 4, 7\}$, with partially transferred parameters taken at epoch 0 (red) upto 1,000 (green).

Against an easily exploitable opponent mixture (Nash distribution over the first two pure policies), an agent with randomly initialized parameters (red) remains capable of learning an effective best response, albeit at a slower pace than agents partially initialized with transferred network parameters

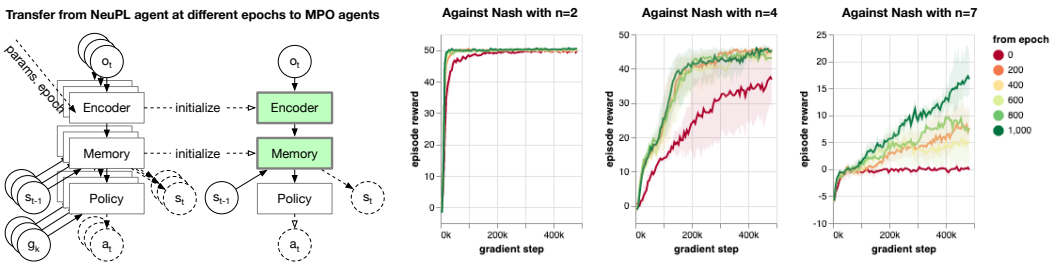


Figure 5: Learning progression of exploiters against incremental Nash mixture policies obtained via NeuPL training (as shown in Figure 3). The red curve corresponds to learning an exploiter from random initialization fully and the green curve corresponds to transferring encoder and memory components from the trained NeuPL network at epoch 1,000. Each experiment is repeated five times.

from later epochs. This difference becomes much more apparent against competent opponent mixtures that executes sophisticated policies (Nash distributions over the first 4 and 7 policies) where a randomly initialized agent fails to discover effective exploits from random initialization. By contrast, agents starting with partially transferred network parameters successfully discovered exploits, leveraging effective representation of the environment dynamics and crucially, of diverse opponents. Learning through NeuPL, this shared representation capacity enables the discovery of novel policies that are inaccessible to a randomly initialized agent. For a NeuPL population, learning incremental best responses becomes easier, not harder, as the agent gains better understanding of the game. This property is particularly attractive in real-world games where representation learning is often a challenge by itself (Jaderberg et al., 2019; Vinyals et al., 2019; Liu et al., 2021).

### 2.3 Does NeuPL Outperform Comparable Baselines?

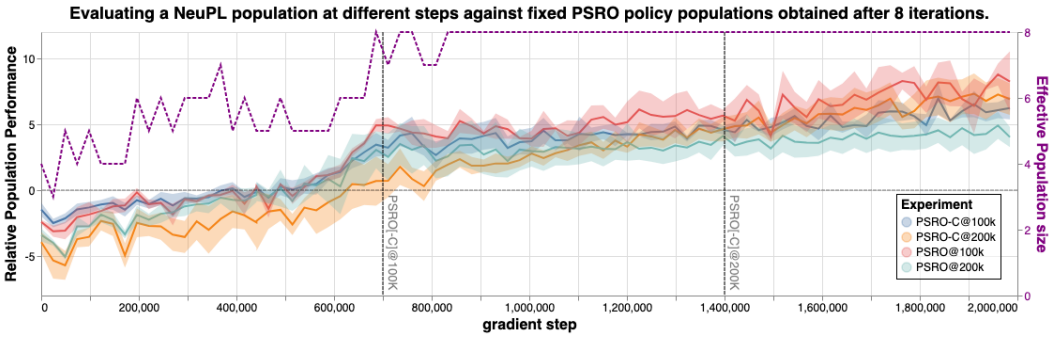


Figure 6: Relative Population Performance between a NeuPL population and policy populations obtained in PSRO baselines. Each PSRO variant is repeated with 3 trials, shown in shades.

We compare a NeuPL population implementing $\mathcal{F}_{\text{PSRO-N}}$ to four baselines implementing variants of PSRO. While PSRO does not prescribe a schedule for generating novel learning objectives, we investigate PSRO baselines with iteration length of 100k and 200k gradient steps respectively. Further, we consider the effect of continued training across iterations, by learning iterative best responses starting from the policy obtained at the end of the preceding iteration, instead of random initialization. We refer to this continued variant as PSRO-C. All PSRO populations are initialized with the same sink policy used in the NeuPL population. Figure 6 illustrates the quantitative benefits of NeuPL, measuring the Relative Population Performance between a NeuPL population of maximum population size of 8 against the *final* population of 8 policies obtained via PSRO after 7 iterations. A positive RPP value indicates that the NeuPL population beats the population of PSRO policies while a negative value indicates the opposite. The vertical dashed lines indicate that both the NeuPL population and the PSRO population have cumulatively undergone identical amount of training to allow for fair comparison. In purple, we show the effective population size of the NeuPL population which has been shown previously in Figure 4. We make the following observations: i) with a population size of 8, the NeuPL population successfully exploits PSRO baselines representing an equal number of policies, even when the latter received more training per iteration; ii) the increase in RPP coincides with an increase in the effective population size, from 5 to 8, reaching the maximum number of distinct policies that this NeuPL population can represent; iii) the amount of training each PSRO generation received has limited impact on the robustness of policy populations at convergence. This corroborates our observations of PSRO in Figure 5, where the agent (in red) failed to exploit strong opponents despite continued training. Interestingly, PSRO-C proves equally exploitable by the NeuPL population. We hypothesize that this is because at each iteration, the learned policy overfits to a specific opponent, rather than developing a versatile representation supporting diverse strategies. We show the payoff matrices for the PSRO baselines across three seeds in Appendix B.3.

### 2.4 Does NeuPL Scale to Complex Game-of-Skills?

In contrast to domains that primarily focus on strategic complexity, MuJoCo Football domain (Liu et al., 2019) challenges agents to integrate high-frequency, continuous actions to perform long-horizon team coordination. Figure 7 shows the learning progression of NeuPL implementing $\mathcal{F}_{\text{PSRO-N}}$ in

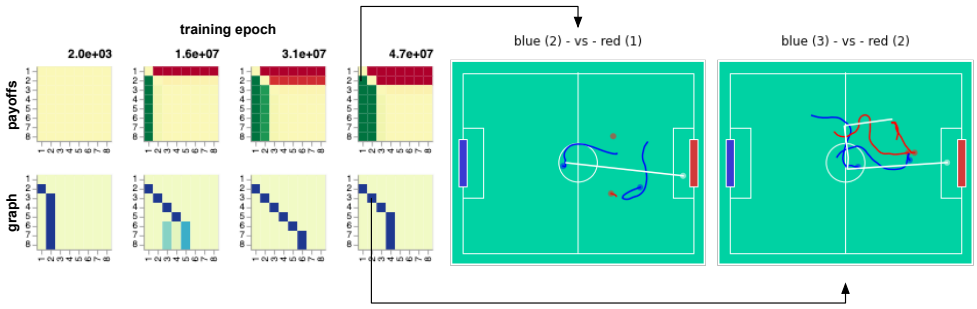


Figure 7: NeuPL in 2-vs-2 MuJoCo Football implementing $\mathcal{F}_{\text{PSRO-N}}$, with a static sink policy. The red/blue/white traces correspond to the trajectories of red/blue players and the ball respectively. The circles highlight the latest position of different entities.

this domain, starting with a sink policy that exerts zero torque on all actuators. Qualitatively, we observe that $\Pi_\theta(\cdot|o_{\leq}, \sigma_2)$ optimally exploits the sink policy by making rapid, long-range shots from a distance (Figure 7 **(Middle)**). This policy is in turn countered by $\Pi_\theta(\cdot|o_{\leq}, \sigma_3)$, which learned to intercept shots and coordinate as a team to score. Most impressively, the off-ball blue player blocked off red defenders, creating scoring opportunities for its teammate (Figure 7 **(Right)**).

Besides scaling up to this complex domain and developing a sequence of interesting and coordinated behaviours, NeuPL further revealed the underlying transitive structure of this domain, with the meta-graph solver $\mathcal{F}_{\text{PSRO-N}}$ concentrating on the lower-diagonal elements. This implies policy $i$ dominates $\Pi_{<i}$ and this regime effectively reduces to that of self-play, an effective strategy in purely transitive games (Balduzzi et al., 2019). For consistency, we used similar encoder, memory network architectures as in Liu et al. (2019) while keeping the RL algorithm the same with the rest of our experiments. Additional videos showing the behaviours of individual policies are available on the project website.

## 3 Related Work

Our work shares similar motivation as multi-task learning where a single agent learns to solve a range of related tasks (Caruana, 1997), either simultaneously, or in a continual setting (French, 1999; Barreto et al., 2020). In this work, we adopted similar techniques as in Schaul et al. (2015), but arguably in a more natural setting where tasks are generated by the task solvers themselves, calling for transfer between tasks, but also, over time. From a population learning perspective, our work targets the same setting as Lanctot et al. (2017), but embeds approximate best-responses with a focus on representational and computational efficiencies. Concurrent optimisation of several population-level objectives was featured in Garnelo et al. (2021), too, but in a setting where a fixed number of policies are trained. Vadori et al. (2020) attempted at representing policies of different types via a shared network, though types are pre-defined with domain-specific prior-knowledge.

## 4 Conclusion and Future Work

We proposed a principled population learning framework that optimizes a single conditional model to efficiently learn and represent a range of diverse policies in real-world games. Importantly, our work reduces the computational costs of population learning to that of self-play, making population learning on real-world games accessible to the broader research community. In this work, we restricted ourselves to simple network architecture and multi-task learning techniques but exploring suitable inductive biases in the context of structured population learning could be a productive direction. From a game theory perspective, generalizing NeuPL beyond the symmetric zero-sum setting (Muller et al., 2020; Marris et al., 2021) remain interesting future works. Finally, scaling up our method to very large-scale, strategically rich domains such as StarCraft could be an exciting next step.

## REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 434–443. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/balduzzi19a.html.

André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117 (48):30079–30087, 2020. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1907370117. URL https://www.pnas.org/content/117/48/30079. Publisher: National Academy of Sciences Section: Colloquium on the Science of Deep Learning.

George W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 1951.

Rich Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, July 1997. ISSN 0885-6125. doi: 10.1023/A:1007379606734. URL https://doi.org/10.1023/A:1007379606734.

Wojciech M. Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17443–17454. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/ca172e964907a97d5ebd876bfdd4adbd-Paper.pdf.

Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. ISSN 1364-6613. doi: https://doi.org/10.1016/S1364-6613(99)01294-2. URL https://www.sciencedirect.com/science/article/pii/S1364661399012942.

Marta Garnelo, Wojciech Marian Czarnecki, Siqi Liu, Dhruva Tirumala, Junhyuk Oh, Gauthier Gidel, Hado van Hasselt, and David Balduzzi. Pick your battles: Interaction graphs as population-level objectives for strategic diversity. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, pp. 1501–1503, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 805–813, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/heinrich15.html.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

John W Krakauer, Pietro Mazzoni, Ali Ghazizadeh, Roshni Ravindran, and Reza Shadmehr. Generalization of motor learning depends on the history of prior action. *PLOS Biology*, 4(10):1–11, 09 2006. doi: 10.1371/journal.pbio.0040316. URL https://doi.org/10.1371/journal.pbio.0040316.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3323fe11e9595c09af38fe67567a9394-Paper.pdf.

Siqi Liu, Guy Lever, Nicholas Heess, Josh Merel, Saran Tunyasuvunakool, and Thore Graepel. Emergent coordination through competition. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BkG8sjR5Km.

Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, S. M. Ali Eslami, Daniel Hennes, Wojciech M. Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, Noah Y. Siegel, Leonard Hasenclever, Luke Marris, Saran Tunyasuvunakool, H. Francis Song, Markus Wulfmeier, Paul Muller, Tuomas Haarnoja, Brendan D. Tracey, Karl Tuyls, Thore Graepel, and Nicolas Heess. From motor control to team play in simulated humanoid football. 2021. URL http://arxiv.org/abs/2105.12196.

Luke Marris, Paul Muller, Marc Lanctot, Karl Tuyls, and Thore Graepel. Multi-agent training beyond zero-sum with correlated equilibrium meta-solvers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7480–7491. PMLR, 18–24 Jul 2021. URL http://proceedings.mlr.press/v139/marris21a.html.

Stephen Mcaleer, JB Lanier, Roy Fox, and Pierre Baldi. Pipeline psro: A scalable approach for finding approximate nash equilibria in large games. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20238–20248. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/e9bcd1b063077573285ae1a41025f5dc-Paper.pdf.

H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 536–543. AAAI Press, 2003. ISBN 978-1-57735-189-4.

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, and Remi Munos. A generalized training approach for multiagent learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Bkl5kxrKDr.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/schaul15.html.

L. S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10): 1095–1100, 1953. ISSN 0027-8424. doi: 10.1073/pnas.39.10.1095. URL https://www.pnas.org/content/39/10/1095. Publisher: National Academy of Sciences _eprint: https://www.pnas.org/content/39/10/1095.full.pdf.

Yoav Shoham and Kevin Leyton-Brown. Multiagent systems: Algorithmic, game-theoretic, and logical foundations. In *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*, chapter 4. Cambridge University Press, 2008.

Nelson Vadori, Sumitra Ganesh, Prashant Reddy, and Manuela Veloso. Calibration of shared equilibria in general sum partially observable markov games. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14118–14128. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/a2f04745390fd6897d09772b2cd1f581-Paper.pdf.

Alexander Vezhnevets, Yuhuai Wu, Maria Eckstein, Rémi Leblond, and Joel Z Leibo. OPtions as REsponses: Grounding behavioural hierarchies in multi-agent reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9733–9742. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/vezhnevets20a.html.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

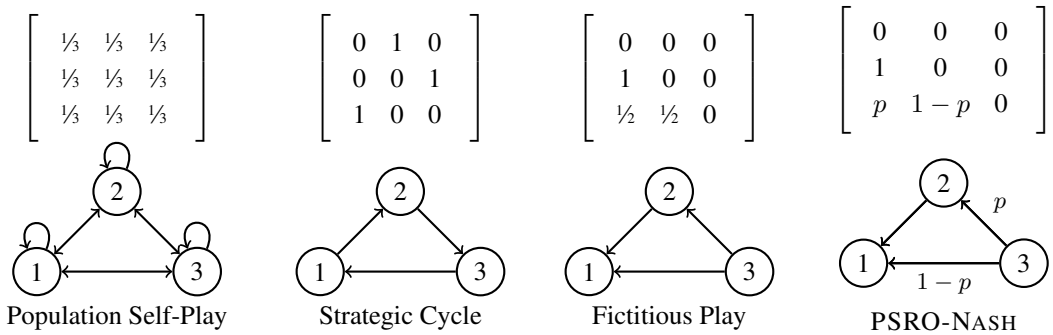


Figure 8: Example population learning algorithms implemented as directed interaction graphs (bottom row), or equivalently, a set of meta-game mixture strategies $\Sigma \in \mathbb{R}^{3\times 3} := \{\sigma_i\}_{i=1}^3$ (top row). A directed edge from $i$ to $j$ with weight $\sigma_{ij}$ indicates that policy $i$ optimizes against $j$, with probability $\sigma_{ij}$. Unless labeled, out edges from each node are weighted equally and the weights sum up to one.

# A METHODS

## A.1 POPULATION LEARNING ALGORITHMS AS INTERACTION GRAPHS

Figure 8 shows several population learning algorithms represented as directed interaction graphs (Garnelo et al., 2021). In particular, Policy Space Response Oracle with a Nash meta-game solver (PSRO-NASH) (Lanctot et al., 2017) proposes to learn a best-response $\pi_i$ to the Nash equilibrium over policies $\Pi_{<i}$ at each iteration, resulting in an adaptive interaction graph. Specifically, the out-edges of node $i+1$ are weighted according to the Nash equilibrium over the first $i$ policies (e.g. shown as $(1-p, p)$ for node 3 in Figure 8).

## A.2 RELATIVE POPULATION PERFORMANCE

Relative population performance was first introduced in Balduzzi et al. (2019) as the performance of individual agents is meaningless in purely cyclical games. Intuitively, a relative population performance measure of $v(\mathfrak{B}, \mathfrak{D})$ implies that there exists a mixture policy in the population $\mathfrak{B}$ that achieves a payoff at least $v(\mathfrak{B}, \mathfrak{D})$ against all mixture policies in the opponent population $\mathfrak{D}$. This measure is defined formally in Definition A.1.

**Definition A.1** (Relative Population Performance)**.** Given two populations of policies $\mathfrak{B}, \mathfrak{D}$ and let $(\mathbf{p}, \mathbf{q})$ be a Nash equilibrium over the zero-sum game on $\mathcal{U}_{\mathfrak{B},\mathfrak{D}} \in \mathbb{R}^{M\times N}$, Relative Population Performance measures their relative performance: $v(\mathfrak{B}, \mathfrak{D}) := \mathbf{p}^{\mathbf{T}} \cdot \mathcal{U}_{\mathfrak{B},\mathfrak{D}} \cdot \mathbf{q}$.

## A.3 NEUPL WITH STATIC INTERACTION GRAPH AND ABR BY RL

In this section, we describe a specialised implementation of NeuPL given *static* meta-graph solvers $\mathcal{F}$ with an approximate best-response procedure implemented via deep Reinforcement Learning. The algorithm is described in Algorithm 4.

## A.4 NEUPL WITH ADAPTIVE META-GRAPH SOLVERS AND ABR BY RL

Building on Algorithm 4, we now extend to the case of *adaptive* meta-graph solvers $\mathcal{F}$ that are functions of empirical payoff matrices $\mathcal{U}$. Specifically, the set of meta-game strategies the set of policies seek to best-respond to are given by $\Sigma \leftarrow \mathcal{F}(\mathcal{U})$. This algorithm is described in Algorithm 5.

# B EXPERIMENTS

## B.1 RUNNING-WITH-SCISSORS ENVIRONMENT

Figure 9 shows an example view of the *running-with-scissors* environment at the start of an episode. The dashed squares shows the type of resources initialized in the enclosed area with some of them

**Algorithm 4** Neural Population Learning by RL (static $\mathcal{F}$)

**function** RUN-EPISODES($\Sigma, \Pi_\theta, M$) ▷ Visualized in Figure 10 (Right).
  $\mathcal{D}_\Pi \leftarrow \{\}; \mathcal{D}_Q \leftarrow \{\}$ ▷ Initialize actor, critic trajectory buffers.
  **for** Episode $\in \{1, \ldots, M\}$ **do** ▷ Generate trajectories for $M$ episodes.
    Sample $\sigma_i \sim$ UNIQUE($\Sigma$) uniformly (excluding sink nodes) and $\sigma_j \sim P(\sigma_i)$. ▷ Match-making.
    Sample trajectories $\mathcal{T}^{\sigma_i}, \mathcal{T}^{\sigma_j}$ from interactions between $(\Pi_\theta(\cdot|o, \sigma_i), \Pi_\theta(\cdot|o', \sigma_j))$.
    $\mathcal{D}_\Pi \leftarrow \mathcal{D}_\Pi \cup \mathcal{T}^{\sigma_i}; \mathcal{D}_Q \leftarrow \mathcal{D}_Q \cup \mathcal{T}^{\sigma_i} \cup \mathcal{T}^{\sigma_j}$.
  **return** $\mathcal{D}_\Pi, \mathcal{D}_Q$

**function** NEUPL-RL-STATIC($\Sigma, T$) ▷ $\Sigma \in \mathbb{R}^{N \times N}$ the static interaction graph.
  Initialize $\Pi_\theta(a|o, \sigma_i)$, the task-conditioned policy.
  Initialize $Q_\theta(o, a, \sigma_i, \sigma_j)$, the action-value function.
  **for** Iteration $t \in \{1, \ldots, T\}$ **do** ▷ $T$ the number of iterations to run for.
    $\mathcal{D}_\Pi, \mathcal{D}_Q \leftarrow$ RUN-EPISODES($\Sigma, \Pi_\theta, M$)
    Optimize policy $\Pi_\theta$ from $\mathcal{D}_\Pi$ and $Q_\theta$ from $\mathcal{D}_Q$ by RL.

**Algorithm 5** Neural Population Learning by RL (adaptive $\mathcal{F}$)

**function** NEUPL-RL-ADAPTIVE($\mathcal{F}, K, T, M, \epsilon$) ▷ $\mathcal{F} : \mathbb{R}^{N \times N} \to \mathbb{R}^{N \times N}$.
  Initialize $\Pi_\theta(a|o_{\leq}, \sigma_i)$, the neural population network.
  Initialize $Q_\theta(o_{\leq}, a, \sigma_i, \sigma_j)$, the action-value function.
  Initialize $\phi_\omega(\sigma_i, \sigma_j)$, the empirical payoff estimator.
  Let $\Sigma_c \in \mathbb{R}^{N \times N} \leftarrow \mathbf{1}/\mathbf{N}$. ▷ All-to-all interactions.
  **for** Epoch $\in \{1, \ldots, K\}$ **do** ▷ $K$ epochs to run.
    $\forall i, j : \mathcal{U}_{ij} \leftarrow \phi_\omega(\sigma_i, \sigma_j)$ the payoff matrix over $N$ policies. ▷ Re-compute payoff estimates.
    $\Sigma \in \mathbb{R}^{N \times N} \leftarrow \mathcal{F}(\mathcal{U})$ the interaction graph over $N$ policies. ▷ Update the interaction graph.
    **for** Iteration $t \in \{1, \ldots, T\}$ **do** ▷ $T$ iterations per epoch.
      $\mathcal{D}_\Pi, \mathcal{D}_Q \leftarrow$ RUN-EPISODES($\Sigma, \pi_\theta, M \times (1 - \epsilon)$) ▷ $M$ episodes per iteration.
      $_, \mathcal{D}_Q^\epsilon \leftarrow$ RUN-EPISODES($\Sigma_c, \pi_\theta, M \times \epsilon$) ▷ $\epsilon$ the prop. of evaluation episodes.
      Optimize policy $\pi_\theta$ from $\mathcal{D}_\Pi$ and $Q_\theta$ from $\mathcal{D}_Q \cup \mathcal{D}_Q^\epsilon$ by RL.
      Optimize $\phi_\omega$ from $\mathcal{D}_Q \cup \mathcal{D}_Q^\epsilon$ by minimizing $\mathcal{L}_{ij}$.

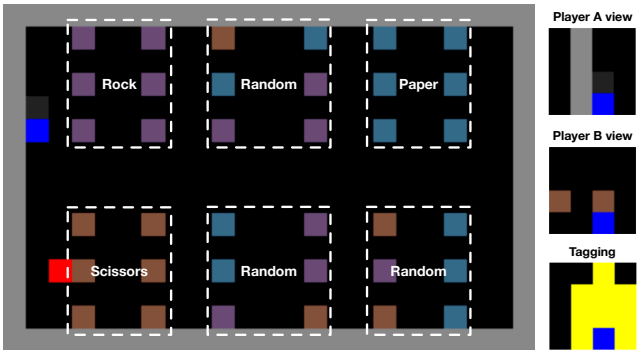


Figure 9: An example view of the *running-with-scissors* environment upon initialization.

consistently initialized with one type of resources while others randomly initialized with one of the three types. On the right, we visualize the 4x4 pixel observations of the two players. The visual observation is oriented along each player's forward direction. In addition to the visual observations, each player observes their current inventory of the three types of resources, expressed in terms of their normalized ratios. Each player is initialized with an equal weight inventory at the start of an episode. To move around, a player can turn left, turn right, strafe left, strafe right, move forward or move backwards. Finally, each player can proactively seek out its opponent and "tag" it to terminate the game. A player is considered tagged if it falls into the tagging area of its opponent. Bottom right view illustrates the shape of the tagging area in front of a player. If neither player tags its opponent, the game ends after a fixed number of 500 steps. On the terminal step, the game resolves by comparing the inventory of both players and the rewards are assigned according to the classical anti-symmetric payoff matrix of *rock-paper-scissors*. On all other steps, both players receive zero rewards.

### B.2 Network Architecture, Hyper-Parameters & Training Infrastructure

Figure 10 (Left) illustrates the general network architecture of a NeuPL population. For experiments on the normal form game of *rock-paper-scissors*, the "Encoder" and "Memory" components are omitted and the policy head network $\pi_\theta$ alone is parameterized and learned, with its only input $g$. The policy and action-value networks are both parameterized by 4-layer MLPs, with 32 neurons at each layer. We use a small entropy cost of $0.01$, learning rates of $0.001$ and $0.01$ for the main networks and the MPO dual variables (Abdolmaleki et al., 2018) respectively. As is typical in an MPO agent, the online network parameters are copied to target networks periodically at every 10 gradient steps. For the spatiotemporal, partially observable game of *running-with-scissors*, we use a small convolutional network to encode the agent's partial observation of the environment (a 4x4 grid surrounding itself). The encoding is further concatenated with the agent's own inventory information and encoded through a 2-layer MLP with 256 neurons each, with `relu` activation. The memory module corresponds to a recurrent LSTM network, with a hidden size of 256. The policy, action-value networks are parameterized as 4-layer MLP networks. The learned payoff estimator $\phi_\omega(\sigma_i, \sigma_j)$ is parameterized as a 3-layer MLP network. The learning rate of the agent networks is set to $0.0001$ while the MPO dual variables are optimized with a learning rate of $0.001$. The online network parameters are copied to target networks every 100 gradient steps. The same network architecture and hyper-parameters are used across all experiments for the same environment.

Figure 10 (Right) illustrates how self-play experience data is generated for a NeuPL population at training time. At the start of an episode, a pair of conditioning vectors is sampled $(\sigma_i, \sigma_j)$ and used to condition the pair of policies interacting in the game. In *running-with-scissors*, each NeuPL uses 128 actor workers running the policy environment interaction loops and a single TPU-v2 chip running gradient updates to the agent networks. The same computational resources are used across different maximum population size for NeuPL as well as the PSRO baseline experiments. For the game of *rock-paper-scissors*, a single CPU worker is used instead. In *running-with-scissors*, all experiments converge within 2 days wall-clock time. For all experiments using NeuPL, an evaluation split $\epsilon = 0.3$ is used.

### B.3 Baseline PSRO experiments

Figure 11 visualizes the empirical payoff matrices obtained in 12 independent experiments. In each experiment, we develop a discrete population of policies via 8 iterations of PSRO training. In particular, we show the payoff matrices between trained policies where each PSRO iteration lasts 100k or 200k gradient steps. PSRO-C further shows the effect of initializing each iteration by the policy obtained at the end of the preceding iteration instead of random initialization as prescribed in (Lanctot et al., 2017). We note that across all experiments, policy populations successfully recover the *rock-paper-scissors* dynamics among the first three policies. In most but not all trails, PSRO also manages to learn a reliable exploiter of the mixture of the three pure strategies. Finally, we note that continued training appears to facilitate the discovery of richer strategic cycles, and so does allowing each PSRO iteration to perform more gradient updates.

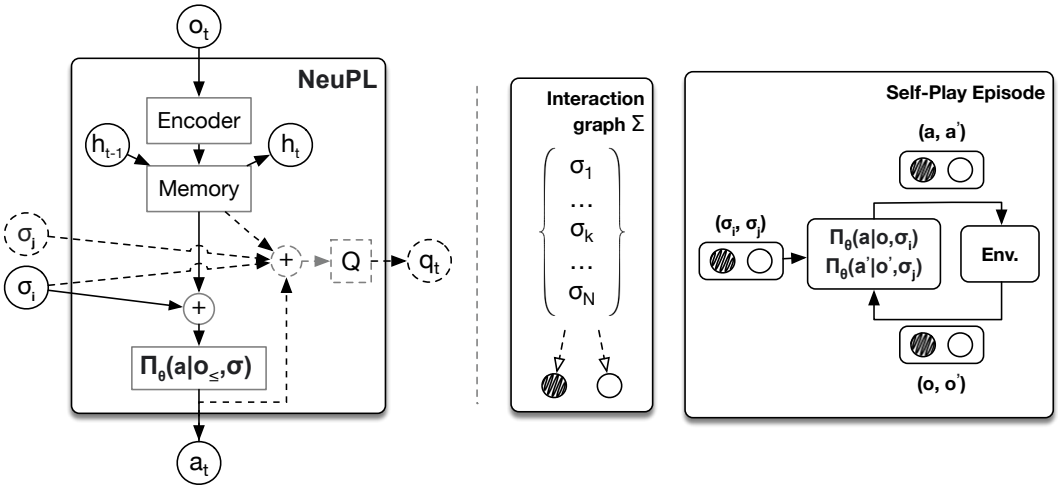


Figure 10: **(Left)** The conditional neural population network architecture for a Q-learning based RL agent. The "$\bigoplus$" sign denotes a concatenation of all inputs tensors and dashed components are only used for training (and not necessary for acting); **(Right)** A diagram illustrating the mechanism of a NeuPL self-play episode: at the start of an episode, a pair of conditioning vectors are sampled $(\sigma_i, \sigma_j)$ from the interaction graph $\Sigma$. The pair of policies, using the same conditional policy network $\Pi_\theta(a|o_{\leq}, \sigma)$, act simultaneously in the environment. The circles correspond to the conditioning vectors, observations and actions for the exploiter policy (shaded) and its opponent (blank).

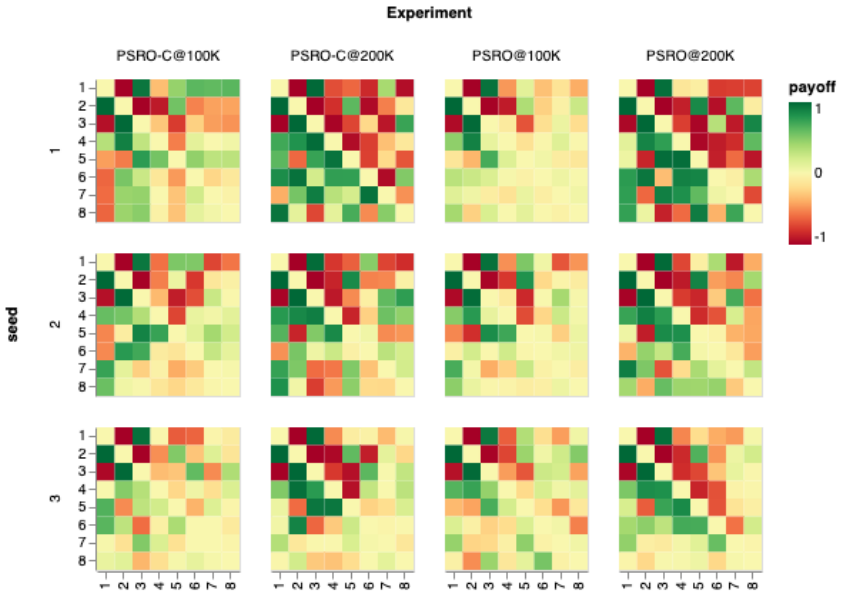


Figure 11: Visualization of empirical payoff matrices for each PSRO population after 8 iterations across three independent trials. Each iteration lasts 100k or 200k gradient steps. PSRO-C indicates that the policy trained at iteration $i+1$ is initialized from the policy obtained at iteration $i$ instead of from random initialization. A payoff of 1.0 (-1.0) indicates a win (loss) probability of 100% for the row player.

# C CONVERGENCE PROOFS

In this section we detail the conditions upon which NeuPL will converge to a unique sets of policies. First, we discuss the theory when the interaction graph is static and grounded (Section C.1). Then we discuss the theory when the interaction graph is a function of the payoff matrix, a so-called meta-graph solver (Section C.2). Finally we discuss a specific function, Nash Equilibrium meta-graph solver, that is popular in the literature, and prove NeuPL's convergence to a normal-form Nash Equilibrium under certain conditions (Section C.3).

## C.1 GROUNDED INTERACTION GRAPHS

**Assumption C.1** (Unique, Exact, Finite Best Response)**.** *Assume that we have access to a best response (BR) operator that responds to a policy, $\pi_n$, (or equivalently a mixture over policies) exactly converges to a unique best response, $\pi_{n+1} = BR(\pi_n)$, in finite time.*

When using RL as the BR operator, uniqueness can be approximated with a small entropy term on policy being optimized. Therefore, in the situation where there is more than one best response, the BR operator will opt for the one with maximum entropy. This is a common additional loss that is often added to RL agents, and is also believed to aid exploration (Haarnoja et al., 2018).

The second assumption, that the BR will be exact, is idealised. With sufficient model capacity, enough time, and appropriate hyper-parameter annealing schedules, RL can get close to an exact BR.

The final assumption, that the BR operator converges in finite time, is assumed because proofs follow easily from this assumption. It may be possible to relax this assumption to converge as $t \to \infty$, but we leave this for future work.

**Definition C.1** (Grounded Interaction Graph)**.** An interaction graph's edges define how policies should best respond to each other. We call an interaction graph *grounded* if its structure imposes convergence to a unique set of policies, where each policy is a unique, exact, finite BR over opponent policies defined by the interaction graph.

**Theorem 1** (Grounded Lower Triangular)**.** *All lower-triangular interaction graphs ($\Sigma_{i \geq j} = 0$) are grounded.*

*Proof.* The first row of a lower triangular matrix is all zeros meaning that the first policy does not respond to any other policies. The first policy is therefore static (and arbitrary) and does not change over time. The second row only responds to the first policy. Since the first policy is static and we are using a unique, exact and finite BR, the second policy will converge exactly to a unique policy. The $n$th row will respond to the $n-1$ previous policies. Since over time all previous policies will converge to uniquely, so will the $n$th policy. □

It is easy to imagine other grounded interaction graphs with interesting structure, however we will only focus on lower-triangular graphs for the purpose of this section. Research into other grounded interaction graphs may be interesting future work.

**Definition C.2** (Static Interaction Graph)**.** We call an interaction graph *static* if it does not change over time: $\Sigma^t = \Sigma$.

We now prove our first NeuPL result, that it can find an exact $N$-step best response under certain assumptions.

**Theorem 2** (NeuPL Static Lower-Triangular Exact $N$-Step Best Response)**.** *NeuPL with a static, lower triangular interaction graph, an arbitrary fixed initial policy, $\pi_0$, and $N$ more policies, will converge to an exact $N$-step best response, assuming that the BR is unique, exact, and converges in finite time.*

*Proof.* The bulk of this proof can recycle the arguments in Theorem 1. Note that NeuPL trains its conditional policies in parallel according to an interaction graph. If that graph is lower-triangular, it is grounded, and NeuPL will find the conditioned policies corresponding to the $N$-step best response. □

### C.2 GROUNDED META-GRAPH SOLVER

It is possible to extend this result on any deterministic function acting on a sub-payoff, producing a lower triangular interaction graph.

**Definition C.3** (Grounded Meta-Graph Solver)**.** A meta-graph solver is a function that takes the payoff matrix as an argument and outputs an interaction graph as an output. We call a meta-graph solver, $\mathcal{F}$, *grounded* if, when using unique, exact, finite BRs, it converges to a grounded interaction graph in finite time.

**Theorem 3** (Any Deterministic Lower-Triangular Meta-Graph Solver is Grounded)**.** *Any deterministic function, $\mathcal{F}$, that maps a* sub-payoff, *$\mathcal{U}_{<i,<i}$, to a row, $\Sigma_{i,<i}$, in a lower-triangular interaction graph is a grounded meta-graph solver.*

$$\Sigma_{i,<i} = \mathcal{F}(\mathcal{U}_{<i,<i}) \tag{2}$$

*Proof.* The difficulty here is that in general, as the policies change, so will the payoff, and hence so will interaction graph, etc. Similar to the arguments in Theorem 1 the first policy is static and does not change. The second policy can only respond to the first policy (under the lower-triangular constraint), so the meta-graph solver has no flexibility to do otherwise, and therefore the second policy will also converge to a unique result. Note that as these policies converge, so will their sub-payoffs, $\mathcal{U}_{<3,<3}$. Therefore, for row $n$ any deterministic mapping $\mathcal{F} : \mathcal{U}_{<i,<i} \to \Sigma_{i,<i}$ will result in a unique set of policies. □

We can now make a further claim about NeuPL: that it will converge to a unique set of conditioned policies under certain grounded meta-graph solvers.

**Theorem 4** (NeuPL Deterministic Lower-Triangular Exact $N$-Step Best Response)**.** *Assuming any deterministic lower-triangular meta-graph solver, an arbitrary fixed initial policy, $\pi_0$, and $N$ more policies, NeuPL will converge to an exact $N$-step best response, assuming a unique, exact and finite BR.*

*Proof.* Similar in structure to Theorem 2 and Theorem 3. □

Of course, the more general result, that NeuPL will also converge for any grounded meta-graph solver is also true.

### C.3 NASH EQUILIBRIUM META-GRAPH SOLVER

Note that the maximum entropy Nash Equilibrium (MENE) is such as grounded meta-graph solver. This will result in an algorithm similar to PSRO-NASH (Lanctot et al., 2017) or Double Oracle (McMahan et al., 2003).

**Theorem 5** (NeuPL Nash Exact $N$-Step Best Response)**.** *Using an interaction-graph function that maps payoff to a lower triangular NE distribution:*

$$\mathcal{G}_{n,<n} = NE(U_{<n,<n}) \tag{3}$$

*With a such a function, an arbitrary fixed initial policy, and sufficiently large $N$ policies, NeuPL will converge to an exact normal-form Nash Equilibrium (NE), assuming a unique, exact, finite BR.*

*Proof.* Starting from the proof in Theorem 4, we follow additional arguments from DO (McMahan et al., 2003) to prove that for sufficient $N$ we will converge to a normal form NE. □

Of course, many other algorithms can be recovered using specific meta-graph solvers (Muller et al., 2020; Marris et al., 2021).