ROTATE: Regret-driven Open-ended Training for Ad Hoc Teamwork

Caroline Wang[1]* **Arrasy Rahman**[1]* **Benjamin Nativi**[2] **Johnny Liu**[1] **Jiaxun Cui**[1]
Yoonchang Sung[3] **Peter Stone**[1,4]
[1]The University of Texas at Austin [2]Visa AI [3]Nanyang Technological University [4]Sony AI
{caroline.l.wang, jyliu, cuijiaxun}@utexas.edu, yoonchang.sung@ntu.edu.sg,
{arrasy, pstone}@cs.utexas.edu, benjamin.nativi@outlook.com

Abstract

Learning to collaborate with previously unseen partners is a fundamental generalization challenge, known as Ad Hoc Teamwork (AHT). Existing methods often adopt a two-stage pipeline: first, a fixed population of *teammates* is generated, and second, an *AHT agent* is trained to collaborate with them. As a consequence, training teammates are generated ignoring whether they are informative for the AHT agent to learn from. Similarly, AHT agents are typically trained under the assumption that the training teammate set is uncontrollable, though its composition strongly influences generalization. This paper presents a unified framework for AHT by reformulating the problem as an open-ended learning process between an AHT agent and an adversarial teammate generator. We introduce ROTATE, a regret-driven, open-ended training algorithm that alternates between improving the AHT agent and generating teammates that probe its collaboration deficiencies. Experiments across Overcooked and Level-Based Foraging tasks demonstrate that ROTATE substantially outperforms baselines on unseen teammates, establishing a new standard for robust, generalizable teamwork.

1 Introduction

As AI agents are deployed in diverse applications, it is increasingly crucial that they can collaborate effectively with previously unseen AI agents and humans. While methods for training teams of agents have been explored in cooperative multi-agent reinforcement learning (CMARL) [22, 51], prior work highlighted that CMARL agents fail to perform optimally when collaborating with unfamiliar teammates [65, 48]. Dealing with previously unseen teammates requires adaptive AI agents that efficiently approximate the optimal strategy for collaborating with diverse teammates. The training of such adaptive agents has been explored within ad hoc teamwork (AHT) [8, 58, 39] and zero-shot coordination (ZSC) [23, 17, 38].

Most work has decomposed AHT learning into two stages [39], consisting of first creating a fixed set of teammates, and then training an AHT agent using reinforcement learning (RL), based on interactions with teammates sampled from the set. Methods for AHT agent learning typically rely on human-designed or pretrained teammates [46, 77, 48] and therefore struggle to handle novel behaviors outside the predefined set of teammates [59, 10]. Recent work enhances the generalization capabilities of AHT agent learning methods by substituting the predefined set of teammates with a generated collection of diverse teammates [38, 50], which are trained to maximize different notions of diversity. One such diversity notion is *adversarial diversity* [49, 11], which seeks to generate a set of teams that cooperate well with other team members, but not across different teams.[2]

*Equal contribution. [2]Appendix A reviews prior work on AHT, teammate generation, and open-ended learning.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

This paper addresses two issues that cause current methods to fail to learn policies that effectively collaborate with some teammates. First, two-stage AHT methods [46, 77, 48, 49] learn from interacting with teammates from a small fixed training set. Even when the training set is diverse, the AHT agent remains incapable of collaborating effectively with some teammates sampled from the vast space of possible strategies, specifically those with significantly different behavior from the policies in the training set [65, 48]. Second, other works generate diverse teammate behaviors iteratively [75], but maximize adversarial diversity [11, 49], which yields teammates whose return-diminishing tendencies in certain states make it challenging for a randomly initialized RL-based AHT agent to learn to collaborate with them effectively [18, 54, 32].

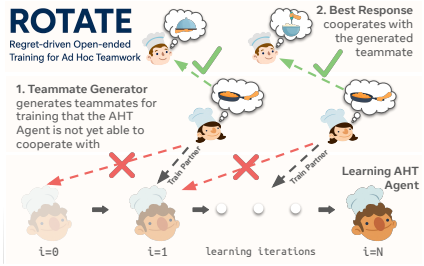


Figure 1: **ROTATE Overview.** ROTATE is an open-ended AHT learning algorithm that improves the AHT agent by iteratively generating diverse teammates with whom the AHT agent struggles yet can still learn to collaborate.

We address the above issues by proposing an open-ended learning framework that continually generates diverse yet cooperative and competent teammates. We formulate our learning objective by observing that maximizing the expected returns of an AHT agent on a known set of teammates is equivalent to minimizing its expected ***cooperative regret***: the utility gap between the best response to a given teammate, and the AHT agent's performance with that teammate. We take inspiration from unsupervised environment design (UED) methods [68, 19, 26, 52] and train an AHT agent to minimize its regret against generated teammates designed to maximize the AHT agent's cooperative regret. Unlike UED methods that only maximize regret from starting states, we generate teammates by maximizing regret in all encountered states while imposing competence constraints. This design aims to avoid generating non-cooperative and incompetent teammate policies during training, which are unlikely to be encountered during evaluation. Building on these foundations, we propose a practical algorithm, ROTATE (Fig. 1), which optimizes a cooperative regret-based minimax objective while maintaining a population of all previously designed teammates.

This paper makes four main contributions. First, we reformulate AHT as a regret-driven game between a teammate generator and the AHT agent, motivating an open-ended AHT learning procedure that continually generates new teammates. Second, we characterize *sabotage* as a property of an objective that induces *incompetent* behavior from teammates, and propose a competence-bounded per-state regret objective that provably bounds the competence of generated teammates on states encountered during training. Third, we propose a novel algorithm, ROTATE, that instantiates the proposed open-ended AHT framework. Fourth, extensive empirical evaluations demonstrate that ROTATE significantly improves return against unseen teammates compared to representative baselines from AHT and open-ended learning on Overcooked and Level-Based Foraging.

2 Problem Formulation

This section provides a brief formalization of the AHT problem explored in existing literature [39]. We start by formalizing the interaction between the learning agent and its teammates. We then detail the objectives of existing AHT learning methods.

The interaction between agents in an AHT setting is formalized as a decentralized Markov decision process (Dec-MDP) [5]. A Dec-MDP is characterized by a 7-tuple, $\langle N, S, \{\mathcal{A}^i\}_{i=1}^{|N|}, P, p_0, R, \gamma \rangle$, where N, S, and γ respectively denote the index set of agents within an interaction, the state space, and a discount rate, $0 \leq \gamma \leq 1$. Every interaction between agents begins from a state sampled from the initial state distribution, $s_0 \sim p_0$. At timestep t, each agent, $i \in N$, jointly executes an action selected from its action space, $a_t^i \in \mathcal{A}^i$, based on the observed state, s_t, and its policy, $\pi^i(s_t)$. We assume that teammates choose their actions only based on the current state. Meanwhile, the AHT agent, also referred to as the *ego agent*, selects actions based on its state-action history, which is necessary to distinguish between different types of teammates effectively. Denoting the set of all probability distributions over a set X as $\Delta(X)$, the execution of the joint action, $a_t = (a_t^1, \ldots, a_t^{|N|})$, results in agents observing a new state, s_{t+1}, sampled according to the environment transition function, $P : S \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^{|N|} \mapsto \Delta(S)$, and a common scalar reward, r_t, based on the reward function, $R : S \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^{|N|} \mapsto \mathbb{R}$.

Within the Dec-MDP framework, AHT methods aim to train ego agents to achieve maximal returns when collaborating with a set of *unknown yet competent* teammate policies, denoted as Π^{eval} [58]. We designate the ego agent as agent $i \in N$, and the remaining $|N|-1$ agents as its *teammates*. Let π^{ego} refer to the ego agent's policy, and π^{-i} denote its $|N|-1$ teammates' joint policy. We denote the return of the team formed by the ego agent and the teammates, starting from state s_0, by:

$$V(s_0|\pi^{-i}, \pi^{\text{ego}}) = \mathbb{E}_{a_t^{\text{ego}} \sim \pi^{\text{ego}},\, a_t^{-i} \sim \pi^{-i}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 \right]. \tag{1}$$

Given a teammate policy π^{-i}, $\text{BR}(\pi^{-i})$ is a best response policy to π^{-i} if and only if the team policy formed by π^{-i} and $\text{BR}(\pi^{-i})$ achieves maximal return, i.e., $\text{BR}(\pi^{-i}) \in \arg\max_{\pi} \mathbb{E}_{s_0 \sim p_0}[V(s_0|\pi^{-i}, \pi)]$. We refer to interactions between a teammate π^{-i} and the ego agent as *cross-play* (**XP**), and to interactions between π^{-i} and its best response $\text{BR}(\pi^{-i})$ as *self-play* (**SP**). We define the set of p_0-competent teammate policies as the set of teammates that achieve a team return of at least K with their best response, $\Pi_K^{-i}(p_0) = \{\pi^{-i} : \mathbb{E}_{s_0 \sim p_0}[V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))] \geq K\}$. In the AHT problem, evaluation teammates are assumed to be competent, i.e. $\Pi^{\text{eval}} \subseteq \Pi_K^{-i}(p_0)$.

Let $\psi^{\text{eval}}(\Pi^{\text{eval}})$ denote the probability distribution used during evaluation to sample teammates from Π^{eval}. The **AHT objective** is formalized as finding an ego policy π^{ego} that maximizes the expected return when collaborating with joint teammate policies sampled from $\psi^{\text{eval}}(\Pi^{\text{eval}})$:

$$\max_{\pi^{\text{ego}}} V(\psi^{\text{eval}}, \Pi^{\text{eval}}, \pi^{\text{ego}}) = \max_{\pi^{\text{ego}}} \mathbb{E}_{\pi^{-i} \sim \psi^{\text{eval}}(\Pi^{\text{eval}}),\, s_0 \sim p_0} \left[V\left(s_0 \mid \pi^{-i}, \pi^{\text{ego}}\right) \right]. \tag{2}$$

An optimal π^{ego} that maximizes Eq. 2 adaptively approximates the *best response policy* performance when collaborating with $\pi^{-i} \in \Pi^{\text{eval}}$.

Since Π^{eval} is unknown, AHT algorithms [39] learn by interacting with policies from the training set, Π^{train}, which are learned or manually designed by leveraging an expert's domain knowledge about the characteristics of Π^{eval}. After forming Π^{train}, current AHT algorithms use RL to discover an ego agent policy. While the precise training objective varies with the AHT algorithm, methods commonly maximize the expected return during interactions with teammate policies sampled uniformly from Π^{train}:

$$\pi^{*,\text{ego}}(\Pi^{\text{train}}) \in \underset{\pi^{\text{ego}}}{\text{argmax}}\, \mathbb{E}_{\pi^{-i} \sim \mathcal{U}(\Pi^{\text{train}}),\, s_0 \sim p_0} \left[V\left(s_0 \mid \pi^{-i}, \pi^{\text{ego}}\right) \right]. \tag{3}$$

Naturally, $\pi^{*,\text{ego}}(\Pi^{\text{train}})$ may be suboptimal with respect to Π^{eval} and ψ^{eval}, if there is distribution shift between the training and evaluation objectives.

3 Ad Hoc Teamwork as an Open-Ended Learning Problem

In this section, we reformulate the AHT objective (Eq. 2) as a minimax regret game between a teammate generator and an ego agent learner, suggesting an open-ended learning procedure to address AHT problems. We show that for a fixed set of teammates Π^{eval} and sampling distribution ψ^{eval} over Π^{eval}, maximizing the return of the ego agent is equivalent to minimizing its cooperative regret. In the absence of knowledge about Π^{eval} and ψ^{eval}, we argue that minimizing the *worst-case* cooperative regret of the ego agent with respect to regret-maximizing, competent teammates leads to ego agents that cooperate well with unknown, competent teammates. Based on this argument, we propose a novel *minimax regret* objective (Eq. 5). Finally, we present an algorithmic framework for optimizing the minimax regret objective in an iterative fashion (Alg. 1).

Definition 1 (Cooperative Regret)**.** *The **cooperative regret** of an ego policy π^{ego} when interacting with a joint teammate policy π^{-i} from a starting state s_0 is:*

$$CR(\pi^{ego}, \pi^{-i}, s_0) = V\left(s_0|\pi^{-i},\, BR(\pi^{-i})\right) - V\left(s_0|\pi^{-i}, \pi^{ego}\right).$$

Observe that any ego policy that *maximizes* the AHT objective (Eq. 2) must also *minimize* the expected regret over joint teammate policies sampled from $\psi^{\text{eval}}(\Pi^{\text{eval}})$:

$$\text{CR}(\psi^{\text{eval}}, \Pi^{\text{eval}}, \pi^{\text{ego}}) = \mathbb{E}_{\pi^{-i} \sim \psi^{\text{eval}}(\Pi^{\text{eval}}), s_0 \sim p_0} \left[\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0) \right], \tag{4}$$

which results from $V\left(s_0|\pi^{-i},\, \text{BR}(\pi^{-i})\right)$ being independent of π^{ego} for any π^{-i} and s_0.

The above equivalence suggests training ego policies to minimize the expected cooperative regret, $\text{CR}(\psi^{\text{eval}}, \Pi^{\text{eval}}, \pi^{\text{ego}})$. However, as Π^{eval} and ψ^{eval} are unknown, we propose optimizing π^{ego} to minimize the *worst-case regret* that could be induced by any teammate policy within the set of

p_0-competent teammates, $\pi^{-i} \in \Pi_K^{-i}(p_0)$:

$$\min_{\pi^{\text{ego}}} \max_{\pi^{-i} \in \Pi_K^{-i}(p_0)} \mathbb{E}_{s_0 \sim p_0} \left[\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0) \right]. \tag{5}$$

The minimax regret game in Eq. 5 has two properties that make it a suitable training objective for AHT. First, for the ego agent, finding π^{ego} that achieves zero worst-case regret is equivalent to finding an ego agent that achieves the best-response return with any joint teammate policy π^{-i}. If such a π^{ego} exists, then this AHT agent maximizes Eq. 2 for any ψ^{eval} and Π^{eval}. However, its existence is not guaranteed.[3] In practice, we are therefore content with *minimizing* the worst-case regret. Second, for the teammate, maximizing the regret of the ego agent encourages discovering π^{-i} for which the ego agent policy has substantial room for improvement, inducing a curriculum for the ego agent learner. In UED, similar minimax regret formulations have successfully induced curricula to improve the ability of RL agents to generalize to unseen environment variations [19, 68]. Note that regret maximization does not require access to an exact best response, so $\text{BR}(\pi^{-i})$ may be trained with RL, rather than assumed to be available.[4]

In Eq. 5, constraining regret maximization to p_0-competent teammates is especially important, both to align the optimization procedure to the AHT problem definition, and because unconstrained regret maximization may lead to incompetent teammates that perform poorly against any π^{ego}. Such teammates are undesirable because they introduce unnecessary learning challenges for RL-based AHT learning algorithms, and are unlikely to be encountered in coordination scenarios. We expand upon this discussion in Sec. 4, where we propose a practical regret-maximization objective.

Algorithm 1 Open-Ended Ad Hoc Teamwork Framework

1: **Input:** Environment Env, iterations T^{iter}, initial ego parameters θ^{ego}
2: Initialize teammate policy buffer $\text{B}_\pi \leftarrow \emptyset$
3: **for** $j = 1$ **to** T^{iter} **do**
4: $\quad \text{B}_\pi^{\text{new}} \leftarrow \text{TeammateGenerator}(\text{Env}, \theta^{\text{ego}}, \text{B}_\pi)$ // Train teammates to maximize regret
5: $\quad \theta^{\text{ego}} \leftarrow \text{EgoUpdate}(\text{Env}, \theta^{\text{ego}}, \text{B}_\pi^{\text{new}})$ // Train ego agents to minimize regret
6: $\quad \text{B}_\pi \leftarrow \text{B}_\pi^{\text{new}}$
7: **end for**
8: **Output:** Final ego policy θ^{ego}

Alg. 1 outlines our general framework for training an ego agent to minimize the worst-case cooperative regret induced by any teammate $\pi^{-i} \in \Pi_K^{-i}(p_0)$. Like coordinate ascent algorithms [20], the framework alternates between optimizing for π^{-i} and π^{ego} for T^{iter} iterations, while holding the other fixed. We call a phase in which π^{ego} is fixed and π^{-i} is updated to maximize the ego agent's regret the **teammate generation phase** (Line 4). Meanwhile, assuming that π^{-i} is fixed, the **ego agent update phase** (Line 5) updates π^{ego} to minimize regret.

Sec. 4 & 5 describe ROTATE, our practical instantiation of Alg. 1 as an open-ended learning approach for AHT. Sec. 4 focuses on generating regret-maximizing teammate policies that remain competent. Meanwhile, Sec. 5 covers the practical experience collection process for ROTATE's teammate generation and ego agent update phases. ROTATE is an open-ended learning procedure according to the definition of Hughes et al. [25] because it continually generates *novel* yet *learnable* artifacts (i.e., teammates) for an observer (i.e., ego agent), as discussed in App. D.1.

4 Generating Regret-Maximizing Yet Competent Teammates

Given a fixed π^{ego}, we wish to generate teammate policies that maximize the cooperative regret of π^{ego}. However, naive regret maximization may lead to incompetent teammates that act adversarially towards the ego agent, introducing learning challenges. We propose a novel regret-based objective that addresses the problem in two steps. First, teammate competence is bounded on the initial

[3] Consider a one-step task in which two agents simultaneously retrieve one of two objects and succeed only if the objects differ. Against a teammate that always retrieves the first object and another that always retrieves the second, any ego policy has positive regret with at least one of the two teammates.

[4] Define the regret as $V(s_0|\pi^{-i}, \pi^{\text{tm}}) - V(s_0|\pi^{-i}, \pi^{\text{ego}})$, where $(\pi^{-i}, \pi^{\text{tm}})$ jointly maximize regret while π^{ego} minimizes it. To maximize regret, π^{tm} can only act on the first term to maximize the value with π^{-i}, recovering our original minimax regret formulation (cf. Thm. 2 of Dennis et al. [19]). Under this view, π^{tm} need not be the exact best response for the objective to be well posed.

state distribution. Second, because a teammate that is competent from p_0 may still be incompetent at the states into which it steers the ego agent, we define *sabotage* as a property of a teammate generation objective and show that bounding competence from p_0 does not remove the incentive to sabotage. Optimizing competence-bounded regret from all states visited during training then yields the ROTATE teammate generation objective, which bounds teammate competence on those states. The presented theoretical results are proven in App. B.2.

4.1 Ensuring Teammate Competence

The most direct instantiation of the regret maximization component of Eq. 5 as an RL objective is the following **per-trajectory regret** objective:

$$\max_{\pi^{-i}} \mathbb{E}_{s_0 \sim p_0} \left[\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0) \right]. \tag{6}$$

Eq. 6 is the regret over trajectories starting from the initial state distribution. The per-trajectory regret objective corresponds directly to the adversarial diversity objectives previously proposed in the teammate generation [49, 11] and UED literature [68, 19].

However, maximizing per-trajectory regret alone does not ensure that the generated teammate is competent. In App. B.1, a matrix game example shows that a regret-maximizing teammate can minimize returns against any π^{ego}. Prior AHT work has observed that adversarial diversity objectives frequently yield teammates that minimize their partners' returns [18, 54, 12, 32], and has proposed modifications to the objective to counteract this behavior. Because Π^{eval} is assumed to consist of p_0-competent teammates, training against incompetent teammates exposes the ego agent to teammates unlikely to be encountered during evaluation. Furthermore, because an incompetent teammate yields low returns regardless of the ego agent's actions, training against such a teammate provides little signal about the coordination behaviors that competent teammates reward, so the ego agent may fail to cooperate well even with the teammates that it does encounter at evaluation time.

To ensure that maximizing regret leads to p_0-competent teammates, we augment per-trajectory regret with the λ-weighted return of the teammate and its best response, yielding the **competence-bounded per-trajectory regret** objective:

$$\max_{\pi^{-i}} \mathbb{E}_{s_0 \sim p_0} \left[\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0) + \lambda V\left(s_0 | \pi^{-i}, \text{BR}(\pi^{-i})\right) \right]. \tag{7}$$

The following theorem characterizes the competence that each value of λ enforces on any π^{-i} that maximizes Eq. 7.[5] Choosing λ greater than the provided constant yields teammates that are p_0-competent in the same sense assumed of Π^{eval}.

Theorem 1 (Per-Trajectory Competence Lower Bound)**.** *Let* $\lambda \in \mathbb{R}^{>0}$*, and let* $R^{\max}_{p_0} = \max_{\pi'} \mathbb{E}_{s_0 \sim p_0} \left[V(s_0 | \pi', BR(\pi')) \right]$ *and* $R^{\min}_{p_0} = \min_{\pi'} \mathbb{E}_{s_0 \sim p_0} \left[V(s_0 | \pi', \pi^{ego}) \right]$*. If* π^{-i} *maximizes Eq. 7, then* $\mathbb{E}_{s_0 \sim p_0} \left[V\left(s_0 | \pi^{-i}, BR(\pi^{-i})\right) \right] \geq \frac{\lambda}{\lambda+1} R^{\max}_{p_0} + \frac{1}{\lambda+1} R^{\min}_{p_0}$*. Consequently, for any* $K \in [R^{\min}_{p_0}, R^{\max}_{p_0})$*, choosing* $\lambda \geq \frac{K - R^{\min}_{p_0}}{R^{\max}_{p_0} - K}$ *guarantees* $\pi^{-i} \in \Pi^{-i}_K(p_0)$.

4.2 Ensuring Teammate Competence on a Per-State Basis

An objective incentivizes **sabotage** if a teammate can increase the objective by becoming *incompetent* starting from states visited during XP interactions between π^{-i} and π^{ego}, which lowers the return that *any* partner can obtain from those states. Competence-bounded per-trajectory regret alone (Eq. 7) does not mitigate sabotage because p_0-competence constrains only the return that π^{-i} achieves with its BR from p_0. Thus, incompetence at states visited only with π^{ego} raises regret without lowering p_0-competence. For example, a teammate that acts competently only at states reached with its BR and incompetently everywhere else is p_0-competent and achieves high per-trajectory regret, yet obtains low return with any π^{ego} that ever strays from the BR's states.

Teammates generated by an objective that incentivizes sabotage are undesirable for two reasons. First, such teammates are not expected to be encountered in AHT, as the teammates in Π^{eval} are assumed to act in good faith towards the shared task. Second, they introduce learning challenges (see Sec. 4.1).

[5] Eq. 7 is the Lagrangian of the problem of maximizing per-trajectory regret subject to the constraint $\pi^{-i} \in \Pi^{-i}_K(p_0)$, with λ held fixed. Primal-dual methods for single-agent constrained RL rely on the strong duality of constrained MDPs [47], which relies on the linearity of the objective in the occupancy measure induced by the learning agent. Here, the objective involves the BR value $\max_{\pi'} V(s_0 | \pi^{-i}, \pi')$, which is not linear in the occupancy measure of π^{-i}. Thus, strong duality does not follow from Paternain et al. [47]. Theorem 1 instead directly characterizes the competence attained with a fixed λ.

This problem is not encountered by typical UED approaches [19, 27, 52], which generate only an environment's initial state under fixed transition dynamics.

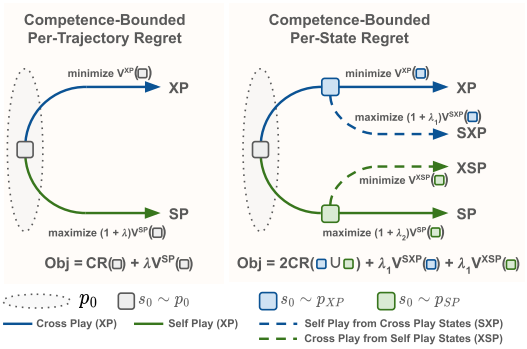


Figure 2: Teammate policy optimization objectives.

Sabotage is defined as a property of an objective rather than of a policy, as sabotage suggests deliberate intent, which is encoded by the objective. For example, the same incompetent behavior may arise from two objectives that represent very different incentives. Consider a robot chef that cooks competently in a single-width corridor kitchen but performs a safety stop for the remainder of the episode if a partner draws too close. The robot is p_0-competent, since its BR keeps clear and never triggers the stop, but per-state incompetent once the stop triggers, since no partner can complete a dish from a blocked kitchen. The robot could learn to stop as a side effect of a safety-prioritizing objective, or because halting on states reached only with π^{ego} raises the competence-bounded per-trajectory regret of π^{ego} at no cost to p_0-competence. Characterizing sabotage as a behavioral property fails to capture the difference between these two learning scenarios.

Prior work characterizes sabotage as return-minimizing behavior of a teammate when interacting with partners other than its BR [18, 54, 12]. Our definition differs in two respects. First, sabotage is defined as a property of an objective rather than of a policy, as argued above. Second, sabotage is defined in terms of per-state incompetence rather than return minimization, so that the return is evaluated with respect to the *best possible partner* from the affected states, rather than with the non-BR partner that a teammate is currently interacting with.[6] In fact, given that the competence constraints on teammate behavior are active, it is *desirable* for the teammate to minimize the return of π^{ego} because it indicates that the teammate has learned a convention that the ego agent does not know. The self-sabotage definition of Lauffer et al. [32] likewise defines sabotage as a property of an objective, but constrains the teammate differently. It requires the teammate to be a BR to some partner from p_0 only, whereas per-state competence requires that the teammate can achieve return above a threshold with some partner from every state the ego agent encounters. Neither condition implies the other, since a teammate may best-respond to a partner from p_0 while sabotaging states that partner never visits, and a competent teammate need not be optimal for any partner.

We mitigate the incentive to sabotage by training π^{-i} to maximize regret *while remaining competent* starting from the distribution of *all* states visited by the teammate during training. In particular, regret estimation involves the SP and XP state visitation distributions. Let $d(\pi^1, \pi^2; p)$ denote the state visitation distribution when π^1 and π^2 interact based on a starting state distribution p. The SP and XP state visitation distributions are denoted by the shorthand $p_{\text{SP}}(\pi^{-i}) := d(\pi^{-i}, \text{BR}(\pi^{-i}); p_0)$ and $p_{\text{XP}}(\pi^{-i}) := d(\pi^{-i}, \pi^{\text{ego}}; p_0)$. We write p_{SP} and p_{XP} for $p_{\text{SP}}(\pi^{-i})$ and $p_{\text{XP}}(\pi^{-i})$ when the teammate is clear from context. Based on p_{SP} and p_{XP}, the **per-state regret** of the ego agent is defined as the regret $\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0)$ with s_0 drawn from $\mathcal{U}(p_{\text{XP}}, p_{\text{SP}})$, i.e., from p_{XP} or p_{SP} with equal probability.

Analogous to Sec. 4.1, per-state regret is augmented with the λ-weighted return of the teammate and its BR to obtain the final ROTATE teammate generation objective, which we refer to as **competence-bounded per-state regret**. Here, $\text{BR}(\pi^{-i})$ is the best response to π^{-i} at every visited state rather than only from p_0, so that $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ measures the best return that some partner can achieve with π^{-i} from arbitrary start state distributions.

$$
\begin{aligned}
J(\pi^{-i}, \pi^{\text{ego}}) = {} & 2\mathbb{E}_{s_0 \sim \mathcal{U}(p_{\text{XP}},\, p_{\text{SP}})} \left[\text{CR}\left(\pi^{\text{ego}}, \pi^{-i}, s_0\right)\right] \\
& + \lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}} \left[V\left(s_0 | \pi^{-i}, \text{BR}(\pi^{-i})\right)\right] \\
& + \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}} \left[V\left(s_0 | \pi^{-i}, \text{BR}(\pi^{-i})\right)\right]. \quad (8)
\end{aligned}
$$

Optimizing competence-bounded per-state regret (Eq. 8) offers two intuitive benefits. First, it yields a π^{-i} with which π^{ego} can improve its action selection across a broader range of states. Second,

[6] Note that the behaviors identified as sabotage in prior work, such as deliberately playing unplayable cards in Hanabi [18], lower the best possible return from the affected states and are therefore also per-state incompetent.

maximizing regret from SP states disincentivizes signals for unconditional cooperation (or *handshake conventions* [54]) by requiring π^{-i} to decrease the ego agent's return starting from any SP state.

The following theorem bounds the competence of any π^{-i} that maximizes Eq. 8.

Theorem 2 (Per-State Competence Lower Bound)**.** *Let $\lambda_1, \lambda_2 \in \mathbb{R}^{\geq 0}$ with $\Lambda := \lambda_1 + \lambda_2 > 0$. For any teammate π', let $p_{XP}(\pi')$ and $p_{SP}(\pi')$ denote the XP and SP state visitation distributions induced by π', and define the mixtures $U(\pi') := \mathcal{U}(p_{XP}(\pi'), p_{SP}(\pi'))$, $\mu(\pi') := \frac{\lambda_1}{\Lambda} p_{XP}(\pi') + \frac{\lambda_2}{\Lambda} p_{SP}(\pi')$, and $\nu(\pi') := \frac{\lambda_1+1}{\Lambda+2} p_{XP}(\pi') + \frac{\lambda_2+1}{\Lambda+2} p_{SP}(\pi')$. Let $R_\mu^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim \mu(\pi')} [V(s_0|\pi', BR(\pi'))]$ and $R_U^{\min} = \min_{\pi'} \mathbb{E}_{s_0 \sim U(\pi')} [V(s_0|\pi', \pi^{ego})]$. If π^{-i} maximizes Eq. 8, then*

$$\mathbb{E}_{s_0 \sim \nu(\pi^{-i})} \left[V\left(s_0 | \pi^{-i}, BR(\pi^{-i})\right) \right] \geq \frac{\Lambda}{\Lambda + 2} R_\mu^{\max} + \frac{2}{\Lambda + 2} R_U^{\min}.$$

$R_U^{\min}$ and $R_\mu^{\max}$ are teammate-independent, fixed quantities. Theorem 2 guarantees a minimum level of competence for any teammate that maximizes the ROTATE objective in Eq. 8. As the total competence weight increases, if the ratio between λ_1 and λ_2 is fixed, the bound increases monotonically from $R_U^{\min}$ towards $R_\mu^{\max}$. Thus, the competence weights control the strength of the guarantee. Unlike Theorem 1, Theorem 2 certifies competence with respect to $\nu(\pi^{-i})$, the states encountered during ego training, rather than to p_0. Thus, compared to Eq. 7, Eq. 8 trades alignment with the definition of Π^{eval} [58] for cooperativeness on the states over which the ego agent learns.

5 Practical Algorithm: ROTATE

This section presents ROTATE, our practical algorithm that instantiates Alg. 1 by specifying the `TeammateGenerator` and `EgoUpdate` procedures. In ROTATE, all policies are estimated using Proximal Policy Optimization (PPO) [56]. See App. F for a detailed description and pseudocode.

ROTATE Teammate Generator Optimizing Eq. 8 requires a different data collection process compared to the usual RL methods. While maximizing $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ over p_{SP} is straightforward, maximizing the other terms from Eq. 8 requires collecting experiences using either environment resetting or policy switching. Taking the maximization of $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ from $s_0 \sim p_{\text{XP}}$ as an example, if an environment supports resetting to arbitrary states, then states encountered during XP interaction can be stored and used as the initial state for SP interactions. We call such data "self-play from cross-play" (SXP) data. Otherwise, we may sample a random timestep t, run XP interaction until timestep t, and then switch to SP interaction [54]. Only data gathered after timestep t is used to optimize the expected value of $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ over p_{XP}. The same SXP data is used to train $\text{BR}(\pi^{-i})$, so that it is a best response to π^{-i} at every state the teammate is observed in, as required in Sec. 4.2. Both data collection schemes assume it is possible for agent policies to act starting from arbitrary points within an interaction. While the assumption is straightforwardly satisfied by fully observable settings with Markov states, it may not be in partially observable settings, in which teammates typically require memory. ROTATE extends to this setting by conditioning non-recurrent agent policies on a shared recurrent belief state (App. C.3).

ROTATE Ego Agent Update At each iteration, ROTATE creates a teammate that highlights the cooperative weaknesses of the previous iteration's ego agent by maximizing the ego agent's per-state regret. To allow the ego agent to improve its robustness over time and reduce the possibility that it forgets how to cope with previously generated teammates, ROTATE maintains a *population buffer* of generated teammates. At each iteration, the ego agent is trained with teammates sampled uniformly from the population buffer. While prior AHT methods that train against fixed teammate populations have proposed worst-case teammate sampling strategies [21, 66], worst-case sampling risks concentrating ego agent training on teammates with irreducible regret (e.g., due to convention conflict), causing ego learning to stagnate [6]. Uniform sampling does not remove adversarial pressure from ego training, as the teammate generator can regenerate any convention in the population buffer that the ego agent is unable to coordinate with. Combining the ROTATE teammate generator with adversarial buffer curation is left for future work (Sec. 7).

6 Experimental Results

This section presents the empirical evaluation of ROTATE compared to baseline methods, as well as several ablations. Code and supplemental results are in the Appendix. The main research questions are:

- **RQ1**: Does ROTATE generalize better to unseen teammates than AHT and UED baselines? (Yes.)
- **RQ2**: Does the ROTATE teammate generation objective mitigate the emergence of teammates that are difficult to learn with? (Yes.)
- **RQ3**: What is the influence of the competence weights, λ_1, λ_2, on generalization performance?

6.1 Experimental Setup

This section describes the tasks, baselines, evaluation set construction, and evaluation metric.

Tasks. ROTATE is evaluated on a didactic matrix game and six benchmark tasks. The benchmark tasks are Level-Based Foraging (LBF) [2] and the five classic layouts from the Overcooked suite [10]: Cramped Room (CR), Asymmetric Advantages (AA), Counter Circuit (CC), Coordination Ring (CoR), and Forced Coordination (FC). All tasks support multiple conventions, and are commonly used in the AHT literature [2, 14, 46]. In LBF, two agents cooperate to collect randomly placed apples in a gridworld. In Overcooked, two agents collaborate in kitchen gridworlds to prepare dishes.

Baselines. As our method is most closely related to methods from UED and teammate generation, we compare against two UED methods adapted for AHT (PAIRED [19], Minimax Return [42]), three AHT algorithms focused on teammate generation (Fictitious Co-Play [59], BRDiv [49], CoMeDi [54]), and an open-ended AHT method based on an empirical game-theoretic framework (COLE [33]). Curator-based UED methods such as PLR [26, 27] are excluded, as they are orthogonal to ours [21, 66, 13]. Similarly, we do not compare against AHT algorithms for ego learning [3]. Fuller descriptions of each baseline, including how the UED baselines are adapted for AHT, are provided in App. G. For fair comparison, all methods received manual and automated hyperparameter tuning and were trained for the same number of environment interactions (App. I.4).

Construction of Π^{eval}. We wish to evaluate all methods on as diverse a set of evaluation teammates as feasible, while ensuring that each teammate is competent. To achieve this goal, for each task, we construct 9 to 13 evaluation teammates using three strategies: independent PPO (IPPO) [74] with varied seeds and reward shaping, BRDiv, and manually programmed heuristic agents. The evaluation set sizes and sources are comparable to those of prior AHT work [67, 50, 11]. Candidate teammates that added no diversity, as measured by the cross-play matrix among evaluation teammates, were pruned (fig. 20). App. J fully describes the generation procedure and the final teammates in Π^{eval}.

Evaluation Metrics. Ego agent policies are evaluated with each teammate in Π^{eval} for 64 evaluation episodes, where the episode return is normalized with a zero lower bound and an estimated best-response upper bound for each teammate. The upper bounds are obtained by training best-response policies via RL, then revising each bound upward whenever any evaluated partner policy exceeds it. Performance of a method on Π^{eval} is primarily reported as the normalized mean return with bootstrapped 95% confidence intervals across six trials for the core comparison (Fig. 3 (left)) and three trials for all other experiments, computed via the `rliable` library [1]. The normalized return metric is a member of the BRProx metric family recommended by Wang et al. [69]. Following Agarwal et al. [1], 95% bootstrapped confidence interval estimates are used to summarize comparisons rather than significance tests. Because changing the evaluation teammate changes the transition and reward dynamics faced by the ego agent, each evaluation teammate is treated as a distinct evaluation task, and the bootstrap resamples training seeds within each evaluation task. Details about the normalization procedure and specific bounds for each teammate are reported in App. J.1.

6.2 Results

This section addresses the research questions introduced at the beginning of Sec. 6. Extensive supplemental analysis is provided in the appendix, most of which is highlighted within this section. Among the analyses that are not highlighted, App. C.7 considers ablations of ROTATE in which the teammate generation objective is replaced by competence-bounded per-trajectory regret or a mixed-play variant, while App. C.8 ablates the population buffer and analyzes the independent utility of the ROTATE population. Finally, App. C.11 provides learning curves for all variants of ROTATE.

RQ1: Does ROTATE generalize better to unseen teammates than baselines? (Yes.) Fig. 3 (left) compares the normalized mean returns for ROTATE and baseline methods on Π^{eval} across the six tasks. ROTATE attains the highest mean normalized return on five of the six tasks, with the sole exception of AA, where Minimax Return performs best.

CoMeDi and FCP are particularly strong baselines. We attribute CoMeDi's strong performance to the resemblance of its mixed-play objective to our per-state regret objective, which we discuss in

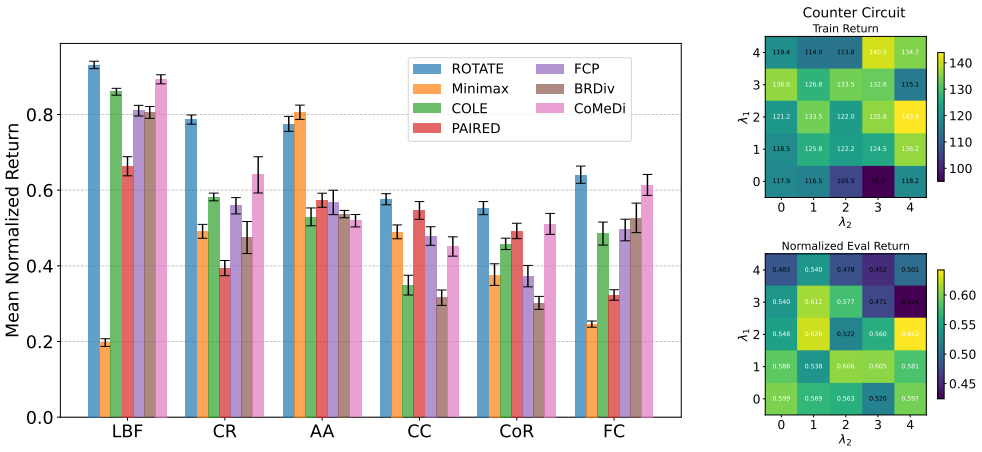


Figure 3: **(Left)** ROTATE achieves the highest normalized evaluation return on five of six tasks compared to baselines. **(Right)** Higher competence weights (λ_1, λ_2) lead to higher training returns, reflecting the increased cooperativity of training teammates. However, intermediate competence weights correspond to the highest evaluation returns, demonstrating that the most cooperative teammates are not necessarily the best training partners for learning generalizable cooperative behaviors.

App. D.2. Meanwhile, FCP tends to perform especially well with the IPPO policies in Π^{eval}, likely because the IPPO evaluation teammates are in-distribution relative to the teammates constructed by FCP. Likewise, on CoR and FC, where CoMeDi performs similarly to ROTATE, CoMeDi's returns are driven by high performance with the IPPO evaluation teammates.

Minimax Return's strong performance in AA can be attributed to AA's separated kitchen halves, which give each agent all resources needed for individual task completion. Thus, fully independent agents may achieve high returns, although coordination yields still higher returns (App. H.1). By contrast, on LBF and FC, where coordination is crucial to obtain positive returns, Minimax Return is the worst-performing baseline.

BRDiv and PAIRED exhibit comparatively worse performance, which may be partially attributed to their teammate generation objectives that resemble per-trajectory regret. Furthermore, PAIRED's update structure involves a lockstep training process for the teammate generator, best response, and ego agent, in contrast to ROTATE, which trains a new teammate-BR pair for each iteration. This synchronized training may hinder the emergence of diverse conventions crucial for AHT.

Supplemental analyses in the appendix further support the main comparison. App. C.1 shows that ROTATE attains the highest interquartile mean, median, and probability of improvement on the five tasks identified above, even under unnormalized or worst-case returns. The per-teammate breakdown in App. Fig. 7 further shows that ROTATE has more robust and well-rounded cooperative capabilities compared to baselines, improving more evenly across evaluation teammates. App. C.2–C.4 extend the evaluation to a human proxy study on Overcooked and to N-agent and partially observable scenarios, in which ROTATE again matches or outperforms baselines. App. C.9 finds interpretable conventions, no handshake sabotage, largely competent teammates, and reasonable induced regret in the ROTATE populations on LBF and CC. App. C.10 finds that the co-trained BRs show some approximation error but no systematic under- or over-estimation.

RQ2: Does the ROTATE teammate generation objective mitigate the emergence of teammates that are difficult to learn with? (Yes.) We design a *sabotage game* to investigate whether the proposed objective leads to teammate policies that are easier to learn to interact with compared to the per-trajectory regret objective. The sabotage game is a fully cooperative, iterated matrix game lasting five timesteps, with payoff matrix in Fig. 4 (top). Each agent observes a game state consisting of the complete history of joint actions, and chooses among three actions: H, T, and S(abotage). H and T lead to two cooperative outcomes, while S leads to a reward of -1 and immediate episode termination if selected by either agent. Thus, choosing sabotage corresponds to a grim trigger strategy, in which an agent acts to minimize the team's payoffs regardless of the other agent's actions.

We compare ROTATE with a variant in which teammates optimize per-trajectory regret. To measure the extent to which the learned teammate policies engage in sabotage, the probability of the sabotage

action is measured at each of the game's 341 non-terminal states for the last generated teammate policy. Fig. 4 (bottom) shows that ROTATE generates a teammate policy with a near-zero probability of taking the sabotage action at all non-terminal states, while the per-trajectory regret objective yields P(S) near 1.0 in over a third of states. Learning useful collaboration strategies with these grim trigger teammates is difficult, since the ego agent rarely receives positive rewards. In App. C.5, competence weights are swept for both per-state and per-trajectory regret variants on the sabotage game, showing that applying competence bounds to per-state regret mitigates the emergence of sabotage strategies more effectively than applying the same bound to per-trajectory regret.

RQ3: What is the influence of the competence weights, λ_1, λ_2, on generalization performance? Sec. 4.2 shows theoretically that maximizing the sum of per-state regret and $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ provides a lower bound on the competence of the teammate, where λ_1, λ_2 are the weights of the return from XP and SP states, respectively. Fig. 3 (right) compares the training and evaluation returns of ROTATE as λ_1 and λ_2 are varied on CC. The training returns are the mean returns of the final ROTATE ego agent when paired with the population of generated teammates, while the evaluation returns are its mean returns with the evaluation teammates. Results for all tasks are presented in App. C.6. As λ_1 and λ_2 are increased, the training returns increase, which reflects the increased cooperativeness of the generated teammates. On the other hand, the influence of the competence weights on the evaluation returns—which reflect the ability of the ego agent to generalize its cooperative behavior—is task-dependent. On CC, intermediate values of λ_1 and λ_2 lead to the best evaluation returns, demonstrating that teammates that are overly cooperative with the ego agent are not ideal for learning generalizable cooperative behavior. However, on LBF, increased competence weights improve both training and evaluation returns.

	H	*T*	*S*
H	1	0	-1
T	0	1	-1
S	-1	-1	-1

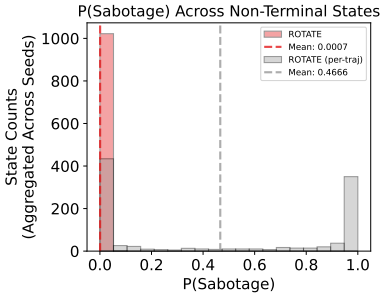


Figure 4: **(Top)** Payoff matrix for the sabotage game. **(Bottom)** Probability of the sabotage action at all non-terminal states in the sabotage game for ROTATE teammates vs those trained with per-trajectory regret. Results are aggregated across three trials.

7 Discussion and Conclusion

This paper reformulates AHT as an open-ended learning problem and introduces ROTATE, a regret-driven algorithm. ROTATE iteratively alternates between improving an AHT agent and generating challenging yet cooperative teammates. The teammates are generated by optimizing a regret-based objective designed to discover competent teammates that exploit cooperative vulnerabilities yet remain learnable. To this end, sabotage is characterized as a property of an objective that induces incompetent teammates, and the proposed objective is shown to bound teammate competence on the states encountered during AHT agent training. Experiments on an illustrative matrix game demonstrate that the proposed objective mitigates sabotage. Extensive evaluations across six benchmark tasks demonstrate that ROTATE significantly improves AHT agent generalization to previously unseen teammates, outperforming baselines from both AHT and UED.

The current work has three limitations that suggest future work. First, the theoretical results show that mixing regret and return maximization bounds the competence of generated teammates, but do not prove that bounding competence on teammate-ego interactions is beneficial for ego agent learning. Second, each open-ended iteration trains the teammate and best response from scratch, which is computationally expensive and may bias regret estimates due to approximation error, motivating amortized training schemes. Third, this work focuses on the teammate generation phase of open-ended AHT. Future work might explore both ego agent training methods that better handle the nonstationarity induced by open-ended teammate generation, e.g., via continual learning techniques, and alternative methods to sample teammates from the population buffer. Compared to uniform sampling, a worst-case sampler over the buffer [9, 66] could improve worst-case robustness, but coupling it with the already-adversarial ROTATE teammate generator ties the ego agent's training distribution to its own current weaknesses, risking strategic cycling and catastrophic forgetting. Finally, previously generated teammates and ego agents could be treated as an empirical meta-game, using meta-strategy solvers to compute both agents' training distributions [30].

Acknowledgments and Disclosure of Funding

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (FAIN-2019844, NRT-2125858, STC-2535195), ONR (N00014-24-1-2550), ARO (W911NF-17-2-0181, W911NF-23-2-0004, W911NF-25-1-0065), Lockheed Martin, Lyda Hill, and Good Systems. Peter Stone serves as the Chief Scientist of Sony AI and receives financial compensation for that role. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

[2] Stefano V. Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, AAMAS '13, pages 1155–1156, Richland, SC, May 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-1993-5.

[3] Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2018.01.002. URL https://www.sciencedirect.com/science/article/pii/S0004370218300249.

[4] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent Tool Use From Multi-Agent Autocurricula. In *International Conference on Learning Representations*, September 2019. URL https://openreview.net/forum?id=SkxpxJBKwS.

[5] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4): 819–840, 2002.

[6] Michael Beukman, Samuel Coward, Michael Matthews, Mattie Fellows, Minqi Jiang, Michael Dennis, and Jakob Foerster. Refining minimax regret for unsupervised environment design. In *International Conference on Machine Learning*, 2024.

[7] Clément Bonnet, Daniel Luo, Donal John Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence Illing Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma Mahjoub, Matthew Macfarlane, Andries Petrus Smit, Nathan Grinsztajn, Raphael Boige, Cemlyn Neil Waters, Mohamed Ali Ali Mimouni, Ulrich Armel Mbou Sob, Ruan John de Kock, Siddarth Singh, Daniel Furelos-Blanco, Victor Le, Arnu Pretorius, and Alexandre Laterre. Jumanji: a Diverse Suite of Scalable Reinforcement Learning Environments in JAX. In *The Twelfth International Conference on Learning Representations*, October 2023. URL https://openreview.net/forum?id=C4CxQmp9wc.

[8] Michael Bowling and Peter McCracken. Coordination and adaptation in impromptu teams. In *AAAI*, volume 5, pages 53–58, 2005.

[9] Thomas Kleine Buening, Christos Dimitrakakis, Hannes Eriksson, Divya Grover, and Emilio Jorge. Minimax-bayes reinforcement learning. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 7511–7527. PMLR, April 2023.

[10] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the Utility of Learning about Humans for Human-AI Coordination. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/f5b1b89d98b7286673128a5fb112cb9a-Abstract.html.

[11] Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. Generating diverse cooperative agents by learning incompatible policies. In *The Eleventh International Conference on*

Learning Representations, 2023. URL https://openreview.net/forum?id=UkU05GOH7_6.

[12] Rujikorn Charakorn, Poramate Manoonpong, and Nat Dilokthanakul. Diversity is not all you need: Training a robust cooperative agent needs specialist partners. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 56401–56423. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/66b35d2e8d524706f39cc21f5337b002-Paper-Conference.pdf.

[13] Paresh Chaudhary, Yancheng Liang, Daphne Chen, Simon S. Du, and Natasha Jaques. Improving human-ai coordination through adversarial training and generative models, 2025. URL https://arxiv.org/abs/2504.15457.

[14] Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[15] Filippos Christianos, Georgios Papoudakis, Muhammad A. Rahman, and Stefano V. Albrecht. Scaling Multi-Agent Reinforcement Learning with Selective Parameter Sharing. In *Proceedings of the 38th International Conference on Machine Learning*, pages 1989–1998. PMLR, July 2021. URL https://proceedings.mlr.press/v139/christianos21a.html.

[16] Daphne Cornelisse and Eugene Vinitsky. Human-compatible driving partners through data-regularized self-play reinforcement learning, June 2024. URL http://arxiv.org/abs/2403.19648.

[17] Brandon Cui, Hengyuan Hu, Luis Pineda, and Jakob Foerster. K-level reasoning for zero-shot coordination in hanabi. *Advances in Neural Information Processing Systems*, 34:8215–8228, 2021.

[18] Brandon Cui, Andrei Lupu, Samuel Sokota, Hengyuan Hu, David J Wu, and Jakob Nicolaus Foerster. Adversarial diversity in hanabi. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=uLE3WF3-H_5.

[19] 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. In *Advances in Neural Information Processing Systems*, volume 33, pages 13049–13061. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html.

[20] DA d'Esopo. A convex programming procedure. *Naval Research Logistics Quarterly*, 6(1): 33–42, 1959.

[21] Hannah Erlebach and Jonathan Cook. RACCOON: Regret-based adaptive curricula for cooperation. In *Coordination and Cooperation for Multi-Agent Reinforcement Learning Methods Workshop*, 2024. URL https://openreview.net/forum?id=jAH5JNY3Qd.

[22] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[23] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

[24] Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, Noam Brown, and Jakob Foerster. Off-belief learning. In *International Conference on Machine Learning*, pages 4369–4379. PMLR, 2021.

[25] Edward Hughes, Michael D. Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge Shi, Tom Schaul, and Tim Rocktäschel. Position: Open-Endedness is Essential for Artificial Superhuman Intelligence. In *Proceedings of the 41st International Conference on Machine Learning*, pages 20597–20616. PMLR, July 2024. URL https://proceedings.mlr.press/v235/hughes24a.html.

[26] Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34:1884–1897, 2021.

[27] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021.

[28] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference for Learning Representations*, San Diego, CA, 2015. doi: 10.48550/arXiv.1412.6980. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

[29] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/10.1073/pnas.1611835114. Publisher: Proceedings of the National Academy of Sciences.

[30] 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 *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[31] WB Langdon. Pfeiffer–a distributed open-ended evolutionary system. In *AISB*, volume 5, pages 7–13. Citeseer, 2005.

[32] Niklas Lauffer, Ameesh Shah, Micah Carroll, Sanjit A. Seshia, Stuart Russell, and Michael D. Dennis. Robust and Diverse Multi-Agent Learning via Rational Policy Gradient. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

[33] Yang Li, Shao Zhang, Jichen Sun, Yali Du, Ying Wen, Xinbing Wang, and Wei Pan. Cooperative Open-ended Learning Framework for Zero-Shot Coordination. In *Proceedings of the 40th International Conference on Machine Learning*, pages 20470–20484. PMLR, July 2023. URL https://proceedings.mlr.press/v202/li23au.html. ISSN: 2640-3498.

[34] Fanqi Lin, Shiyu Huang, Tim Pearce, Wenze Chen, and Wei-Wei Tu. TiZero: Mastering Multi-Agent Football with Curriculum Learning and Self-Play. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pages 67–76, Richland, SC, May 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-9432-1.

[35] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023.

[36] Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Nicolaus Foerster, Satinder Singh, and Feryal Behbahani. Structured State Space Models for In-Context Reinforcement Learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, November 2023. URL https://openreview.net/forum?id=4W9FVg1j6I&noteId=38Anv4M4TW.

[37] Keane Lucas and Ross E Allen. Any-play: An intrinsic augmentation for zero-shot coordination. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 853–861, 2022.

[38] Andrei Lupu, Brandon Cui, Hengyuan Hu, and Jakob Foerster. Trajectory diversity for zero-shot coordination. In *International conference on machine learning*, pages 7204–7213. PMLR, 2021.

[39] Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V Albrecht. A survey of ad hoc teamwork research. In *European conference on multi-agent systems*, pages 275–293. Springer, 2022.

[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236.

[41] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

[42] Jun Morimoto and Kenji Doya. Robust Reinforcement Learning. *Neural Comput.*, 17(2): 335–359, February 2005. ISSN 0899-7667. doi: 10.1162/0899766053011528. URL https://doi.org/10.1162/0899766053011528.

[43] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining and Knowledge Discovery*, 2(1):86–97, 2012. doi: https://doi.org/10.1002/widm.53. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.53.

[44] Hadi Nekoei, Akilesh Badrinaaraayanan, Aaron Courville, and Sarath Chandar. Continuous Coordination As a Realistic Scenario for Lifelong Learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8016–8024. PMLR, July 2021. URL https://proceedings.mlr.press/v139/nekoei21a.html. ISSN: 2640-3498.

[45] Hadi Nekoei, Xutong Zhao, Janarthanan Rajendran, Miao Liu, and Sarath Chandar. Towards Few-shot Coordination: Revisiting Ad-hoc Teamplay Challenge In the Game of Hanabi. In *Proceedings of The 2nd Conference on Lifelong Learning Agents*, pages 861–877. PMLR, November 2023. URL https://proceedings.mlr.press/v232/nekoei23b.html. ISSN: 2640-3498.

[46] Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

[47] Santiago Paternain, Luiz F. O. Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[48] Arrasy Rahman, Niklas Höpner, Filippos Christianos, and Stefano V. Albrecht. Towards Open Ad Hoc Teamwork Using Graph-based Policy Learning. In *Proceedings of the 38 th International Conference on Machine Learning*, volume 139. PMLR, June 2021.

[49] Arrasy Rahman, Elliot Fosong, Ignacio Carlucho, and Stefano V Albrecht. Generating teammates for training robust ad hoc teamwork agents via best-response diversity. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=l5BzfQhR0l.

[50] Muhammad Rahman, Jiaxun Cui, and Peter Stone. Minimum coverage sets for training robust ad hoc teamwork agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17523–17530, 2024.

[51] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21:1–51, 2020.

[52] Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Nicolaus Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=iEeiZlTbts.

[53] Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian S. de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu, and Jakob Foerster. JaxMARL: Multi-Agent RL Environments and Algorithms in JAX. In *Advances in Neural Information Processing Systems*, volume 37, pages 50925–50951, December 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5aee125f052c90e326dcf6f380df94f6-Abstract-Datasets_and_Benchmarks_Track.html.

[54] Bidipta Sarkar, Andy Shih, and Dorsa Sadigh. Diverse conventions for human-AI collaboration. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=MljeRycu9s.

[55] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015. URL https://api.semanticscholar.org/CorpusID:3075448.

[56] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017. URL https://api.semanticscholar.org/CorpusID:28695052.

[57] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[58] Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1504–1509, July 2010. doi: 10.1609/aaai.v24i1.7529. URL https://ojs.aaai.org/index.php/AAAI/article/view/7529.

[59] DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with Humans without Human Data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14502–14515, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/797134c3e42371bb4979a462eb2f042a-Paper.pdf.

[60] Mingfei Sun, Sam Devlin, Jacob Beck, Katja Hofmann, and Shimon Whiteson. Trust Region Bounds for Decentralized PPO Under Non-stationarity. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pages 5–13, Richland, SC, May 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-9432-1.

[61] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[62] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 1999.

[63] Tim Taylor. Evolutionary innovations and where to find them: Routes to open-ended evolution in natural and artificial systems. *Artificial life*, 25(2):207–224, 2019.

[64] Warren S. Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 17(4): 401–419, 1952. doi: 10.1007/BF02288916.

[65] Alexander Vezhnevets, Yuhuai Wu, Maria Eckstein, Rémi Leblond, and Joel Z Leibo. Options as responses: Grounding behavioural hierarchies in multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 9733–9742. PMLR, 2020.

[66] Victor Villin, Thomas Kleine Buening, and Christos Dimitrakakis. A minimax approach to ad hoc teamwork, 2025. URL https://arxiv.org/abs/2502.02377.

[67] Caroline Wang, Arrasy Rahman, Ishan Durugkar, Elad Liebman, and Peter Stone. N-agent ad hoc teamwork. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=q7TxGUWlhD.

[68] Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International conference on machine learning*, pages 9940–9951. PMLR, 2020.

[69] Xihuai Wang, Shao Zhang, Wenhao Zhang, Wentao Dong, Jingxiao Chen, Ying Wen, and Weinan Zhang. ZSC-Eval: An Evaluation Toolkit and Benchmark for Multi-agent Zero-shot Coordination. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024. URL https://openreview.net/forum?id=9aXjIBLwKc#discussion.

[70] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[71] Dong Xing, Qianhui Liu, Qian Zheng, and Gang Pan. Learning with generated teammates to achieve type-free ad-hoc teamwork. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 472–478. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/66. URL https://doi.org/10.24963/ijcai.2021/66. Main Track.

[72] Ke Xue, Yutong Wang, Cong Guan, Lei Yuan, Haobo Fu, Qiang Fu, Chao Qian, and Yang Yu. Heterogeneous Multiagent Zero-Shot Coordination by Coevolution. *IEEE Transactions on Evolutionary Computation*, 29(5):2229–2243, October 2025. ISSN 1941-0026. doi: 10.1109/TEVC.2024.3485177. URL https://ieeexplore.ieee.org/document/10730789.

[73] Xue Yan, Jiaxian Guo, Xingzhou Lou, Jun Wang, Haifeng Zhang, and Yali Du. An efficient end-to-end training approach for zero-shot human-AI coordination. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=6ePsuwXUwf.

[74] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*, volume 35, pages 24611–24624, December 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract-Datasets_and_Benchmarks.html.

[75] Lei Yuan, Lihe Li, Ziqian Zhang, Feng Chen, Tianyi Zhang, Cong Guan, Yang Yu, and Zhi-Hua Zhou. Learning to coordinate with anyone. In *Proceedings of the Fifth International Conference on Distributed Artificial Intelligence*, pages 1–9, 2023.

[76] Rui Zhao, Jinming Song, Yufeng Yuan, Haifeng Hu, Yang Gao, Yi Wu, Zhongqian Sun, and Wei Yang. Maximum entropy population-based training for zero-shot human-ai coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6145–6153, 2023.

[77] Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. *arXiv preprint arXiv:2101.03864*, 2021.

Appendix

Please find the code for this paper at https://github.com/carolinewang01/rotate/.

A Related Work

Training AHT Agents. Most AHT methods follow a two-stage design process, in which a fixed training set of teammate policies is generated, followed by AHT training. Given teammates from the training set, AHT methods [39] train an ego agent to model teammates [3] by first modeling important characteristics such as goals, beliefs, policies, and then estimating the best-response policy to these teammates based on the inferred characteristics. Recent AHT methods [48, 46, 77, 67] use neural networks trained using reinforcement learning [56, 41]. To further improve AHT training, several approaches learn a distribution for sampling teammate policies during training based on maximizing the worst-case returns [66] or regret [21, 13] of trained agents. While few, exceptions to the two-stage process include methods designed for continual AHT [44, 45, 75], methods that co-evolve populations of ego agents and teammates [72, 75], self-play based methods, which do not explicitly optimize for diversity [73, 16], and empirical game theoretic methods that optimize for cooperative diversity as a heuristic to induce generalization to unseen teammates [33].

Teammate Generation for AHT & ZSC. Recent work removes the need to predefine teammate policy sets by generating diverse teammates during or before agent training. Other-Play [23] creates symmetry-equivalent teammates while training the agent policy, while E3T [73] mixes the agent's current policy with a random policy to encourage diversity. FCP [59] trains teammates via repeated CMARL runs with different seeds, later improved by methods maximizing information-theoretic diversity objectives such as Jensen-Shannon divergence [38], mutual information [37], and entropy [71, 76]. More recent approaches [11, 50, 75] generate teammates that require distinct best-response strategies by maximizing adversarial diversity metrics, similar to ROTATE's cooperative regret. Unlike ROTATE, these methods (i) maximize regret between generated teammates rather than with the trained agent, (ii) fix the teammate set prior to training, and (iii) evaluate regret only at the initial state. Prior work characterizes such sabotage as a behavior of the trained teammate, which identifies that it is interacting in cross-play and then minimizes the team return [18, 54, 12]. By contrast, ROTATE adopts a more systematic training objective that we demonstrate leads to performance gains. Lauffer et al. [32] instead define *self-sabotage* as irrational behavior induced by an adversarial optimization problem, and remedy it by constraining the teammate to be a best response to some partner. ROTATE shares this objective-level view but requires competence from the states the ego agent encounters rather than optimality for some partner from the initial state (Sec. 4.2).

Open-Ended Learning (OEL). OEL [31, 63] studies algorithms that continually generate novel tasks to train generally capable agents [25, 4]. Many OEL approaches in RL take the form of unsupervised environment design (UED) [19], which improves generalization by designing or sampling new environments with varied initial states. Some methods directly train neural networks to propose environments that induce high regret in the agent [19], while others selectively sample curated tasks generated by procedural generators based on criteria such as expected return [68], TD-error [27], regret [26], or learnability [52]. In competitive MARL, OEL often produces new opponents through self-play [57, 34]. For AHT, MACOP [75] generates novel teammates via an adversarial diversity objective optimized with evolutionary methods and similar to the objectives studied by Charakorn et al. [11] and Rahman et al. [49]. Applying this objective only to the initial state can yield teammates that diminish returns in some states, introducing learning challenges for an AHT agent [12, 54, 32]. Prior work characterizes such sabotage as a behavior of the trained teammate, which identifies that it is interacting in cross-play and then minimizes the team return [18, 54, 12]. Lauffer et al. [32] instead define *self-sabotage* as irrational behavior induced by an adversarial optimization problem, and remedy it by constraining the teammate to be a best response to some partner. ROTATE shares this objective-level view but requires competence from the states the ego agent encounters rather than optimality for some partner from the initial state (Sec. 4.2). By contrast, ROTATE adopts a more systematic training objective that we demonstrate leads to performance gains.

B Theory

B.1 Illustrative Example

In this section, we present a simple matrix game to illustrate that a teammate that maximizes regret is not guaranteed to be competent. In fact, it can even be the case that a teammate that maximizes regret also *minimizes* return with all ego agents.

Table 1: Simple matrix game where maximizing regret minimizes return.

	X	Y
A	5	1
B	5	0

For the payoff matrix in Table 1, the row player is the ego agent and the column player is the teammate agent, so the ego agent has actions $\{A, B\}$ and the teammate has actions $\{X, Y\}$.

If the teammate maximizes cooperative regret

$$CR(\pi^{\text{ego}}, \pi^{-i}) = V(\pi^{-i}, BR(\pi^{-i})) - V(\pi^{-i}, \pi^{\text{ego}}),$$

then the teammate will actually minimize return with all ego agents.

To see this, suppose π^{ego} plays A with probability p and B with probability $1 - p$. If the teammate plays X, then

$$CR(\pi^{\text{ego}}, X) = V(X, BR(X)) - V(X, \pi^{\text{ego}}) = 5 - (p * 5 + (1 - p) * 5) = 5 - 5 = 0, \quad (9)$$

whereas if the teammate plays Y, then

$$CR(\pi^{\text{ego}}, Y) = V(Y, BR(Y)) - V(Y, \pi^{\text{ego}}) = 1 - (p * 1 + (1 - p) * 0) = 1 - p. \quad (10)$$

Thus $CR(\pi^{\text{ego}}, X) \leq CR(\pi^{\text{ego}}, Y)$ for all π^{ego}, so the teammate that plays Y maximizes regret with all ego agents. However, playing Y also clearly minimizes return with all ego agents.

B.2 Proofs

This section proves the two competence bounds stated in Section 4. Theorem 1 bounds the competence of a teammate maximizing competence-bounded per-trajectory regret (Eq. 7), and Theorem 2 bounds the competence of a teammate maximizing the ROTATE objective (Eq. 8). Each theorem is preceded by a proposition relating the regret and return components of its objective.

Throughout, π^{ego} is fixed. For any teammate policy π', we write $V^{\text{SP}}_{\pi'}(s_0) := V\left(s_0 | \pi', \text{BR}(\pi')\right)$ for the return of π' with its best response and $V^{\text{XP}}_{\pi'}(s_0) := V\left(s_0 | \pi', \pi^{\text{ego}}\right)$ for its return with the ego agent. As mentioned in Sec. 4.2, $\text{BR}(\pi')$ denotes the best response to π' at every state, so $V^{\text{SP}}_{\pi'}(s_0)$ is the best return that some partner can achieve with π' from s_0, whether or not s_0 is reachable under self-play from p_0.

B.2.1 Competence-Bounded Per-Trajectory Regret

Recall the competence-bounded per-trajectory objective of Sec. 4.1,

$$\max_{\pi^{-i}} \mathbb{E}_{s_0 \sim p_0}\left[\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0) + \lambda V\left(s_0 | \pi^{-i}, \text{BR}(\pi^{-i})\right)\right]. \quad (7)$$

Proposition 1. *Let $\lambda \in \mathbb{R}^{>0}$. If the teammate π^{-i} maximizes Eq. 7, then*

$$\mathbb{E}_{s_0 \sim p_0}\left[CR\left(\pi^{ego}, \pi^{-i}, s_0\right)\right] \geq \lambda\left(R^{\max}_{p_0} - \mathbb{E}_{s_0 \sim p_0}\left[V^{SP}_{\pi^{-i}}(s_0)\right]\right),$$

where $R^{\max}_{p_0} = \max_{\pi'} \mathbb{E}_{s_0 \sim p_0}\left[V^{SP}_{\pi'}(s_0)\right]$.

Proof. Because cooperative regret is always non-negative, we get

$$\max_{\pi'} \mathbb{E}_{s_0 \sim p_0}\left[CR\left(\pi^{\text{ego}}, \pi', s_0\right) + \lambda V^{\text{SP}}_{\pi'}(s_0)\right] \geq \max_{\pi'} \mathbb{E}_{s_0 \sim p_0}\left[\lambda V^{\text{SP}}_{\pi'}(s_0)\right] = \lambda R^{\max}_{p_0}.$$

Because π^{-i} maximizes the left-hand side,

$$\mathbb{E}_{s_0 \sim p_0}\left[CR\left(\pi^{\text{ego}}, \pi^{-i}, s_0\right)\right] + \lambda \mathbb{E}_{s_0 \sim p_0}\left[V^{\text{SP}}_{\pi^{-i}}(s_0)\right] \geq \lambda R^{\max}_{p_0},$$

and rearranging shows that the proposition holds. □

This proposition states that cooperative regret, which is the gap between the SP value and the XP value, must be larger than λ times the gap between the best possible value and the SP value. For larger λ, in order for the inequality to hold, either the cooperative regret has to be larger or the teammate must be more competent (so as to achieve returns closer to $R_{p_0}^{\max}$). We leverage this proposition along with the recognition that cooperative regret may be bounded above to prove the following theorem:

Theorem 1 (Per-Trajectory Competence Lower Bound)**.** *Let* $\lambda \in \mathbb{R}^{>0}$*, and let* $R_{p_0}^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim p_0}\left[V(s_0|\pi', BR(\pi'))\right]$ *and* $R_{p_0}^{\min} = \min_{\pi'} \mathbb{E}_{s_0 \sim p_0}\left[V(s_0|\pi', \pi^{ego})\right]$*. If* π^{-i} *maximizes Eq. 7, then* $\mathbb{E}_{s_0 \sim p_0}\left[V\left(s_0|\pi^{-i}, BR(\pi^{-i})\right)\right] \geq \frac{\lambda}{\lambda+1} R_{p_0}^{\max} + \frac{1}{\lambda+1} R_{p_0}^{\min}$*. Consequently, for any* $K \in [R_{p_0}^{\min}, R_{p_0}^{\max})$*, choosing* $\lambda \geq \frac{K - R_{p_0}^{\min}}{R_{p_0}^{\max} - K}$ *guarantees* $\pi^{-i} \in \Pi_K^{-i}(p_0)$*.*

Proof. By Proposition 1 and the definition of cooperative regret,

$$\begin{aligned}
\mathbb{E}_{s_0 \sim p_0}\left[V_{\pi^{-i}}^{\text{SP}}(s_0)\right] &\geq R_{p_0}^{\max} - \frac{1}{\lambda}\mathbb{E}_{s_0 \sim p_0}\left[V_{\pi^{-i}}^{\text{SP}}(s_0) - V_{\pi^{-i}}^{\text{XP}}(s_0)\right] \\
&\geq R_{p_0}^{\max} - \frac{1}{\lambda}\mathbb{E}_{s_0 \sim p_0}\left[V_{\pi^{-i}}^{\text{SP}}(s_0)\right] + \frac{1}{\lambda} R_{p_0}^{\min},
\end{aligned}$$

where the second inequality holds because π^{-i} is one of the teammates over which the minimum defining $R_{p_0}^{\min}$ is taken. Collecting terms gives $\frac{\lambda+1}{\lambda}\mathbb{E}_{s_0 \sim p_0}\left[V_{\pi^{-i}}^{\text{SP}}(s_0)\right] \geq R_{p_0}^{\max} + \frac{1}{\lambda} R_{p_0}^{\min}$, and multiplying by $\frac{\lambda}{\lambda+1}$ yields the stated bound.

For the second claim, rewrite the right-hand side of the bound as

$$\frac{\lambda}{\lambda+1} R_{p_0}^{\max} + \frac{1}{\lambda+1} R_{p_0}^{\min} = R_{p_0}^{\min} + \frac{\lambda}{\lambda+1}\left(R_{p_0}^{\max} - R_{p_0}^{\min}\right).$$

Because $K < R_{p_0}^{\max}$, this quantity is at least K if and only if

$$\begin{aligned}
&\frac{\lambda}{\lambda+1} \geq \frac{K - R_{p_0}^{\min}}{R_{p_0}^{\max} - R_{p_0}^{\min}} \\
\iff &\lambda\left(R_{p_0}^{\max} - R_{p_0}^{\min}\right) \geq (\lambda+1)\left(K - R_{p_0}^{\min}\right) \\
\iff &\lambda\left(R_{p_0}^{\max} - K\right) \geq K - R_{p_0}^{\min},
\end{aligned}$$

and dividing by $R_{p_0}^{\max} - K > 0$ gives $\lambda \geq \frac{K - R_{p_0}^{\min}}{R_{p_0}^{\max} - K}$. Therefore, any such λ yields $\mathbb{E}_{s_0 \sim p_0}\left[V_{\pi^{-i}}^{\text{SP}}(s_0)\right] \geq K$, so $\pi^{-i} \in \Pi_K^{-i}(p_0)$. □

B.2.2 Competence-Bounded Per-State Regret

The ROTATE teammate generation objective (Sec. 4.2, Eq. 8) has two components, and is restated below for convenience. One component is the cooperative regret maximization over $U(\pi^{-i})$. The other component is the BR-teammate return maximization over $p_{\text{XP}}(\pi^{-i})$ and $p_{\text{SP}}(\pi^{-i})$.

$$\begin{aligned}
J(\pi^{-i}, \pi^{\text{ego}}) = &2\mathbb{E}_{s_0 \sim \mathcal{U}(p_{\text{XP}},\, p_{\text{SP}})}\left[\text{CR}\left(\pi^{\text{ego}}, \pi^{-i}, s_0\right)\right] \\
&+ \lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}}\left[V\left(s_0|\pi^{-i}, \text{BR}(\pi^{-i})\right)\right] \\
&+ \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}}\left[V\left(s_0|\pi^{-i}, \text{BR}(\pi^{-i})\right)\right]. \qquad (8)
\end{aligned}$$

Following the same strategy as for the per-trajectory regret results, we first give a proposition stating the relationship between these two components of the teammate generation objective to prove the theorem in Section 4.2.

For any teammate π', the XP and SP state visitation distributions are $p_{\text{XP}}(\pi') := d(\pi', \pi^{\text{ego}}; p_0)$ and $p_{\text{SP}}(\pi') := d(\pi', \text{BR}(\pi'); p_0)$. Let $\Lambda := \lambda_1 + \lambda_2 > 0$, and define the three mixtures

$$\begin{aligned}
U(\pi') &:= \tfrac{1}{2} p_{\text{XP}}(\pi') + \tfrac{1}{2} p_{\text{SP}}(\pi'), \\
\mu(\pi') &:= \tfrac{\lambda_1}{\Lambda} p_{\text{XP}}(\pi') + \tfrac{\lambda_2}{\Lambda} p_{\text{SP}}(\pi'), \\
\nu(\pi') &:= \tfrac{\lambda_1+1}{\Lambda+2} p_{\text{XP}}(\pi') + \tfrac{\lambda_2+1}{\Lambda+2} p_{\text{SP}}(\pi'),
\end{aligned}$$

so that $U(\pi^{-i})$ is the distribution over which regret is maximized in Eq. 8, $\mu(\pi')$ is the λ-weighted mixture obtained by combining the two return terms of Eq. 8 into a single expectation, and $\nu(\pi')$ is the mixture appearing in Theorem 2.

Proposition 2. *Let $\lambda_1, \lambda_2 \in \mathbb{R}^{\geq 0}$ with $\Lambda > 0$. If the teammate π^{-i} maximizes Eq. 8, then*

$$\mathbb{E}_{s_0 \sim U(\pi^{-i})}\left[CR\left(\pi^{ego}, \pi^{-i}, s_0\right)\right] \geq \frac{\Lambda}{2}\left(R_\mu^{\max} - \mathbb{E}_{s_0 \sim \mu(\pi^{-i})}\left[V_{\pi^{-i}}^{SP}(s_0)\right]\right),$$

where

$$R_\mu^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim \mu(\pi')}\left[V_{\pi'}^{SP}(s_0)\right]$$

denotes the maximum μ-weighted return that any teammate achieves with its best response on the states that teammate induces.

Proof. First, for any teammate π' and function $F(s_0)$,

$$\begin{aligned}
&\lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi')}\left[F(s_0)\right] + \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi')}\left[F(s_0)\right] \\
&\quad = \Lambda\left(\frac{\lambda_1}{\Lambda}\mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi')}\left[F(s_0)\right] + \frac{\lambda_2}{\Lambda}\mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi')}\left[F(s_0)\right]\right) \\
&\quad = \Lambda\, \mathbb{E}_{s_0 \sim \mu(\pi')}\left[F(s_0)\right].
\end{aligned}$$

Because cooperative regret is always non-negative, we may obtain

$$\begin{aligned}
&\max_{\pi'}\left[2\mathbb{E}_{s_0 \sim U(\pi')}\left[CR\left(\pi^{\text{ego}}, \pi', s_0\right)\right] + \lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi')}\left[V_{\pi'}^{\text{SP}}(s_0)\right] + \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi')}\left[V_{\pi'}^{\text{SP}}(s_0)\right]\right] \\
&\quad \geq \max_{\pi'}\left[\lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi')}\left[V_{\pi'}^{\text{SP}}(s_0)\right] + \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi')}\left[V_{\pi'}^{\text{SP}}(s_0)\right]\right] \\
&\quad = \max_{\pi'}\left[\Lambda\, \mathbb{E}_{s_0 \sim \mu(\pi')}\left[V_{\pi'}^{\text{SP}}(s_0)\right]\right] \\
&\quad = \Lambda R_\mu^{\max},
\end{aligned}$$

where the last line is the definition of $R_\mu^{\max}$.

Because π^{-i} maximizes this equation, we get

$$\begin{aligned}
&2\mathbb{E}_{s_0 \sim U(\pi^{-i})}\left[CR\left(\pi^{\text{ego}}, \pi^{-i}, s_0\right)\right] \\
&\quad \geq \Lambda R_\mu^{\max} - \lambda_1 \mathbb{E}_{s_0 \sim p_{\text{XP}}(\pi^{-i})}\left[V_{\pi^{-i}}^{\text{SP}}(s_0)\right] - \lambda_2 \mathbb{E}_{s_0 \sim p_{\text{SP}}(\pi^{-i})}\left[V_{\pi^{-i}}^{\text{SP}}(s_0)\right] \\
&\quad = \Lambda\left(R_\mu^{\max} - \mathbb{E}_{s_0 \sim \mu(\pi^{-i})}\left[V_{\pi^{-i}}^{\text{SP}}(s_0)\right]\right).
\end{aligned}$$

Dividing by 2 shows that the proposition holds. □

This proposition states that, for any teammate that maximizes the ROTATE teammate generation objective, it must be true that the cooperative regret on $U(\pi^{-i})$ is at least as large as $\frac{\Lambda}{2}$ times the gap between the overall best possible μ-weighted return $R_\mu^{\max}$, and the best possible return with π^{-i} on the state distribution $\mu(\pi^{-i})$. The following theorem relies on the proposition to show that the guaranteed minimum competence of the teammate increases as λ_1 and λ_2 increase.

Theorem 2 (Per-State Competence Lower Bound)**.** *Let $\lambda_1, \lambda_2 \in \mathbb{R}^{\geq 0}$ with $\Lambda := \lambda_1 + \lambda_2 > 0$. For any teammate π', let $p_{XP}(\pi')$ and $p_{SP}(\pi')$ denote the XP and SP state visitation distributions induced by π', and define the mixtures $U(\pi') := \mathcal{U}(p_{XP}(\pi'), p_{SP}(\pi'))$, $\mu(\pi') := \frac{\lambda_1}{\Lambda} p_{XP}(\pi') + \frac{\lambda_2}{\Lambda} p_{SP}(\pi')$, and $\nu(\pi') := \frac{\lambda_1+1}{\Lambda+2} p_{XP}(\pi') + \frac{\lambda_2+1}{\Lambda+2} p_{SP}(\pi')$. Let $R_\mu^{\max} = \max_{\pi'} \mathbb{E}_{s_0 \sim \mu(\pi')}\left[V(s_0|\pi', BR(\pi'))\right]$ and $R_U^{\min} = \min_{\pi'} \mathbb{E}_{s_0 \sim U(\pi')}\left[V\left(s_0|\pi', \pi^{ego}\right)\right]$. If π^{-i} maximizes Eq. 8, then*

$$\mathbb{E}_{s_0 \sim \nu(\pi^{-i})}\left[V\left(s_0|\pi^{-i}, BR(\pi^{-i})\right)\right] \geq \frac{\Lambda}{\Lambda+2} R_\mu^{\max} + \frac{2}{\Lambda+2} R_U^{\min}.$$

Proof. By Proposition 2,

$$\begin{aligned}
&\mathbb{E}_{s_0\sim\mu(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0)\right] \\
&\quad\geq R^{\max}_{\mu} - \frac{2}{\Lambda}\mathbb{E}_{s_0\sim U(\pi^{-i})}\left[CR\left(\pi^{\mathrm{ego}},\pi^{-i},s_0\right)\right] \\
&\quad= R^{\max}_{\mu} - \frac{2}{\Lambda}\mathbb{E}_{s_0\sim U(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0) - V^{\mathrm{XP}}_{\pi^{-i}}(s_0)\right] \\
&\quad= R^{\max}_{\mu} - \frac{2}{\Lambda}\mathbb{E}_{s_0\sim U(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0)\right] + \frac{2}{\Lambda}\mathbb{E}_{s_0\sim U(\pi^{-i})}\left[V^{\mathrm{XP}}_{\pi^{-i}}(s_0)\right] \\
&\quad\geq R^{\max}_{\mu} - \frac{2}{\Lambda}\mathbb{E}_{s_0\sim U(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0)\right] + \frac{2}{\Lambda}R^{\min}_{U},
\end{aligned}$$

where the last inequality holds because π^{-i} is one of the teammates over which the minimum defining $R^{\min}_U$ is taken.

This implies that:

$$\mathbb{E}_{s_0\sim\mu(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0)\right] + \frac{2}{\Lambda}\mathbb{E}_{s_0\sim U(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0)\right] \geq R^{\max}_{\mu} + \frac{2}{\Lambda}R^{\min}_{U}.$$

Multiply both sides with $\dfrac{\Lambda}{\Lambda+2}$ to obtain:

$$\begin{aligned}
&\frac{\Lambda}{\Lambda+2}\mathbb{E}_{s_0\sim\mu(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0)\right] + \frac{2}{\Lambda+2}\mathbb{E}_{s_0\sim U(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0)\right] \\
&\qquad\geq \frac{\Lambda}{\Lambda+2}R^{\max}_{\mu} + \frac{2}{\Lambda+2}R^{\min}_{U}.
\end{aligned}$$

Then

$$\begin{aligned}
&\mathbb{E}_{s_0\sim\frac{\lambda_1}{\Lambda+2}p_{\mathrm{XP}}(\pi^{-i})+\frac{\lambda_2}{\Lambda+2}p_{\mathrm{SP}}(\pi^{-i})+\frac{2}{\Lambda+2}U(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0)\right] \\
&\qquad\geq \frac{\Lambda}{\Lambda+2}R^{\max}_{\mu} + \frac{2}{\Lambda+2}R^{\min}_{U}.
\end{aligned}$$

Combining the distributions, we can rewrite the above inequality as

$$\mathbb{E}_{s_0\sim\nu(\pi^{-i})}\left[V^{\mathrm{SP}}_{\pi^{-i}}(s_0)\right] \geq \frac{\Lambda}{\Lambda+2}R^{\max}_{\mu} + \frac{2}{\Lambda+2}R^{\min}_{U}.$$

□

Theorem 2 certifies competence with respect to $\nu(\pi^{-i})$, which places weight $\frac{\lambda_1+1}{\Lambda+2}$ on the XP state distribution $p_{\mathrm{XP}}(\pi^{-i})$. By contrast, Theorem 1 certifies competence only with respect to p_0, so a teammate maximizing Eq. 7 can become incompetent at states visited only with π^{ego} without weakening its guarantee, which is the incentive to sabotage described in Sec. 4.2. Under the ROTATE objective, becoming incompetent at such states lowers $V^{\mathrm{SP}}_{\pi^{-i}}$ on the XP state distribution, and is therefore penalized by the return terms of Eq. 8.

C Supplemental Results

This section provides additional experiments examining the ROTATE teammate generation objective and population buffer in further detail. It also provides supplemental material supporting the main results. Specifically:

- We present additional analyses of the main result: unnormalized returns, per-task interval estimates, probability of improvement, worst-case returns, and a breakdown by individual evaluation teammate (App. C.1).
- We provide a human proxy evaluation of ROTATE and all baselines on the Overcooked tasks (App. C.2).

- We provide evaluations of ROTATE when extended to a partially observable variant of LBF (App. C.3) and an N-agent variant of LBF (App. C.4).
- We analyze the impact of competence weights on sabotage rates for per-state and per-trajectory regret (App. C.5).
- We analyze the generalization performance of ROTATE as competence weights are varied (App. C.6).
- We compare ROTATE to variants that use trajectory regret modulated by return, and CoMeDi-style mixed-play return maximization (App. C.7).
- We ablate the population buffer and examine whether the population generated by ROTATE is useful for training an independent ego agent (App. C.8).
- We analyze the teammates generated by ROTATE for their conventions, presence of hand-shake conventions, competence, and the regret they induce on the ego agent (App. C.9). We also assess the quality of the co-trained best responses (App. C.10).
- We present the learning curves for all variants of ROTATE tested in this paper (App. C.11).

C.1 Additional Analyses of the Main Result

This subsection presents five additional views of Fig. 3 (left), which compares the mean normalized evaluation return of ROTATE to all baselines on all tasks. Here, the same comparison is performed with unnormalized return units, aggregated by various summary statistics (IQM, median, probability of improvement), by worst case returns, and broken down by individual evaluation teammate.

Unnormalized returns. Table 2 reports the unnormalized evaluation returns corresponding to Fig. 3 (left). Under unnormalized evaluation returns, ROTATE remains the best method on five of six tasks with the exception of AA, where Minimax Return attains a higher return. The per-teammate upper bounds used for normalization are reported in Table 21, and per-teammate performance is shown in Fig. 7.

Table 2: Unnormalized mean evaluation returns with 95% stratified bootstrapped CI's. For LBF, the percent eaten is shown, while for Overcooked, the base return is shown.

	LBF	CR	AA	CC	CoR	FC
ROTATE	**89.5** [88.9, 90.1]	**175.1** [173.6, 176.7]	251.7 [248.2, 255.1]	**83.2** [82.2, 84.1]	**115.5** [114.2, 116.8]	**107.0** [105.5, 108.5]
Minimax Return	19.0 [18.4, 19.7]	107.9 [106.7, 109.1]	**258.6** [256.6, 260.5]	64.6 [63.6, 65.6]	77.9 [76.6, 79.2]	36.3 [35.5, 37.0]
COLE	82.5 [81.7, 83.4]	127.1 [125.6, 128.6]	172.7 [169.4, 176.0]	50.0 [48.9, 51.1]	101.6 [100.5, 102.8]	77.7 [76.1, 79.3]
PAIRED	63.8 [62.8, 64.8]	85.8 [84.6, 87.1]	186.2 [183.1, 189.3]	75.3 [74.3, 76.2]	101.6 [100.4, 102.9]	50.5 [49.6, 51.4]
FCP	77.9 [77.0, 78.8]	129.0 [127.4, 130.5]	181.6 [178.9, 184.2]	72.1 [70.8, 73.4]	78.3 [77.0, 79.5]	83.1 [81.5, 84.8]
BRDiv	77.2 [76.2, 78.1]	100.3 [98.3, 102.5]	175.0 [172.5, 177.7]	44.8 [43.8, 45.7]	66.8 [65.8, 67.9]	87.2 [85.4, 89.0]
CoMeDi	85.8 [85.1, 86.6]	145.2 [143.1, 147.4]	168.3 [164.9, 171.7]	71.7 [70.5, 72.8]	111.6 [110.1, 113.1]	103.5 [101.8, 105.2]

Additional statistics. The main paper reports mean normalized returns with 95% stratified bootstrapped CI's. Following the recommendations of Agarwal et al. [1], Fig. 5 reports the median, interquartile mean (IQM), and mean normalized return per task with 95% stratified bootstrap CI's. ROTATE attains the highest value of all three statistics on five of the six tasks, with the sole exception of AA. The one near-tie is the CC median, on which ROTATE and PAIRED are indistinguishable (0.635 each).

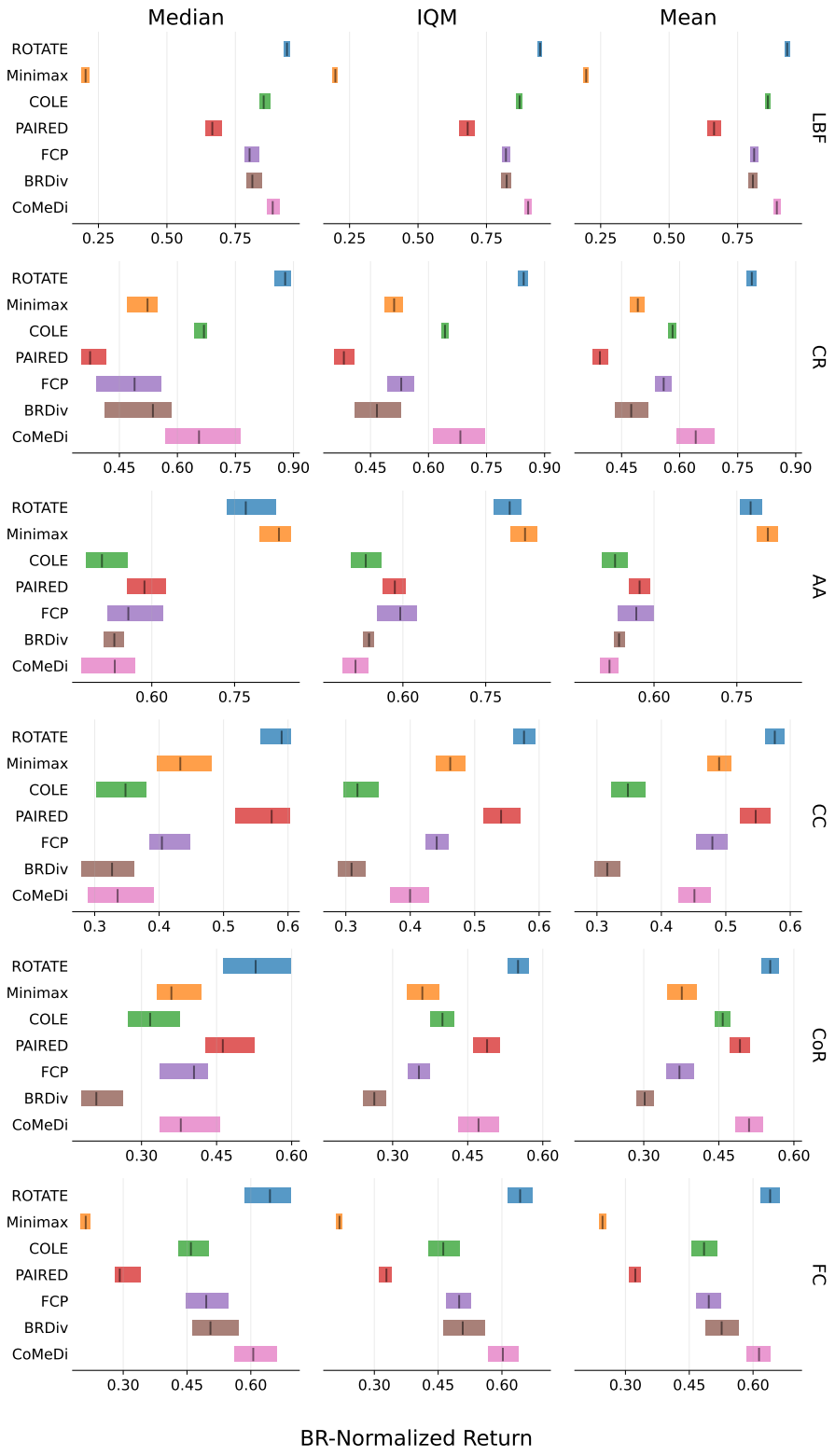


Figure 5: Median, IQM and mean normalized return per task, with 95% stratified bootstrap CI's over six trials.

Fig. 6 also reports the probability that, for each task and baseline, a randomly selected ROTATE run outperforms a randomly selected baseline run on a randomly selected evaluation teammate. ROTATE is favored ($P > 0.5$) in 35 of the 36 (task, baseline) pairs; the exception is Minimax Return on AA (0.44 [0.36, 0.52]). Of the 35 favorable comparisons, all but three have confidence intervals that exclude 0.5: PAIRED on CC, CoMeDi on CoR, and CoMeDi on FC.

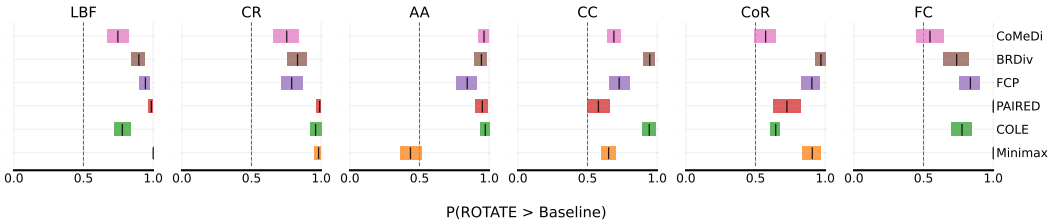


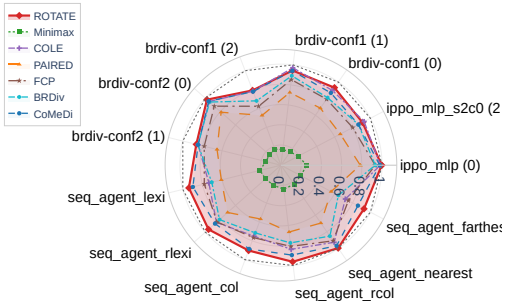
Figure 6: Probability that ROTATE improves over each baseline for each task, with 95% stratified bootstrap confidence intervals over six trials. The dashed line marks $P = 0.5$.

Worst-case returns. Because the goal of ROTATE is robustness to Π^{eval}, all methods are also compared on the *minimum* normalized return achieved against any member of Π^{eval}. Table 3 shows that ROTATE attains the highest worst-case return on five of six tasks with the sole exception of AA, where ROTATE is the second-best method. This result indicates that ROTATE demonstrates better-rounded cooperation with all evaluation teammates by avoiding performance collapse against any particular teammate.

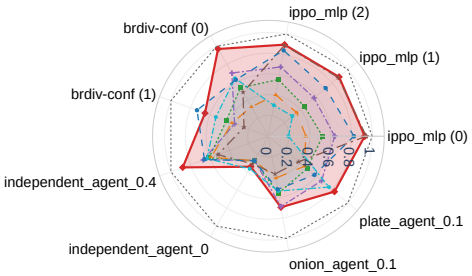
Table 3: Worst-case (minimum) normalized return achieved by a method with any evaluation teammate. 95% stratified bootstrap CI's over six trials displayed.

	LBF	CR	AA	CC	CoR	FC
ROTATE	**0.79** [0.71, 0.87]	**0.34** [0.32, 0.36]	0.56 [0.51, 0.63]	**0.39** [0.36, 0.43]	**0.42** [0.39, 0.45]	**0.47** [0.44, 0.51]
Minimax Return	0.15 [0.11, 0.20]	0.28 [0.25, 0.30]	**0.64** [0.57, 0.70]	0.31 [0.27, 0.35]	0.26 [0.22, 0.30]	0.16 [0.14, 0.19]
COLE	0.72 [0.66, 0.77]	0.27 [0.26, 0.28]	0.34 [0.28, 0.40]	0.28 [0.25, 0.32]	0.22 [0.19, 0.25]	0.32 [0.23, 0.44]
PAIRED	0.53 [0.45, 0.64]	0.27 [0.24, 0.30]	0.40 [0.34, 0.46]	0.38 [0.33, 0.43]	0.33 [0.26, 0.39]	0.26 [0.21, 0.31]
FCP	0.62 [0.55, 0.68]	0.25 [0.19, 0.31]	0.40 [0.30, 0.51]	0.30 [0.28, 0.32]	0.22 [0.19, 0.24]	0.37 [0.34, 0.39]
BRDiv	0.64 [0.59, 0.70]	0.19 [0.13, 0.27]	0.44 [0.41, 0.47]	0.16 [0.14, 0.18]	0.13 [0.10, 0.16]	0.43 [0.36, 0.51]
CoMeDi	0.78 [0.73, 0.83]	0.27 [0.22, 0.35]	0.37 [0.35, 0.39]	0.24 [0.18, 0.32]	0.28 [0.25, 0.33]	0.42 [0.40, 0.44]

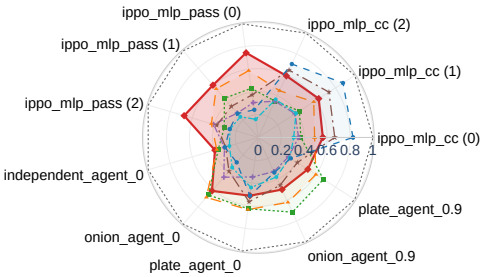
Per evaluation teammate performance breakdown. We break down the performance of ROTATE and all baseline methods by individual evaluation teammate policies as radar charts in Fig. 7. The radar charts show that ROTATE achieves higher performance across a larger number and variety of evaluation teammates than baselines. The best baseline, CoMeDi, achieves unusually high returns with the IPPO teammates on CC and CoR, and with BRDiv teammates on CR and FC, which explains CoMeDi's high mean return on these tasks. The radar charts also show that the second-best baseline, FCP, is strong specifically against IPPO teammates and relatively weaker on heuristics and BRDiv teammates, especially in CR and CC. As mentioned in the main paper, we attribute FCP's relative strength on IPPO evaluation teammates to the fact that the IPPO evaluation teammates are closer to the training teammate distribution constructed by FCP. While FCP is not especially strong against the "IPPO pass" agents in CC, these agents were trained via reward shaping to solve the task by passing onions across the counter rather than navigating around the counter, which is the policy found by IPPO without reward shaping (denoted as "IPPO CC" in the figures). COLE is also strong against IPPO teammates on CR, CC, CoR, perhaps because it relies on a pure return-maximization objective to generate novel teammates. Finally, on AA, Minimax Return is strong against the role-based heuristic teammates and somewhat adversarial BRDiv teammates, while being weaker against the cooperative IPPO teammates.



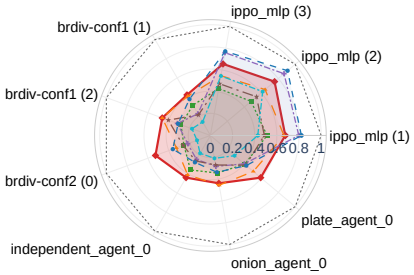
(a) LBF.



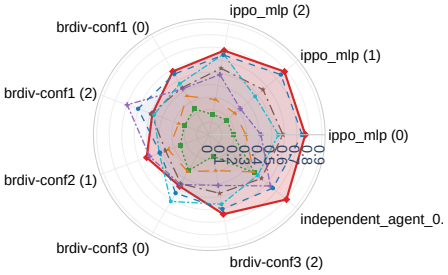
(b) Cramped Room.



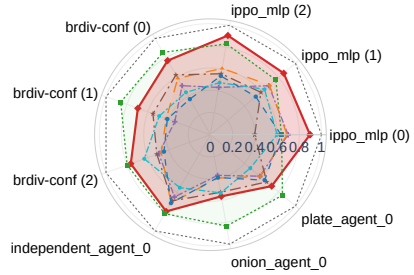
(c) Counter Circuit.



(d) Coordination Ring.



(e) Forced Coordination.



(f) Asymmetric Advantages.

Figure 7: Normalized mean returns of ROTATE and all baselines across all tasks, broken down by evaluation teammate in Π^{eval}. Legend shown for LBF applies for all plots.

C.2 Human Proxy Evaluations

The results in the main paper show that ROTATE is better able to generalize to unseen partners compared to baseline methods. Here, we evaluate the ability of ROTATE to generalize to *human* partners, compared to baselines. The evaluation is conducted with human proxy teammates generated by behavior cloning on the human gameplay dataset published by Carroll et al. [10]. Returns are normalized by the self-play performance of the human proxy teammates. Table 4 shows that ROTATE matches or exceeds the strongest baseline on all five Overcooked layouts. No baseline attains a significantly higher return than ROTATE on any layout, and ROTATE holds the highest mean return on CoR and FC. ROTATE is also the most consistent across layouts, with the other strong baselines collapsing in performance on at least one layout. CoMeDi collapses to 0.43 on AA, Minimax Return to 0.25 on FC, and PAIRED to 0.32 on FC, while ROTATE does not fall below 0.61. This

pattern mirrors the main result with the default evaluation teammates, in which ROTATE's advantage similarly comes from consistently avoiding cooperative failure with particular teammates.

Table 4: Normalized evaluation return with the human proxy teammate (95% stratified bootstrap CI's over six trials shown).

	CR	AA	CC	CoR	FC
ROTATE	0.84 [0.82, 0.87]	0.70 [0.68, 0.71]	0.87 [0.83, 0.92]	**0.68** [0.66, 0.73]	**0.61** [0.58, 0.65]
CoMeDi	**0.86** [0.86, 0.86]	0.43 [0.42, 0.44]	**0.89** [0.82, 0.96]	0.65 [0.63, 0.68]	0.57 [0.45, 0.68]
Minimax Return	0.53 [0.39, 0.64]	**0.73** [0.67, 0.76]	0.80 [0.75, 0.84]	0.40 [0.32, 0.53]	0.25 [0.22, 0.30]
PAIRED	0.49 [0.43, 0.58]	0.61 [0.59, 0.62]	**0.89** [0.78, 0.98]	0.50 [0.45, 0.56]	0.32 [0.30, 0.33]
FCP	0.24 [0.16, 0.32]	0.62 [0.55, 0.66]	0.66 [0.62, 0.72]	0.43 [0.37, 0.52]	0.55 [0.51, 0.57]
BRDiv	0.63 [0.62, 0.64]	0.57 [0.52, 0.61]	0.62 [0.47, 0.70]	0.59 [0.55, 0.61]	0.50 [0.42, 0.58]
COLE	0.69 [0.66, 0.71]	0.45 [0.42, 0.46]	0.57 [0.55, 0.61]	0.67 [0.66, 0.67]	0.55 [0.53, 0.57]

C.3 Extension to a Partially Observable Scenario

We consider a partially observable LBF variant where agents have a field of view limited to radius 2 in the 7×7 grid. A naive extension of ROTATE that simply uses recurrent networks for teammates and BR policies leads to a problem in SXP and XSP trajectory collection. Specifically, under recurrent policies, the BR's hidden state at the switching point is not meaningful to the ego agent.

Instead, we train a recurrent belief model inspired by Off-Belief Learning [24]. The belief model is an LSTM encoder-decoder that takes partial observations as input and predicts the full environment state. The use of a belief model allows all agents to condition on a shared state estimate, sidestepping the hidden-state incompatibility. The belief model is trained via supervised learning on rollouts collected from a population of self-play IPPO teams, achieving prediction accuracies of $\sim 42\%$ and $\sim 36\%$ for hidden foods and teammate position, respectively (in contrast to $\sim 9\%$ accuracy for a random model).

Table 5: ROTATE outperforms CoMeDi on partially observable LBF. Mean returns and bootstrapped 95% CI's are shown.

Method	PO LBF
ROTATE + belief model	**0.89** [0.87, 0.90]
ROTATE (recurrent)	0.75 [0.72, 0.77]
CoMeDi (recurrent)	0.70 [0.68, 0.72]

Table 5 shows that ROTATE achieves higher returns on partially observable LBF compared to CoMeDi. Even the naive recurrent extension of ROTATE already outperforms CoMeDi under partial observability. The belief-model variant improves further.

C.4 Extension to N-Agent Scenario

We consider an N-agent variant of LBF with 2 ego agents and 2 teammates. The extension of ROTATE and CoMeDi is performed using the most straightforward strategy of parameter sharing [15], in which the agent ID is appended to the observation to permit behavior differentiation between agents. The ego agents are considered one subteam, and the teammates another. Table 6 shows that ROTATE achieves higher returns on N-agent LBF compared to CoMeDi.

C.5 Impact of Competence Weights on Sabotage Rates for Per-State and Per-Trajectory Regret

This section analyzes the influence of competence weights on the rate of sabotage in the sabotage game specified in Fig. 4 (top) for ROTATE with both per-state regret and per-trajectory regret. We show that as the competence weight increases, the probability of the sabotage action decreases for

Table 6: ROTATE outperforms CoMeDi on N-agent LBF. Mean returns and bootstrapped 95% CI's are shown.

Method	N-agent LBF
ROTATE	**0.887** [0.871, 0.902]
CoMeDi	0.672 [0.651, 0.692]

both per-state and per-trajectory regret, with a much stronger effect observed for per-state regret. Further, the competence-bounded per-trajectory regret objective of Sec. 4.1 (Eq. 7) does not fully reduce the probability of the sabotage action.

Recall that Eq. 7 (competence-bounded per-trajectory regret) differs from the full ROTATE teammate generation objective only in the distributions over which the regret and teammate-BR return are maximized, and that its competence guarantee (Theorem 1) is with respect to the initial state distribution p_0.

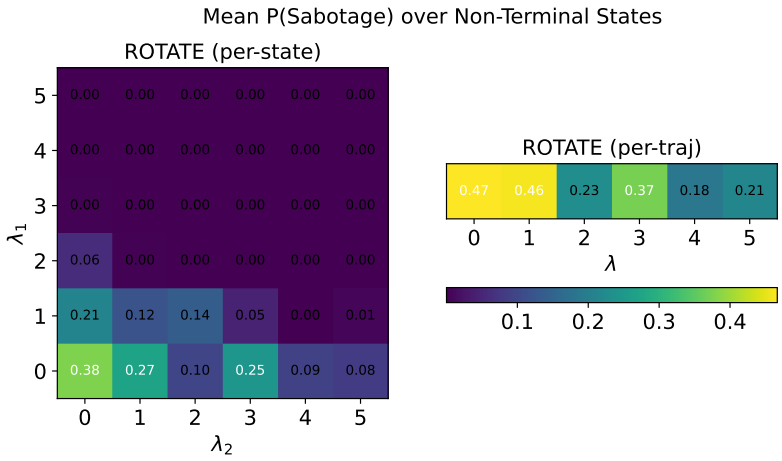


Figure 8: Mean probability of the sabotage action, aggregated across non-terminal states in the sabotage game, as the competence weights are varied. **(Left)** ROTATE with competence-bounded per-state regret. **(Right)** ROTATE with competence-bounded per-trajectory regret.

Fig. 8 displays the mean probability of sabotage, aggregated across non-terminal states for per-trajectory regret (right) and per-state regret (left), as the competence weights vary from 0 to 5. For *per-state regret*, the mean probability of sabotage rapidly decreases to 0 as the competence weights increase. The effect of λ_1 is somewhat stronger than that of λ_2 at combating sabotage, with $\lambda_1 = 3, \lambda_2 = 0$ leading to a zero probability of sabotage, whereas $\lambda_1 = 0, \lambda_2 = 5$ still corresponds to a sabotage probability of 0.08. However, high enough values of either weight can lead to a zero probability of sabotage. For *per-trajectory regret*, increasing the competence weight similarly decreases the probability of sabotage, but the effect is much weaker than for per-state regret. In fact, even with $\lambda = 5$, the probability of sabotage is still 0.21.

C.6 Generalization Performance as Competence Weights are Varied

Figures 9 and 10 show the training and evaluation returns of ROTATE as the competence weights λ_1, λ_2 are varied, on all tasks. As the competence weights increase, the training returns tend to increase. For evaluation returns, intermediate competence weights perform best on CC, AA, and FC, while higher competence weights perform best on LBF, CR, and CoR. Therefore, the optimal competence weights for improving cooperative generalization are task-dependent.

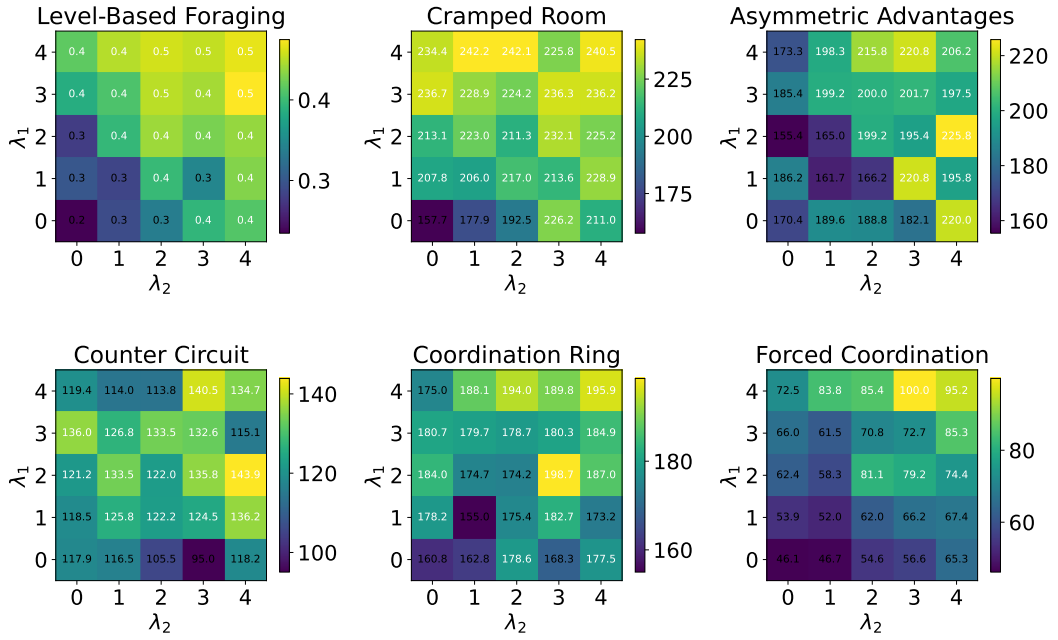


Figure 9: Train returns of ROTATE as the competence weights are varied.

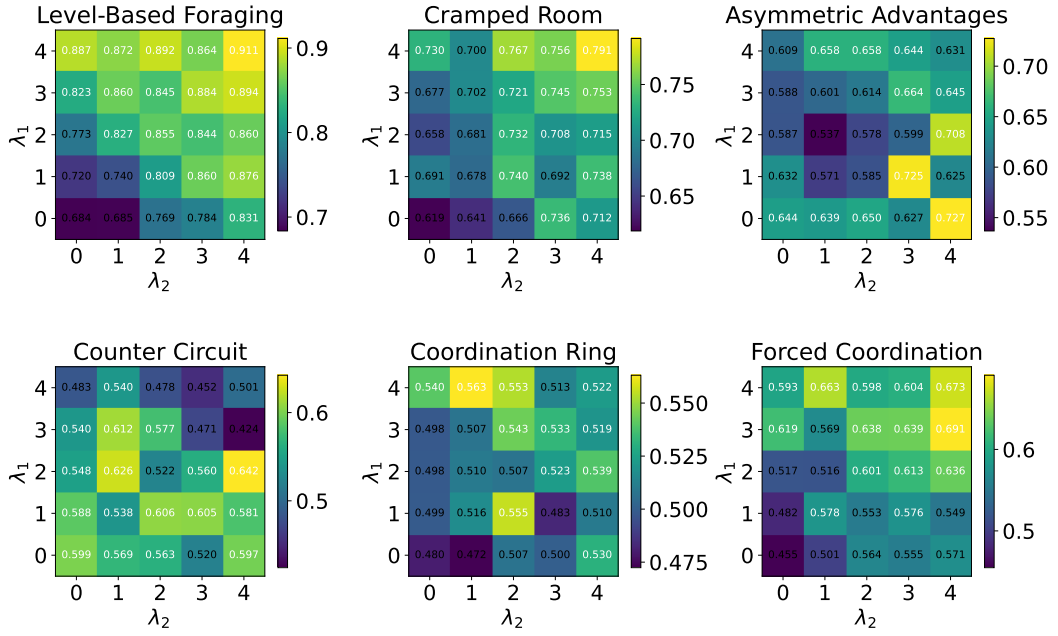
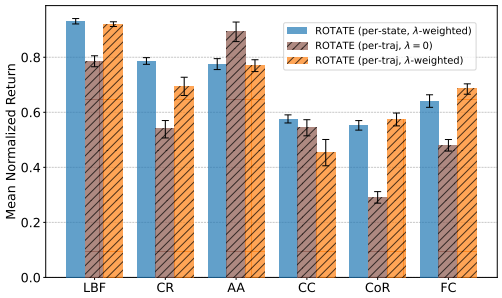


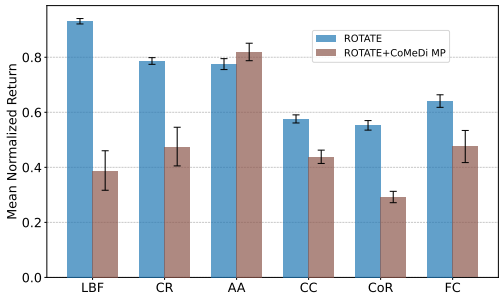
Figure 10: Normalized evaluation returns of ROTATE as the competence weights are varied.

C.7 Ablations of Teammate Generation Objective

We consider two alternatives to the teammate generation objective of ROTATE within the ROTATE framework. First, we consider competence-bounded per-trajectory regret. Second, we consider a version of ROTATE in which expected returns are maximized in states sampled from CoMeDi's mixed-play state distribution, p_{MP} (defined in App. D.2), rather than from SXP states, which resembles the mixed-play objective of CoMeDi [54] (as motivated in App. D.2).



(a) ROTATE with per-state vs per-trajectory regret, with and without λ-enforced competence bounds.



(b) ROTATE vs ROTATE with CoMeDi's mixed-play (MP) objective.

Figure 11: The mean normalized returns of ROTATE and ablations designed to evaluate the effectiveness of ROTATE's per-state regret-based teammate generation objective. 95% stratified bootstrapped CI's over three trials are shown.

Competence Bounded Trajectory Regret The ROTATE teammate generation objective consists of competence-bounded per-state regret. A natural question is how competence-bounded per-trajectory regret (Eq. 7) might compare in terms of training strong ego agents. As shown in Fig. 11a, ROTATE outperforms per-trajectory regret ($\lambda = 0$) on every task except AA. Because AA is also the task where adversarially robust behavior is optimal, this provides indirect evidence that per-trajectory regret induces adversarial behavior. Bounding competence by adding the same per-task λ values used by ROTATE to the per-trajectory objective raises its return on most tasks relative to $\lambda = 0$, and reduces the gap to ROTATE on 4/6 tasks. Therefore, for these benchmark tasks, placing a p_0 competence bound on per-trajectory regret is sufficient to reduce sabotaging behavior of plain per-trajectory regret. We still expect competence-bounded per-state regret (the ROTATE objective) to produce stronger cooperative generalization in broader tasks. As discussed in Sec. 4.2, enforcing competence on a per-state rather than per-trajectory basis is important for reducing the teammate's incentive to sabotage, which is demonstrated on the sabotage matrix game in App. C.5.

Mixed Play Ablation For fair comparison, the mixed-play variant uses the same hyperparameters as ROTATE and CoMeDi's tuned mixed-play weights. ROTATE's teammate generation objective leads to higher normalized returns than maximizing CoMeDi's mixed-play objective in all environments except for Overcooked's Asymmetric Advantages (AA) setting (Fig. 11b). Following the difference in starting states of trajectories for which these two maximize self-play returns, we conjecture that this is because ROTATE generates competent teammate policies in states from XP while the CoMeDi-like approach imposes the same thing in states from p_{MSTART}. Imposing good faith within policies in p_{XP} is likely more important for training an ego agent that initially interacts with π^{-i} during training by visiting states from p_{XP}.

C.8 Ablations of the ROTATE Population

This subsection presents two ablations of the ROTATE population, both reported in Fig. 12. The first removes the population buffer described in Sec. 5, and the second trains an independent ego agent on the final ROTATE population.

Is the population buffer necessary for ROTATE to learn well? (Yes.) We hypothesize that collecting all previously generated teammates in a population buffer helps the ROTATE agent improve in robustness against all previously discovered conventions. On the other hand, if there is no population buffer, then it becomes possible for the ROTATE ego agent to forget how to collaborate with teammates seen at earlier iterations of open-ended learning [29], which creates the possibility that the ego agent and teammate generator oscillate between conventions. As shown in Fig. 12, ROTATE without the population buffer attains lower evaluation returns than the full ROTATE method on all tasks except AA, thus supporting the hypothesis that the population buffer improves ego agent learning. As discussed in the RQ1 results (Sec. 6.2), AA is a unique layout where agents can complete the task independently, even in the presence of an adversarial partner. As a corollary,

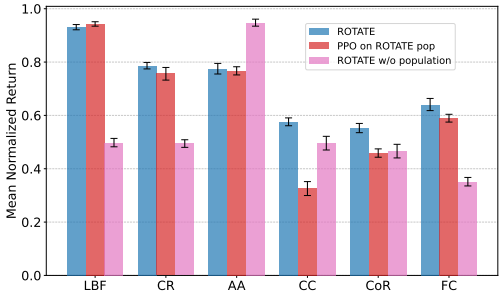


Figure 12: The mean normalized returns of ROTATE compared to an independently trained ego agent on ROTATE's population and an ablation of ROTATE without the population. 95% stratified bootstrapped CI's over three trials are shown.

there are few meaningful cooperative conventions that can be discovered, and no scenarios where convention mismatch leads to zero return (unlike LBF and FC).

Is the ROTATE population useful for training an independent ego agent? (Yes.) Two-stage AHT algorithms first generate a population of teammates, and next train an ego agent against the population. Although ROTATE's teammate generation mechanism relies on the learning process of a particular ego agent, here, we investigate whether the population generated by ROTATE is useful for training independently generated ego agents. Fig. 12 compares the mean evaluation returns of the ROTATE ego agent against the mean evaluation returns of an independently trained PPO ego agent that was trained on the full, final ROTATE population, using the same configuration as ROTATE. In 3/6 tasks (CC, CoR, and FC) the ROTATE ego agent outperforms the independently trained ego agent with non-overlapping confidence intervals, and on the remaining three tasks (LBF, CR, and AA) the two perform similarly. This result suggests that the ROTATE population is a useful population of teammates even independent of the particular ego agent generated.

Note that because the population used to train the independent ego agent was itself produced by ROTATE's coupled generation, this experiment isolates the ego agent's training regime—the curriculum induced by coupled teammate generation versus joint training on the full, final population—rather than the value of coupled generation itself. In continual learning, joint training on all tasks is often considered an oracle upper bound for sequential training [35]. Viewed this way, the independently trained ego agent's strong performance is unsurprising: it faces a stationary training distribution and interacts with all teammates uniformly throughout training, whereas the ROTATE ego agent faces a nonstationary, growing population and trains against later teammates for fewer iterations. Closing the remaining gap to joint training (e.g., via curated buffer sampling) is left to future work.

C.9 Inspection of ROTATE Generated Teammates

This section analyzes the ROTATE teammate populations on LBF and CC for the learned conventions, presence of handshake conventions, competence and diversity (as measured by XP matrices), and the regret induced on the ego agent during training. Results are in *unnormalized* return units. For reference, the maximum return on LBF is 0.5 (corresponding to 100% of food eaten in the percent-eaten results; see App. H), while CC has no well-defined maximum but awards 20 points per delivered dish.

Conventions. On CC, the behavior of ROTATE teammates is characterized by using 63 features to create embedding vectors that characterize interactions between each teammate and its co-trained BR. The features track handoffs, region occupancy, resource usage, role specialization, and routing. The resultant embedding vectors, z-scored and weighted by feature group and test-retest reliability, are clustered by cosine distance in feature space using agglomerative clustering [43] with average linkage, where the number of clusters is chosen by silhouette score together with a bootstrap stability check over episodes. Because competence and conventions could jointly influence cluster structure, we verified that similar-return teammates do not always share clusters.

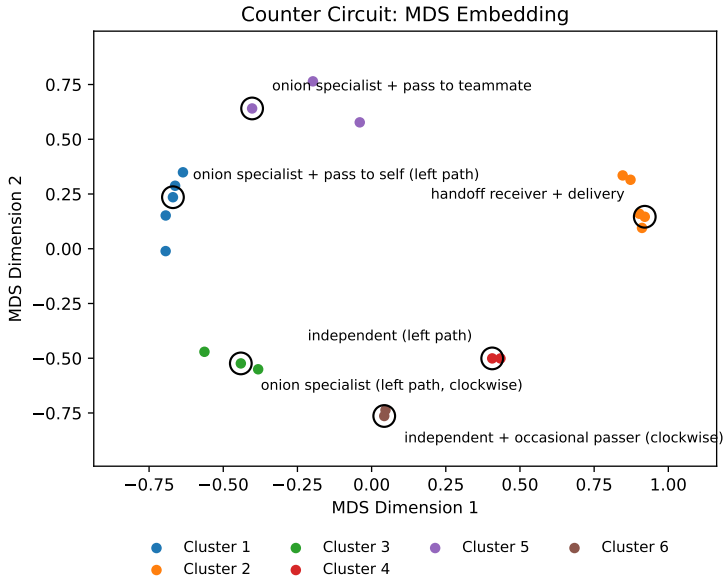


Figure 13: MDS embedding of the pairwise cosine distances between the behavioral feature vectors of Counter Circuit teammates, colored by agglomerative clustering. Cluster medoids are labeled with the convention identified by visually inspecting trajectory replays.

To analyze the conventions contained in the ROTATE teammate population, we visualize the teammate-BR corresponding to each cluster medoid, to find that the population includes the following 6 conventions:

- *3 onion specialists (pass to self, left path / pass to teammate / left path, clockwise)*: pass onions to themselves across the counter routing along the left side, pass onions across the counter to the partner, or pass onions routing along the left side and traveling clockwise, respectively.
- *Handoff receiver with delivery*: receives onions from a giver, pots them, then plates and delivers.
- *Independent, left path*: completes the full task alone via the left path.
- *Independent with occasional passing, clockwise*: mostly solo, but coordinates via occasional onion passes, moving clockwise with the partner.

We observe no sabotage-like behaviors such as blocking, resource hoarding, or delivery prevention. Fig. 13 visualizes the cluster structure by applying multidimensional scaling (MDS) [64] to the pairwise cosine distance matrix. Each cluster medoid is labeled by the corresponding convention.

Perturbation analysis of handshake-style sabotage. ROTATE's teammate generation objective is designed to mitigate adversarial behaviors such as *handshake conventions* [54] (policies that coordinate only in SP states and otherwise act in ways that harm coordination). ROTATE gathers XP interactions (teammate-ego) starting from states visited under SP (teammate-BR), and gathers SP interactions starting from states visited under XP, so no state is exclusive to SP or XP interaction, thereby discouraging handshakes.

To empirically examine whether ROTATE teammates-BRs have learned handshake conventions, we measure each teammate π's return when paired with its noise-perturbed BR, $(1-\epsilon)\cdot BR(\pi)+\epsilon\cdot\pi_{\text{rand}}$, sweeping ϵ from 0 to 1. The returns are averaged over all teammates in the population (dropping two teammates that collapsed to zero return on LBF). A teammate playing a handshake policy would show a sharp decline in team return at some level of ϵ. Instead, Table 7 shows that as ϵ increases, the returns decline smoothly starting from $\epsilon = 0.8$ on LBF and from $\epsilon = 0.2$ on CC.

Table 7: Mean return of each ROTATE teammate paired with its noise-perturbed best response, $(1-\epsilon)\cdot BR(\pi)+\epsilon\cdot\pi_{\text{rand}}$, as the perturbation level ϵ is swept from 0 to 1 on LBF and Counter Circuit (CC).

Task	ϵ=0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
LBF	0.498	0.499	0.499	0.499	0.498	0.497	0.488	0.455	0.368	0.252	0.131
CC	140.6	140.4	136.0	128.4	116.8	102.3	83.5	63.0	43.2	27.0	16.3

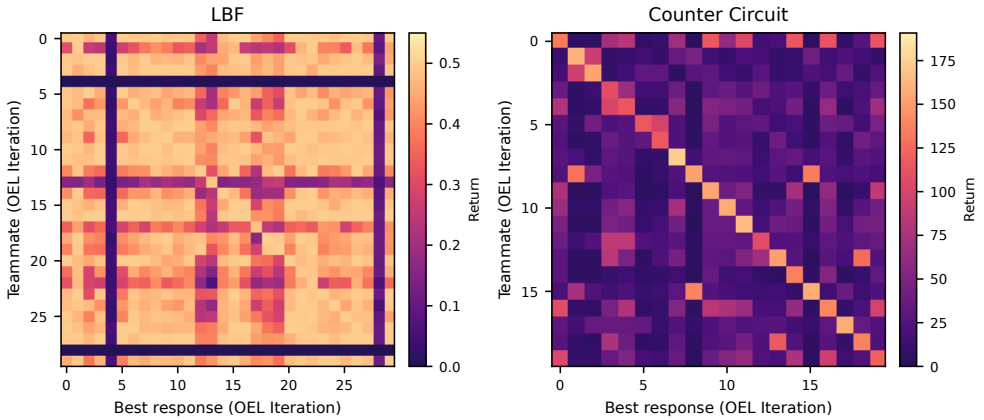


Figure 14: Cross-play matrices of ROTATE teammates (rows) evaluated with each teammate's co-trained best response (columns) on LBF and Counter Circuit, in raw return units.

Teammate competence and diversity (XP matrix). To analyze the competence and diversity of ROTATE's teammate population, we compute the cross-play matrix of ROTATE teammates and co-trained BRs (Fig. 14). The diagonal displays the estimated SP returns, and therefore estimates the teammates' competence. The ratio between diagonal and off-diagonal returns provides evidence of convention diversity, since prior sections verified the absence of adversarial conventions.

On LBF, the XP matrix shows generally high values, but still shows zones of lower returns. It also reveals that 2/30 teammates achieve uniformly zero returns; visual inspection confirmed that these teammates collapsed to idle behavior. The diagonal returns are near-optimal, and remain somewhat higher than the off-diagonals (mean of 0.499 vs 0.427, with the two idle teammates removed). Taken together with the fact that ROTATE achieves high cooperative generalization with LBF evaluation teammates (Fig. 3 (left)), the XP matrix shows that there are few truly incompatible conventions on LBF, which is consistent with the XP matrix of the evaluation teammates (Fig. 20). On CC, the XP matrix shows a much higher gap between the diagonal and off-diagonal returns, consistent with our conventions analysis which showed that ROTATE teammates learned distinct, interpretable conventions on CC.

Regret induced by teammates on the ego agent. Fig. 15 visualizes the (per-trajectory) regret induced by the teammate on the ego agent during each *teammate generation phase* of open-ended learning (i.e., $\mathbb{E}_{s_0\sim p_0}[V(s_0 \mid \pi^{-i}, \text{BR}(\pi^{-i})) - V(s_0 \mid \pi^{-i}, \pi^{\text{ego}})]$, in unnormalized return units). Note that the ego policy is frozen during the teammate generation phase, while the teammate and BR are initialized from scratch, which leads to the spikes and dips observed in the regret curves. At the start of each OEL iteration, the teammate and BR are learning from scratch against the current ego agent, so regret is initially negative. Within each iteration, as the teammate/BR learn to cooperate, regret increases smoothly, rising above zero in all tasks except for AA, where the regret remains negative after the 8th OEL iteration. We partially attribute the negative regret in AA to the increasing robustness of the ego agent over OEL iterations, and attribute the remainder to lack of sufficient training timesteps per-iteration. Indeed, in most tasks, the maximum per-iteration regret declines over iterations, with the effect being especially strong in LBF, CC, and FC. In AA, the teammates have minimal coordination opportunities with the ego agent, which may explain why the regret there remains negative.

Due to the ROTATE competence-bounded per-state regret objective, the teammate must still act cooperatively with the ego, so the per-iteration maximum regret does not become too high. On LBF, the maximum return is 0.5 (in raw return units; see App. H), but the per-iteration regret peaks are typically less than 0.1. On CC, the regret peaks are between 25-50, which reflects a difference in SP and XP returns corresponding to delivering 2-4 dishes.

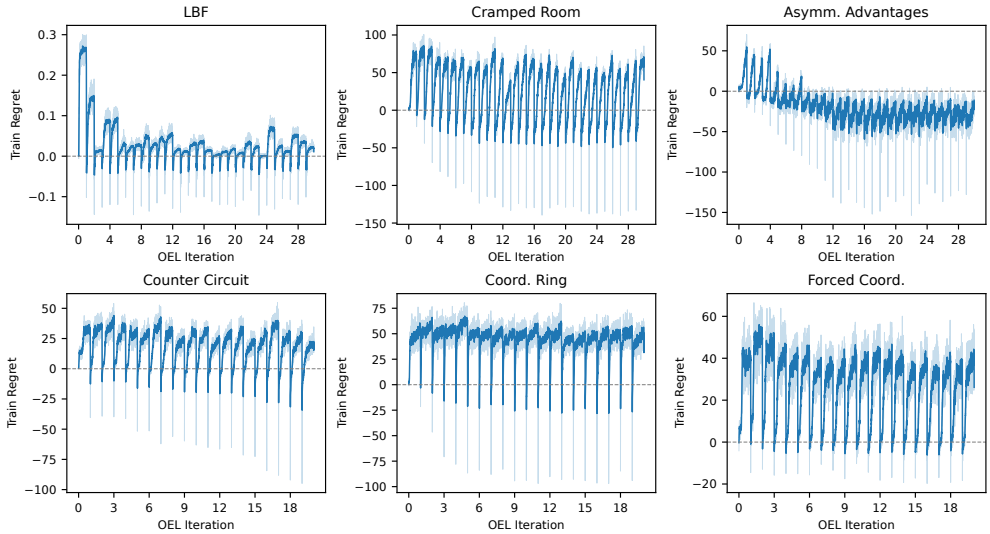


Figure 15: Regret induced on the ego agent during each teammate generation phase of ROTATE open-ended learning, in unnormalized return units.

C.10 Quality of ROTATE Co-Trained Best Responses

ROTATE's regret estimates rely on co-trained PPO approximations of the BR, which may fail to find the true BR. To assess the quality of these approximations, we independently train PPO BR policies (three seeds) for five ROTATE teammates on LBF and CC, with twice the training budget of the co-trained BR. Tables 8 and 9 report the per-teammate returns. The results show no systematic underestimation in either task. On LBF, the mean gap across 5 teammates (ind - co) is +0.0052, while on CC, the mean gap is +0.9. While it may be surprising that the co-trained BRs can sometimes outperform the independently trained BRs, we attribute it to the fact that the co-trained BRs have a different training regime from the independently trained BRs. Specifically, the co-trained BRs are trained to maximize SP returns from XP states (i.e. SXP states) in order to remain a BR to the teammate everywhere that the two interact.

Table 8: Mean return of each ROTATE teammate's co-trained best response on LBF vs. an independently trained PPO best response, with 95% CIs displayed.

Teammate (iter.)	Co-trained BR	Independent BR	Gap (Ind - Co)
0	0.493 [0.483, 0.500]	0.497 [0.493, 0.500]	+0.004
5	0.488 [0.473, 0.500]	0.499 [0.496, 0.500]	+0.010
10	0.492 [0.480, 0.500]	0.495 [0.490, 0.499]	+0.003
13	0.487 [0.474, 0.497]	0.496 [0.492, 0.499]	+0.009
25	0.499 [0.496, 0.500]	0.499 [0.496, 0.500]	-0.000

C.11 Learning Curves

Fig. 16 shows learning curves for ROTATE and all ROTATE variations tested in this paper, where the x-axis is the open-ended learning iteration, and the y-axis corresponds to the mean evaluation return. On 4/6 tasks (LBF, CR, CC, and FC), ROTATE has better sample efficiency than the variants. On 3/6 tasks (LBF, CR, and FC), ROTATE dominates the variants at almost all points in learning.

Table 9: Return of each ROTATE teammate's co-trained best response on Counter Circuit versus an independently trained PPO best response with 95% CIs displayed.

Teammate (iter.)	Co-trained BR	Independent BR	Gap (Ind - Co)
3	107.2 [103.6, 110.5]	109.0 [107.1, 110.8]	+1.8
6	113.4 [107.8, 118.6]	136.8 [132.6, 140.8]	+23.4
7	178.4 [174.7, 182.0]	149.3 [144.3, 154.2]	-29.1
12	116.1 [110.6, 121.1]	117.5 [114.7, 120.1]	+1.4
18	130.3 [123.0, 137.0]	137.3 [133.5, 140.9]	+7.0

D Supplemental Discussion

Here, we describe how ROTATE satisfies the definition of an open-ended system given by Hughes et al. [25]. Next, we compare and contrast CoMeDi's mixed-play mechanism to ROTATE's per-state regret. Finally, we discuss alternative estimators for ROTATE per-state regret.

D.1 ROTATE as an Open-Ended System

D.1.1 Definition of Open-ended Systems

To formalize open-endedness, we employ the definition of open-ended systems provided by Hughes et al. [25]. Hughes et al. [25] consider a system S that produces a sequence of artifacts X_t, indexed by time t. An observer O attempts to predict a new artifact X_T based on observations of past artifacts seen up until time t.

> From the perspective of an observer, a system is *open-ended* if and only if the sequence of artifacts it produces is both novel and learnable.

A sequence of artifacts is novel if it becomes increasingly unpredictable with respect to an observer's model at any fixed time t. Formally:

$$\forall t, \forall T > t, \exists T' > T : \mathbb{E}[\ell(t, T')] > \mathbb{E}[\ell(t, T)]. \tag{11}$$

This is quantified by a loss metric, $\ell(t, T)$, where the first argument refers to the timestep that the observer has observed up until (i.e., artifacts $X_1, \ldots, X_t$), while the second argument refers to the artifact that the observer must now predict (i.e., X_T).

Conversely, the system is learnable whenever conditioning on a longer history makes artifacts more predictable:

$$\forall T, \forall t < T, \forall T > t' > t : \mathbb{E}[\ell(t', T)] < \mathbb{E}[\ell(t, T)]. \tag{12}$$

D.1.2 Open-endedness in ROTATE

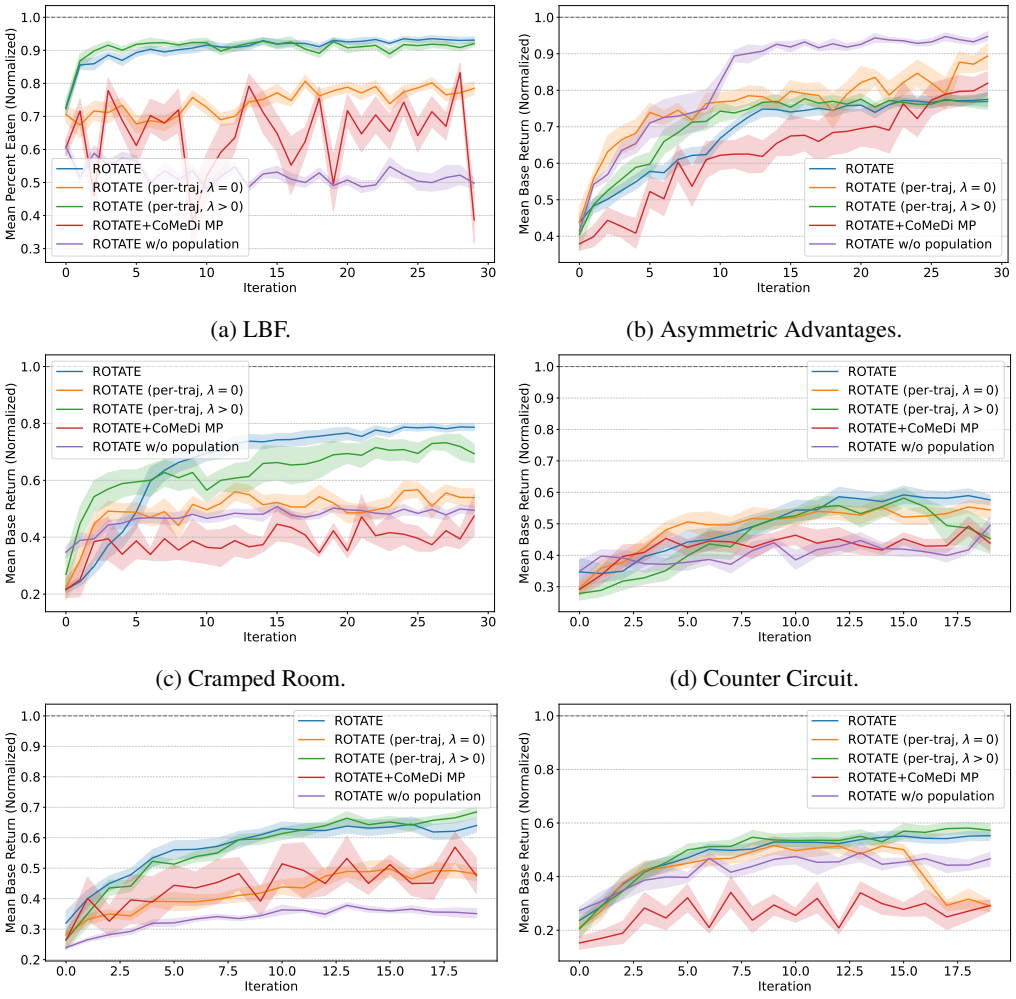
ROTATE displays both novelty and learnability according to the definition above, as we show below. Let the ego agent be the observer, the sequence of generated teammates be the artifacts, and $\ell(t, T)$ be the cooperative regret of the ego agent trained up to iteration t, evaluated on the teammate generated at iteration T.

Novelty. At open-ended iteration $t + 1$, ROTATE seeks to generate a teammate that *challenges* the ego agent from iteration t by directly maximizing its cooperative regret. By construction, for $T' = T + 1$:

$$\mathbb{E}[\ell(t, T')] > \mathbb{E}[\ell(t, T)]. \tag{13}$$

Learnability. In the context of ROTATE, this condition states that for some *future unseen teammate from time* T, the cooperative regret of ROTATE's ego agent from iteration t' should be lower than the cooperative regret of ROTATE's ego agent from an earlier iteration t.

Intuitively, since all generated teammates are added to a population buffer that is uniformly sampled, ROTATE's ego agent at future iterations should always be more capable than the ego agent at previous



(a) LBF.

(b) Asymmetric Advantages.

(c) Cramped Room.

(d) Counter Circuit.

(e) Forced Coordination.

(f) Coordination Ring.

Figure 16: Learning curves of ROTATE and all variations of ROTATE considered in this paper. Normalized mean returns and stratified bootstrapped 95% confidence intervals over 3 trials on Π^{eval} are shown.

iterations. Empirically, we observe this in the learning curves presented in Fig. 16. The normalized return against the test set of unseen teammates increases as the iterations increase. Importantly, the return is min-max normalized so that 1.0 represents the return achieved by a teammate with its best response, while 0.0 represents a known lower bound on returns. For a fixed teammate π^{-i}, the upper bound $V(\pi^{-i}, \text{BR}(\pi^{-i}))$ and the lower bound η are constant with respect to the ego agent, so the min-max normalized return is an affine, strictly decreasing function of the cooperative regret, $\text{NRet}(t) = 1 - \text{CR}(t)/c$ with $c = V(\pi^{-i}, \text{BR}(\pi^{-i})) - \eta > 0$. An increase in normalized return from iteration t to t' is therefore equivalent to a decrease in cooperative regret over the same interval.

D.2 Relationship To CoMeDi and Mixed Play

CoMeDi [54] is a two-stage teammate generation AHT algorithm, whose teammate generation process trains one teammate per iteration, with an objective that encourages the new teammates to be distinct from previously discovered teammates.

CoMeDi adds trained teammate policies to a teammate policy buffer, Π^{train}. Each iteration begins by identifying the teammate policy that is most *compatible* with the currently trained teammate π^{-i}, out

of all previously generated policies:

$$\pi^{\text{comp}} = \underset{\pi^{-j} \in \Pi^{\text{train}}}{\text{argmax}} \, \mathbb{E}_{s_0 \sim p_0}[V(s_0|\pi^{-i}, \pi^{-j})]. \tag{14}$$

The new teammate policy π^{-i} is trained with an objective that improves the per-trajectory regret objective (Eq. 6) by adding a term that maximizes the returns from states gathered in *mixed-play*, which we describe below.

Let mixed-play starting states be sampled from states visited when π^{-i} interacts with the *mixed* policy, which uniformly samples actions from π^{comp} and $\text{BR}(\pi^{-i})$ at each timestep:

$$p_{\text{MSTART}} := d\left(\pi^{-i}, \frac{1}{2}\pi^{\text{comp}} + \frac{1}{2}\text{BR}(\pi^{-i}); p_0\right). \tag{15}$$

From these starting states, CoMeDi then gathers mixed-play interaction data, where π^{-i} interacts with $\text{BR}(\pi^{-i})$. The resulting mixed-play state visitation is then expressed as:

$$p_{\text{MP}} := d\left(\pi^{-i}, \text{BR}(\pi^{-i}); p_{\text{MSTART}}\right). \tag{16}$$

The complete objective that Sarkar et al. [54] optimizes to train a collection of diverse teammates is then defined as:

$$\max_{\pi} \left(\mathbb{E}_{s_0 \sim p_0}\left[\text{CR}(\pi^{\text{comp}}, \pi^{-i}, s_0)\right] + \underbrace{\mathbb{E}_{s_0 \sim p_{\text{MP}}}[V(s_0|\pi, \text{BR}(\pi))]}_{\text{mixed-play return maximization}}\right). \tag{17}$$

CoMeDi [54] optimizes this objective to discourage π^{-i} from learning poor actions for collaborations outside of p_{SP}. This is because π^{-i} is now also trained to maximize returns in states visited during mixed-play, which resembles some states encountered while cooperating with π^{comp}. Discerning whether a state is likely encountered while interacting with π^{comp} and consequently choosing to diminish team returns will no longer be an optimal policy to maximize Eq. 17.

Although the importance of using p_{MSTART} as a starting state for data collection is questionable, we take inspiration from CoMeDi's maximization of $V(s_0|\pi, \text{BR}(\pi))$ outside of states from p_{SP}. We argue that maximizing $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ is a key component towards making π^{-i} act in good faith by always choosing actions yielding optimal collective returns assuming $\text{BR}(\pi^{-i})$ is substituted as the partner policy. Unlike CoMeDi, ROTATE maximizes $V(s_0|\pi^{-i}, \text{BR}(\pi^{-i}))$ on trajectories gathered from a starting state from p_{XP} (i.e., SXP states) instead of p_{MSTART}, which results in the λ-weighted return terms of Eq. 8. We formulate this objective to encourage π^{-i} to act in good faith in states sampled from p_{XP}, which is visited while π^{-i} interacts with π^{ego}. Since π^{-i} is not sabotaging π^{ego} by selecting actions that make collaboration impossible in p_{XP}, the ego policy learning process becomes less challenging. We conjecture that this leads to π^{ego} with better performance, as indicated in Fig. 3.

While Fig. 3 compares ROTATE with CoMeDi, Fig. 11b compares ROTATE with a modified CoMeDi approach that now follows the open-ended training framework described in Algorithm 1. In this modified version of CoMeDi, we train a newly generated teammate policy to maximize Eq. 17 while substituting π^{comp} with the trained π^{ego}. Rather than promoting meaningful differences with previously generated teammate policies, this creates a teammate policy that maximizes the ego agent policy's per-trajectory regret while discouraging teammates from diminishing team returns in certain states. This version of CoMeDi's teammate generation objective within the ROTATE open-ended framework is visualized in Fig. 17.

E The Policy Gradient of Per-State Regret

This section supports Section 4 of the main paper, showing how a teammate policy is trained to maximize the ROTATE per-state regret objective using the standard policy gradient theorem [62, 70]. First, the gradient of the cooperative regret objective is expressed as a difference of two standard policy gradients. Next, the section describes how optimizing this objective differs from standard return maximization. The section ends by describing gradient estimators that trade off bias and variance.

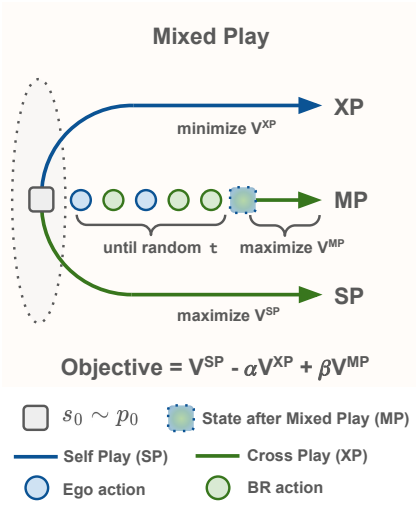


Figure 17: CoMeDi-style mixed-play objective for teammate generation, in the context of open-ended AHT.

The full ROTATE teammate generation objective (Eq. 8) augments the cooperative regret (CR) term with λ-weighted teammate return terms that enforce competence. Because those return terms are maximized by a standard policy gradient, the analysis below focuses on the policy gradient for the per-state regret term, and drops the λ-weighted competence term.

Reduction to standard policy gradients. Let p_{opt} denote the state distribution under which cooperative regret is maximized, which corresponds to the mixture $\mathcal{U}(p_{\text{XP}}, p_{\text{SP}})$ used in Eq. 8. The expected cooperative regret from starting states $s_0 \sim p_{\text{opt}}$ is defined as:

$$\mathbb{E}_{s_0 \sim p_{\text{opt}}}\left[\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0)\right] = \mathbb{E}_{s_0 \sim p_{\text{opt}}}\left[V(s_0|\pi^{-i}, BR(\pi^{-i})) - V(s_0|\pi^{-i}, \pi^{\text{ego}})\right], \tag{18}$$

where the value functions on the right-hand side are respectively defined as:

$$\begin{aligned} V(s_0|\pi^{-i}, BR(\pi^{-i})) &= \mathbb{E}_{\substack{a_t^{-i} \sim \pi^{-i}, \\ a_t^{\text{partner}} \sim \pi^{\text{BR}}, P}}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0\right], \\ V(s_0|\pi^{-i}, \pi^{\text{ego}}) &= \mathbb{E}_{\substack{a_t^{-i} \sim \pi^{-i}, \\ a_t^{\text{partner}} \sim \pi^{\text{ego}}, P}}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0\right], \end{aligned} \tag{19}$$

where a_t^{partner} denotes the action of the teammate's partner ($BR(\pi^{-i})$ in the first value function, and π^{ego} in the second), and $a_t = (a_t^{-i}, a_t^{\text{partner}})$ denotes the joint action.

The teammate policy π^{-i} is a neural network parameterized by θ^{-i}. Note that during teammate policy learning, the ego agent policy π^{ego} is fixed. We assume that the distribution p_{opt} and the best-response policy π^{BR} are fixed with respect to θ^{-i} (justified below). By linearity of the gradient, $\nabla_{\theta^{-i}}$ applied to Eq. 18 splits into a difference of the gradients of two expected returns. Each expected return is an instance of the standard policy gradient setting, in which $\pi^{-i}_{\theta^{-i}}$ is the only learnable policy. Concretely, because the partner policy in each term (π^{BR} in the first term, and π^{ego} in the second) is fixed, the partner can be absorbed into the environment dynamics from the perspective of π^{-i}, so each term can be treated as the expected return in a single-agent Markov decision process. Applying the policy gradient theorem [62, 70] to each term therefore yields:

$$\begin{aligned} \nabla_{\theta^{-i}} \mathbb{E}_{s_0 \sim p_{\text{opt}}}\left[\text{CR}(\pi^{\text{ego}}, \pi^{-i}, s_0)\right] = \mathbb{E}_{s_0 \sim p_{\text{opt}}}\Bigg[&\mathbb{E}_{\tau \sim \mathbb{P}_{\text{SP}}(\cdot|s_0)}\left[\sum_{j=0}^{\infty} \nabla_{\theta^{-i}} \log\left(\pi^{-i}_{\theta^{-i}}(a_j^{-i}|s_j)\right) F^{\gamma}_{j:\infty}(\tau)\right] \\ &- \mathbb{E}_{\tau \sim \mathbb{P}_{\text{XP}}(\cdot|s_0)}\left[\sum_{j=0}^{\infty} \nabla_{\theta^{-i}} \log\left(\pi^{-i}_{\theta^{-i}}(a_j^{-i}|s_j)\right) F^{\gamma}_{j:\infty}(\tau)\right]\Bigg], \end{aligned} \tag{20}$$

where $\tau = \langle (s_t, a_t^{-i}, a_t^{\text{partner}}) \rangle_{t=0}^{\infty}$ denotes an interaction trajectory between the teammate and a partner, and $F_{a:b}^{\gamma}(\tau) = \sum_{t=a}^{b} \gamma^t R(s_t, a_t)$ denotes the discounted return accumulated between timesteps a and b along trajectory τ. The distributions $\mathbb{P}_{\text{SP}}(\cdot|s_0)$ and $\mathbb{P}_{\text{XP}}(\cdot|s_0)$ are distributions over trajectories that begin at the given starting state s_0, generated by rolling out π^{-i} alongside π^{BR} (self-play) and π^{ego} (cross-play), respectively:

$$\mathbb{P}_{\text{SP}}(\tau|s_0) = \prod_{t=0}^{\infty} \left(\pi^{-i}(a_t^{-i}|s_t)\, \pi^{\text{BR}}(a_t^{\text{BR}}|s_t)\, P(s_{t+1}|s_t, a_t^{-i}, a_t^{\text{BR}}) \right),$$
$$\mathbb{P}_{\text{XP}}(\tau|s_0) = \prod_{t=0}^{\infty} \left(\pi^{-i}(a_t^{-i}|s_t)\, \pi^{\text{ego}}(a_t^{\text{ego}}|s_t)\, P(s_{t+1}|s_t, a_t^{-i}, a_t^{\text{ego}}) \right). \tag{21}$$

Note that $\mathbb{P}_{\text{SP}}$ and $\mathbb{P}_{\text{XP}}$ are distributions over trajectories, distinct from the state-visitation distributions p_{SP} and p_{XP} defined in Section 4.

Estimating the policy gradient of per-state regret. Exactly evaluating Eq. 20 is intractable in most applications, because it is often not possible to evaluate $\mathbb{P}_{\text{SP}}(\tau|s_0)$ and $\mathbb{P}_{\text{XP}}(\tau|s_0)$ for all plausible trajectories. The gradient is therefore estimated from samples using the following three steps. First, sample starting states from the distribution under which regret is maximized, $s_0 \sim p_{\text{opt}}$. Second, sample trajectories to estimate the two expectation terms on the right-hand side of Eq. 20. The first term is estimated by rolling out π^{-i} with $\text{BR}(\pi^{-i})$ from s_0 to sample from $\mathbb{P}_{\text{SP}}(\cdot|s_0)$, while the second is estimated by rolling out π^{-i} with π^{ego} from s_0 to sample from $\mathbb{P}_{\text{XP}}(\cdot|s_0)$. Third, the log-likelihood and return terms of Eq. 20 are evaluated on the sampled trajectories.

The error of this estimator, relative to the exact gradient, is governed by its bias and variance, which are controlled by the choice of return estimator for $F_{j:\infty}^{\gamma}(\tau)$. Standard choices span the bias-variance spectrum, from the unbiased but high-variance Monte Carlo return-to-go, to the high-bias but low-variance value or advantage network estimators, to the generalized advantage estimator (GAE) [55], whose λ parameter interpolates between these extremes. ROTATE relies on the GAE to estimate the per-state regret, as described in App. F.1.

Assumptions. The derivation of Eq. 20 treats two quantities as fixed with respect to θ^{-i} in order to apply the standard policy gradient theorem, although neither is truly independent of it: the best-response policy π^{BR} and the starting-state distribution p_{opt}. Each assumption is justified below.

Regarding treating π^{BR} as fixed with respect to θ^{-i}: the exact best response depends on θ^{-i} through the best-response operator, but is treated as fixed at each update because it is approximated by a separately trained network (as described in App. F.1). Consequently, Eq. 20 omits the gradient flowing through the best response, introducing bias relative to the gradient of the true regret. Nonetheless, treating the other agents' policies as fixed during each update is the standard assumption underlying independent learning algorithms in multi-agent reinforcement learning, such as independent PPO (IPPO), which achieves strong empirical performance in cooperative tasks [74]. Sun et al. [60] provide theoretical support for this practice, proving that despite the non-stationarity each agent faces when the other agents' policies update concurrently, a monotonic improvement guarantee is retained as long as every agent's policy update remains within a trust region of the joint policy, a constraint that PPO's ratio clipping enforces. Because ROTATE trains the teammate and best-response policies with PPO losses (Algorithm 2), the same mechanism is expected to mitigate the non-stationarity introduced by the concurrently trained π^{BR}, although the formal guarantee of Sun et al. [60] is established for a shared-return objective rather than for regret.

Regarding treating p_{opt} as fixed with respect to θ^{-i}: the starting-state distribution p_{opt} is the key difference between per-state regret maximization and standard return maximization. In the standard setting, trajectories begin at the environment's initial state distribution p_0, which does not depend on the learned policy. By contrast, $p_{\text{opt}} = \mathcal{U}(p_{\text{XP}}, p_{\text{SP}})$ is a mixture of the XP and SP state-visitation distributions, both of which are induced by $\pi_{\theta^{-i}}^{-i}$ and therefore shift whenever θ^{-i} is updated. The objective is thus self-referential, because θ^{-i} appears both in the regret being averaged and in the distribution the average is taken over. Compared to Eq. 20, the exact gradient of this objective contains an additional term capturing the sensitivity of p_{opt} to θ^{-i}.

Although Eq. 20 is not the exact gradient of the per-state regret objective, it may be understood as the exact gradient of a surrogate objective in which p_{opt} is frozen at the distribution induced

by θ^{-i} at the most recent data-collection step. Computing the gradient with respect to a frozen distribution is a common technique in reinforcement learning to simplify the computation of gradients and sometimes stabilize the optimization process. In particular, this idea appears in semi-gradient temporal-difference methods [61], target networks in deep reinforcement learning [40], and the PPO objective [56]. Among these examples, PPO is the closest analogue, because it likewise evaluates its loss on states and actions drawn from a policy-induced distribution frozen at the parameters from the previous data-collection step. Consequently, the trust-region reasoning applied above to π^{BR} is expected to extend to ROTATE's treatment of p_{opt}. Because each update changes θ^{-i} by a bounded amount, and fresh XP and SP data are collected before every round of updates, the frozen distribution is expected to remain close to the p_{opt} distribution that the current θ^{-i} induces.

F ROTATE Pseudocode

This section presents the pseudocode for ROTATE, which follows the open-ended framework specified by Algorithm 1. Section 3 described an open-ended training framework for training an ego agent that can effectively collaborate with previously unseen teammates, which is detailed in Algorithm 1. In the open-ended training framework, a **TeammateGenerator** function determines a buffer of teammate policy parameters, $\text{B}_\pi^{\text{new}}$, that acts as support for the distribution from which teammate policy parameters are sampled. The teammate generator function considers the ego agent's current policy parameters, θ^{ego}, and the previous buffer of teammate policy parameters, B_π. Ideally, the teammate generation function generates and samples teammates that induce learning challenges for π^{ego}. An **EgoUpdate** function specifies a procedure that updates the ego agent's policy parameters based on the $\text{B}_\pi^{\text{new}}$ designed by the teammate generator.

F.1 ROTATE Teammate Generation Algorithm

ROTATE's teammate generation algorithm is detailed in Algorithm 2. As described in Section 4, this teammate generation algorithm jointly trains the parameters of a teammate policy and an estimate of its best response (BR) policy, based on a provided ego agent policy. The two policies optimize distinct objectives. The BR policy performs standard return maximization, while the teammate policy follows the sampled regret policy gradient derived in App. E, augmented with the λ-weighted competence terms of Eq. 8. Accordingly, this subsection first describes the initialization and data collection steps shared by both policies, then defines the three PPO loss functions from which the updates are constructed, and finally presents the BR policy, teammate policy, and critic updates separately.

Initialization of policy and value function networks. The parameters of the teammate and BR policies, θ^{-i} and θ^{BR}, are initialized in Line 3. The parameters of the BR critic network, σ^{BR}, are initialized in Line 4, while those for the teammate, $\sigma^{-i,\text{BR}}$ and $\sigma^{-i,\text{ego}}$, are initialized in Line 5. Note that the teammate maintains two critics, for separately estimating returns when interacting with the BR and ego agent policies.

Data collection. The training of the teammate and BR policies is based on the SP, XP, XSP, and SXP interaction data gathered in Lines 8 and 10 to 12, which Section 4 motivates and describes. Recall that an SXP interaction requires resetting an environment to start from an available XP state, and an XSP interaction analogously requires resetting to an SP state. Since resetting from all available XP states for SXP interaction is impractical, ROTATE *samples* from XP states to obtain start states for SXP interactions (and similarly for XSP). Experiences from SP, XP, SXP, and XSP interaction are stored in buffers $D_{\text{SP}}, D_{\text{XP}}, D_{\text{SXP}}, D_{\text{XSP}}$ in the form of a collection of tuples, $D = \langle (s_k, a_k, r_k, s'_k) \rangle_{k=1}^{|D|}$.

Note that the data collection lines realize the first two steps of the regret policy gradient estimation procedure described in App. E. Specifically, Line 10 realizes the first step by sampling start states from the recently collected buffers, thereby freezing the start-state distribution p_{opt} at the distribution induced by the current policies, as discussed in the assumptions paragraph of App. E. Lines 11 and 12 realize the second step by rolling out the teammate policy alongside the BR and ego agent policies from the sampled states. Finally, Lines 16 to 25 realize the third step by evaluating the policy and critic losses on the stored experiences, as detailed in the remainder of this subsection.

Generalized PPO loss functions. The policy and critic updates of Algorithm 2 are constructed from three loss functions: a generalized PPO clip loss, $\mathcal{L}_{\text{CLIP}}$, an entropy loss, $\mathcal{L}_{\text{ENT}}$, and a value

Algorithm 2 ROTATE TeammateGenerator Function

Input: Environment Env, ego agent policy $\pi_{\theta^{\text{ego}}}$, teammate policy parameter buffer B_π, number of updates N_{updates}, PPO clipping parameter ϵ, PPO update epochs N_{epochs}, competence weights λ_1, λ_2.
Output: Updated teammate buffer B_π.
$\theta^{-i}, \theta^{\text{BR}} \leftarrow$ **RandomInit**(π), **RandomInit**(π)
$\sigma^{\text{BR}} \leftarrow$ **RandomInit**(V)
$\sigma^{-i,\text{BR}}, \sigma^{-i,\text{ego}} \leftarrow$ **RandomInit**(V), **RandomInit**(V) {Init teammate and BR critic parameters}

for $t_{\text{update}} = 1$ **to** N_{updates} **do**
  $D_{\text{SP}} \leftarrow$ **Interact**$(\pi_{\theta^{\text{BR}}}, \pi_{\theta^{-i}}, p_0^{\text{Env}})$
  $D_{\text{XP}} \leftarrow$ **Interact**$(\pi_{\theta^{\text{ego}}}, \pi_{\theta^{-i}}, p_0^{\text{Env}})$
  $s_{\text{XP}} \leftarrow$ **SampleStates**(D_{XP})
  $s_{\text{SP}} \leftarrow$ **SampleStates**(D_{SP}) {Sample XP/SP states}
  $D_{\text{SXP}} \leftarrow$ **Interact**$(\pi_{\theta^{\text{BR}}}, \pi_{\theta^{-i}}, \mathcal{U}(s_{\text{XP}}))$
  $D_{\text{XSP}} \leftarrow$ **Interact**$(\pi_{\theta^{\text{ego}}}, \pi_{\theta^{-i}}, \mathcal{U}(s_{\text{SP}}))$ {Gather SP, XP, SXP, and XSP data}
  $\theta^{\text{BR}}_{\text{old}}, \theta^{-i}_{\text{old}} \leftarrow \theta^{\text{BR}}, \theta^{-i}$
  $\sigma^{\text{BR}}_{\text{old}}, \sigma^{-i,\text{BR}}_{\text{old}}, \sigma^{-i,\text{ego}}_{\text{old}} \leftarrow \sigma^{\text{BR}}, \sigma^{-i,\text{BR}}, \sigma^{-i,\text{ego}}$ {Store old model parameters.}
  for $k_{\text{update}} = 1$ **to** N_{epochs} **do**
    $\mathcal{L}_{\text{BR}}(\theta^{\text{BR}}) \leftarrow$ Eq. 25 {BR policy loss: standard PPO loss on $D_{\text{SP}} \cup D_{\text{SXP}}$}
    $\mathcal{L}_{\text{REG}}(\theta^{-i}) \leftarrow$ Eq. 26 {Teammate policy loss: per-state regret, competence, and entropy}
    $L_V(\sigma^{\text{BR}}) \leftarrow \mathcal{L}_{\text{VAL}}\left(\sigma^{\text{BR}}, \sigma^{\text{BR}}_{\text{old}}, D_{\text{SP}} \cup D_{\text{SXP}}\right)$
    $L_V(\sigma^{-i,\text{BR}}) \leftarrow \mathcal{L}_{\text{VAL}}\left(\sigma^{-i,\text{BR}}, \sigma^{-i,\text{BR}}_{\text{old}}, D_{\text{SP}} \cup D_{\text{SXP}}\right)$
    $L_V(\sigma^{-i,\text{ego}}) \leftarrow \mathcal{L}_{\text{VAL}}\left(\sigma^{-i,\text{ego}}, \sigma^{-i,\text{ego}}_{\text{old}}, D_{\text{XP}} \cup D_{\text{XSP}}\right)$
    $\theta^{\text{BR}} \leftarrow$ **GradDesc** $\left(\theta^{\text{BR}}, \nabla_{\theta^{\text{BR}}} \mathcal{L}_{\text{BR}}(\theta^{\text{BR}})\right)$
    $\theta^{-i} \leftarrow$ **GradDesc** $\left(\theta^{-i}, \nabla_{\theta^{-i}} \mathcal{L}_{\text{REG}}(\theta^{-i})\right)$ {Update policies}
    $\sigma^{\text{BR}} \leftarrow$ **GradDesc** $\left(\sigma^{\text{BR}}, \nabla_{\sigma^{\text{BR}}} L_V(\sigma^{\text{BR}})\right)$
    $\sigma^{-i,\text{BR}} \leftarrow$ **GradDesc** $\left(\sigma^{-i,\text{BR}}, \nabla_{\sigma^{-i,\text{BR}}} L_V(\sigma^{-i,\text{BR}})\right)$
    $\sigma^{-i,\text{ego}} \leftarrow$ **GradDesc** $\left(\sigma^{-i,\text{ego}}, \nabla_{\sigma^{-i,\text{ego}}} L_V(\sigma^{-i,\text{ego}})\right)$ {Update critics.}
  end for
end for
$\text{B}_\pi \leftarrow \text{B}_\pi \cup \langle \theta^{-i} \rangle$ {Add generated teammate policy parameter}
Return: B_π

function loss, $\mathcal{L}_{\text{VAL}}$. The clip loss receives $(\theta, \theta_{\text{old}}, \sigma_{\text{old}}, D, \epsilon, w, \varsigma)$ as input and evaluates

$$\mathcal{L}_{\text{CLIP}}\left(\theta, \theta_{\text{old}}, \sigma_{\text{old}}, D, \epsilon, w, \varsigma\right) = \mathop{\mathbb{E}}_{(s_t, a_t, r_t, s_{t+1}) \in D}\left[-\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\varsigma w_t A_t, \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right)\varsigma w_t A_t\right)\right], \tag{22}$$

where A_t denotes an estimate of the *advantage* function computed from the critic parameterized by σ_{old} via the Generalized Advantage Estimation (GAE) algorithm [55]. The final two arguments generalize the standard PPO clip loss and are introduced solely to express the teammate update: w_t is a per-transition weight applied to the advantage, through which the teammate update implements the per-state regret estimator, and $\varsigma \in \{+1, -1\}$ is a sign applied to the advantage, through which the teammate update implements the subtraction within the regret policy gradient. The entropy loss receives (θ, D) as input and evaluates

$$\mathcal{L}_{\text{ENT}}\left(\theta, D\right) = \mathop{\mathbb{E}}_{(s_t, a_t, r_t, s_{t+1}) \in D}\left[\pi_\theta(a_t|s_t) \log\left(\pi_\theta(a_t|s_t)\right)\right], \tag{23}$$

whose minimization increases policy entropy. With unit weights ($w = \mathbf{1}$) and a positive sign ($\varsigma = +1$), the sum $\mathcal{L}_{\text{CLIP}} + \mathcal{L}_{\text{ENT}}$ is exactly the standard PPO loss [56], whose minimization maximizes returns

while encouraging sufficient exploration. Finally, the value function loss trains a critic network parameterized by σ. The function receives $(\sigma, \sigma_{\text{old}}, D)$ as input and computes the standard mean squared Bellman error (MSBE) loss:

$$\mathcal{L}_{\text{VAL}}\left(\sigma, \sigma_{\text{old}}, D\right) = \mathop{\mathbb{E}}_{(s,a,r',s')\in D}\left[\left(V_\sigma(s) - V_{\sigma_{\text{old}}}^{\text{targ}}(s)\right)^2\right], \tag{24}$$

where $V_{\sigma_{\text{old}}}^{\text{targ}}(s) := A_{\sigma_{\text{old}}}^{\text{GAE}} + V_{\sigma_{\text{old}}}(s)$ is the target value estimate.

Best response policy update. The BR policy is trained by standard return maximization. Accordingly, Line 16 computes the BR loss by instantiating the standard PPO loss with unit weights and a positive sign:

$$\mathcal{L}_{\text{BR}}(\theta^{\text{BR}}) = \mathcal{L}_{\text{CLIP}}\left(\theta^{\text{BR}}, \theta_{\text{old}}^{\text{BR}}, \sigma_{\text{old}}^{\text{BR}}, D_{\text{SP}} \cup D_{\text{SXP}}, \epsilon, \mathbf{1}, +1\right) + \mathcal{L}_{\text{ENT}}\left(\theta^{\text{BR}}, D_{\text{SP}} \cup D_{\text{SXP}}\right). \tag{25}$$

The loss is evaluated on the merged dataset $D_{\text{SP}} \cup D_{\text{SXP}}$, with advantages estimated by the BR critic σ^{BR}. Including the SXP data ensures that the BR policy learns to maximize returns not only from initially reachable states, but also from states visited by the ego agent, so that the BR policy remains a meaningful best response on the start-state distribution over which regret is estimated.

Teammate policy update. The teammate policy update instantiates the sampled estimator of the regret policy gradient (Eq. 20), augmented with the competence terms of Eq. 8. Line 17 computes the teammate loss $\mathcal{L}_{\text{REG}}$, which combines six instances of the clip loss $\mathcal{L}_{\text{CLIP}}$ with a single entropy loss:

$$\begin{aligned}
&\mathcal{L}_{\text{REG}}(\theta^{-i}) = \\
&\underbrace{\ell\left(D_{\text{SP}}, \sigma_{\text{old}}^{-i,\text{BR}}, w_{\text{ts}}, +1\right) + \ell\left(D_{\text{SXP}}, \sigma_{\text{old}}^{-i,\text{BR}}, \mathbf{1}, +1\right)}_{\text{SP side of the regret gradient (Eq. 20)}} \\
&+ \underbrace{\ell\left(D_{\text{XP}}, \sigma_{\text{old}}^{-i,\text{ego}}, w_{\text{ts}}, -1\right) + \ell\left(D_{\text{XSP}}, \sigma_{\text{old}}^{-i,\text{ego}}, \mathbf{1}, -1\right)}_{\text{XP side of the regret gradient (Eq. 20)}} \\
&+ \underbrace{\lambda_2\,\ell\left(D_{\text{SP}}, \sigma_{\text{old}}^{-i,\text{BR}}, w_{\text{ts}}, +1\right) + \lambda_1\,\ell\left(D_{\text{SXP}}, \sigma_{\text{old}}^{-i,\text{BR}}, \mathbf{1}, +1\right)}_{\text{competence terms of Eq. 8}} \\
&+ \mathcal{L}_{\text{ENT}}\left(\theta^{-i}, D_{\text{SP}} \cup D_{\text{XP}} \cup D_{\text{SXP}} \cup D_{\text{XSP}}\right),
\end{aligned} \tag{26}$$

where $\ell\left(D, \sigma, w, \varsigma\right)$ abbreviates $\mathcal{L}_{\text{CLIP}}\left(\theta^{-i}, \theta_{\text{old}}^{-i}, \sigma, D, \epsilon, w, \varsigma\right)$, and the timestep weight w_{ts} is defined below. The first four terms of Eq. 26 form a surrogate loss whose gradient is the sampled regret policy gradient of Eq. 20, with two standard substitutions. First, the GAE replaces the return estimate $F_{j:\infty}^{\gamma}(\tau)$, trading variance for bias as discussed in App. E. Second, the PPO clipped surrogate replaces the vanilla score-function estimator, encouraging each update to remain within a trust region [56].

The structure of the four regret terms mirrors the structure of Eq. 20, which expresses the regret gradient as the difference of two standard policy gradients, namely an SP side, in which the teammate is partnered with π^{BR}, and an XP side, in which the teammate is partnered with π^{ego}. The sign argument ς carries the minus sign of this difference. Because the same teammate log-probability ratio multiplies the return estimates of both terms of Eq. 20, negating the advantages of the XP-side terms ($\varsigma = -1$) implements the subtraction of the XP-side policy gradient. Consistent with this decomposition, the XP-side terms use advantages estimated by the ego-interaction critic $\sigma^{-i,\text{ego}}$, while the SP-side terms use advantages estimated by the BR-interaction critic $\sigma^{-i,\text{BR}}$. Furthermore, each side of the regret gradient is evaluated over start states drawn from the mixture $p_{\text{opt}} = \mathcal{U}(p_{\text{XP}}, p_{\text{SP}})$, so each side contributes one loss term per start-state distribution, yielding the four regret terms of Eq. 26. Table 10 summarizes this correspondence.

The dataset and weight in each cell of Table 10 follow from how the corresponding expectation is estimated. We use two different strategies for estimating the diagonal and off-diagonal cells of the table; each is described below.

Consider first the two off-diagonal cells, in which the rollout partner matches the origin of the start-state distribution: the XP side evaluated over start states drawn from p_{XP}, and the SP side evaluated

Table 10: Correspondence between the four regret terms of Eq. 26 and the two sides of the regret policy gradient (Eq. 20), evaluated over the two start-state distributions composing p_{opt}.

	Start states $\sim p_{\text{XP}}$	Start states $\sim p_{\text{SP}}$
SP side ($\varsigma = +1$, partner π^{BR}, critic $\sigma^{-i,\text{BR}}$)	D_{SXP}, weight $\mathbf{1}$	D_{SP}, weight w_{ts}
XP side ($\varsigma = -1$, partner π^{ego}, critic $\sigma^{-i,\text{ego}}$)	D_{XP}, weight w_{ts}	D_{XSP}, weight $\mathbf{1}$

over start states drawn from p_{SP}. Focusing on the XP cell, the estimated quantity corresponds to the XP side of Eq. 20, restricted to the p_{XP} half of the mixture p_{opt}:

$$g_{\text{XP}} := -\mathbb{E}_{s_0 \sim p_{\text{XP}}} \mathbb{E}_{\tau \sim \mathbb{P}_{\text{XP}}(\cdot|s_0)} \left[\sum_{j=0}^{\infty} \nabla_{\theta^{-i}} \log \left(\pi^{-i}_{\theta^{-i}}(a_j^{-i} \mid s_j) \right) F^{\gamma}_{j:\infty}(\tau) \right]. \tag{27}$$

Estimating g_{XP} requires start states sampled from p_{XP} and, for each sampled start state, an XP rollout continuing from that state. The dataset D_{XP} supplies both requirements at once. First, by the definition of the state-visitation distribution, the state s_t visited at the t-th timestep of an XP episode is a sample from p_{XP}. Second, because the episode continues from s_t under the same partner π^{ego}, the remainder of the episode serves as the rollout for s_t, so no additional rollouts are needed. Treating each visited state s_t of an episode as a start-state sample may therefore contribute one copy of the sum inside Eq. 27, taken over the transitions at timesteps $j \geq t$ of that episode. Consequently, averaging these copies over the start-state samples replaces the two expectations of Eq. 27 with a double sum of terms, $\sum_t \sum_{j \geq t}$, in which the outer sum ranges over the start-state samples s_t and the inner sum ranges over the transitions of the rollout beginning at s_t. Replacing the return estimate $F^{\gamma}_{j:\infty}(\tau)$ with the GAE, the resulting estimator of g_{XP} is:

$$\hat{g}_{\text{XP}} = -\sum_{t=0}^{\infty} \sum_{j=t}^{\infty} \left[\nabla_{\theta^{-i}} \log \left(\pi^{-i}_{\theta^{-i}}(a_j^{-i} \mid s_j) \right) A^{\text{GAE}}_{\sigma^{-i,\text{ego}}_{\text{old}}}(s_j, a_j) \right]. \tag{28}$$

Based on the observation that the tth transition in an XP trajectory appears $t+1$ times in Eq. 28, and leveraging the finite-horizon nature of the experimental tasks, we can collect terms to express the estimator as a single sum over timesteps, in which each transition is weighted by its timestep:

$$\hat{g}_{\text{XP}} \approx -\mathop{\mathbb{E}}_{\tau \sim D_{\text{XP}}} \left[\sum_{t=0}^{\infty} (t+1) \cdot A^{\text{GAE}}_{\sigma^{-i,\text{ego}}_{\text{old}}}(s_t, a_t) \nabla_{\theta^{-i}} \log \pi_{\theta^{-i}}(a_t \mid s_t) \right]. \tag{29}$$

Thus, for each zero-indexed transition, the weight w_{ts} is set to $t+1$. The SP-side cell on D_{SP} follows analogously, with π^{BR} as the rollout partner and a positive sign.

In contrast, estimating a side of the regret gradient on start states drawn from the *other* side's visitation distribution requires fresh rollouts with the corresponding partner, which is precisely the role of the SXP and XSP data. Each such branch rollout estimates the value of only its single start state, so the D_{SXP} and D_{XSP} terms receive unit weight $\mathbf{1}$.

The two λ-weighted terms of Eq. 26 maximize the λ-weighted returns of the teammate when partnered with the BR policy, enforcing the competence constraint of Eq. 8. Because these competence terms coincide with the SP-side regret terms up to their coefficients, collecting terms shows that the D_{SP} and D_{SXP} losses enter Eq. 26 with total coefficients $(1+\lambda_2)$ and $(1+\lambda_1)$, in which the unit portion belongs to the regret surrogate and the λ portion enforces competence. Finally, the entropy loss enters $\mathcal{L}_{\text{REG}}$ exactly once, evaluated over all four interaction datasets, rather than once per $\mathcal{L}_{\text{CLIP}}$ instance. Line 22 then updates the teammate policy parameters via applying gradient descent on Eq. 26.

Value function updates. Lines 18 to 20 train the three critic networks by evaluating the value function loss $\mathcal{L}_{\text{VAL}}$ of Eq. 24, which measures returns from the interaction between the generated teammate policy and its best response or ego agent policy. Each critic is trained on the same interaction data for which it supplies advantage estimates in the policy losses. Specifically, the BR critic σ^{BR} and the teammate's BR-interaction critic $\sigma^{-i,\text{BR}}$ are trained on $D_{\text{SP}} \cup D_{\text{SXP}}$, while the teammate's ego-interaction critic $\sigma^{-i,\text{ego}}$ is trained on $D_{\text{XP}} \cup D_{\text{XSP}}$.

Algorithm 3 ROTATE EgoUpdate Function

1: **Input:** Environment Env, ego policy parameters θ^{ego}, teammate policy buffer B_π, number of updates N_{updates}, PPO clipping parameter ϵ, PPO update epochs N_{epochs}.
2: **Output:** Updated ego policy θ^{ego}.
3: $\sigma^{\text{ego}} \leftarrow$ **Init**(V) {Initialize critic for ego policy}
4: **for** $t_{\text{update}} = 1$ **to** N_{updates} **do**
5: $\quad\theta^{-i} \sim \mathcal{U}(\text{B}_\pi)$ {Sample teammate parameters uniformly}
6: $\quad D \leftarrow$ **Interact**$(\pi_{\theta^{-i}}, \pi_{\theta^{\text{ego}}}, p_0^{\text{Env}})$
7: $\quad\theta^{\text{ego}}_{\text{old}}, \sigma^{\text{ego}}_{\text{old}} \leftarrow \theta^{\text{ego}}, \sigma^{\text{ego}}$
8: $\quad$**for** $k_{\text{update}} = 1$ **to** N_{epochs} **do**
9: $\quad\quad\mathcal{L}_{\text{EGO}}(\theta^{\text{ego}}) \leftarrow \mathcal{L}_{\text{CLIP}}\left(\theta^{\text{ego}}, \theta^{\text{ego}}_{\text{old}}, \sigma^{\text{ego}}_{\text{old}}, D, \epsilon, \mathbf{1}, +1\right) + \mathcal{L}_{\text{ENT}}\left(\theta^{\text{ego}}, D\right)$ {Ego policy loss}
10: $\quad\quad L_V(\sigma^{\text{ego}}) \leftarrow \mathcal{L}_{\text{VAL}}\left(\sigma^{\text{ego}}, \sigma^{\text{ego}}_{\text{old}}, D\right)$ {Ego critic loss}
11: $\quad\quad\theta^{\text{ego}} \leftarrow$ **GradDesc** $\left(\theta^{\text{ego}}, \nabla_{\theta^{\text{ego}}}\mathcal{L}_{\text{EGO}}(\theta^{\text{ego}})\right)$ {Update policy}
12: $\quad\quad\sigma^{\text{ego}} \leftarrow$ **GradDesc** $\left(\sigma^{\text{ego}}, \nabla_{\sigma^{\text{ego}}}L_V(\sigma^{\text{ego}})\right)$ {Update critic}
13: $\quad$**end for**
14: **end for**
15: **Return:** θ^{ego}

The previously defined loss functions can be minimized using any gradient descent-based optimization technique, as we indicate in Lines 21 to 25. In practice, our implementation uses the Adam optimization technique [28]. At the end of this teammate generation process, Lines 28 and 29 indicate how the generated teammate policy parameter is added to a storage buffer, which is subsequently uniformly sampled to provide teammate policies for ego agent training.

F.2 ROTATE Ego Agent Update Algorithm

The ego agent policy's training process proceeds according to Algorithm 3. Line 5 illustrates how ROTATE creates different teammate policies by uniformly sampling model parameters from the B_π resulting from the teammate generation process. Using the experience collaborating with the sampled policies outlined in Line 6, the ego agent's policy parameters are updated to maximize its returns via PPO in Line 9. The ego policy loss $\mathcal{L}_{\text{EGO}}$ reuses the $\mathcal{L}_{\text{CLIP}}$ and $\mathcal{L}_{\text{ENT}}$ functions of App. F.1 with unit weights and a positive sign, since the ego agent performs standard return maximization from the environment's initial state distribution. The single difference from the teammate generation setting is that the input dataset, D, stores the historical sequence of observed states and executed actions, h, rather than states. Likewise, the critic loss $\mathcal{L}_{\text{VAL}}$ in Line 10 is assumed to be computed on the observation-action history rather than states. Like recent AHT learning algorithms [77, 48, 46], π^{ego} and V^{ego} are conditioned on the ego agent's observation-action history to facilitate an adaptive π^{ego} through an improved characterization of teammates' policies. The history-conditioned ego architecture and other practical implementation details are described in App. I. Because the population buffer grows over the course of training, the ego agent faces a nonstationary training distribution. Experimentally, the ego agent learns effectively against this nonstationary distribution only with a lower entropy coefficient and learning rate than those used to train the teammate and best response agents (typically on the order of 1×10^{-4} for the entropy coefficient and 1×10^{-5} for the learning rate). Finally, the ego agent update function returns the updated ego agent policy parameters, which are provided as part of the inputs for the next call to ROTATE's teammate generation function.

G Baselines Overview

The main paper compares ROTATE to six baselines: PAIRED, Minimax Return, FCP, BRDiv, CoMeDi, and COLE. Each baseline is briefly described below, followed by a discussion of the computational complexity of teammate generation baselines compared to ROTATE, and a discussion of the relationship of Mixed Play (MP) with per-state and per-trajectory regret. Implementation details are described in App. I.

PAIRED [19]: A UED algorithm in which a regret-maximizing "adversary" agent proposes environment variations that an allied antagonist achieves high returns on, but a protagonist agent receives low returns on. The algorithm is directly applicable to AHT by defining a teammate generator for the role of the adversary, a best response agent to the generated teammate for the role of the antagonist, and an ego agent for the role of the protagonist. In contrast to ROTATE, our PAIRED extension jointly trains the ego agent, teammate, and BR. The teammate is optimized to maximize the per-trajectory regret of the ego agent. Following the original PAIRED, which does not maintain a population of environments, our extension does not use a population buffer, so the ego agent trains only against the most recently generated teammate.

Minimax Return [42]: The Minimax Return baseline is based on the minimax return baseline of the PAIRED paper [19], with the original idea due to Morimoto and Doya [42]. It uses the same open-ended framework as ROTATE, but generates teammates that minimize the ego agent's *return* rather than maximizing its regret. Note that the Minimax Bayes algorithm of Villin et al. [66] is not our Minimax Return baseline. While conceptually related, it instead addresses how to *curate* a pre-existing population of teammates to best train an ego learner.

Fictitious Co-Play [59]: A two-stage AHT algorithm in which a pool of teammates is generated by running IPPO [74] with varying seeds, and saving multiple checkpoints to the pool. The ego agent is an IPPO agent that is trained against the pool.

BRDiv [49]: A two-stage AHT algorithm in which a population of "confederate" and best-response agent pairs is generated, and an ego agent is trained against the confederates. BRDiv maintains a cross-play matrix containing the returns for all confederate and best-response pairs. The diagonal returns (self-play) are maximized, while the off-diagonal returns (cross-play) are minimized. BRDiv and LIPO [11] share a similar objective, with three main differences: (1) if `xp_weight` denotes the weight on the XP return, then BRDiv requires that the coefficient on the SP return is always $1 + 2 * \texttt{xp_weight}$, (2) LIPO introduces a secondary diversity metric based on mutual information, and (3) LIPO assumes that agents within a team (i.e., a confederate-BR pair) share parameters.

CoMeDi [54]: CoMeDi is a two-stage AHT algorithm. In the first stage, a population of teammates is generated, and in the second stage, an ego agent is trained against the teammate population. The teammate generation stage trains teammate policies one at a time, in which the nth teammate policy is trained to maximize its SP return, minimize its XP return with the previously generated teammate (i.e., from among teammates $1, \cdots, n-1$) that it best collaborates with, and maximize its "mixed-play" (MP) return. The relationship between the regret objectives described in Section 5 and MP is further discussed in App. D.2.

COLE [33]: Cooperative Open-Ended Learning (COLE) is a population-based framework for AHT that generates teammates iteratively, similar to CoMeDi. At each iteration, a new policy is trained to maximize returns against a distribution over the current population of strategies (policies), and then added to the population. The distribution is based on a cooperative incompatibility score, which measures how difficult each policy is to coordinate with. The cooperative incompatibility of each policy is quantified using a Shapley value metric to determine the contribution of each policy to the overall incompatibility of the population. By training each policy against the least compatible policies in the population, COLE encourages the generation of teammates that are more compatible with a wider range of partners. At the end of the teammate generation process, the teammate with the highest average cross-play return across the population is selected and returned as the final ego agent.

G.1 Computational Complexity of ROTATE versus Teammate Generation Baselines

The computational complexity of ROTATE is compared with that of the teammate generation baselines, in terms of the population size and the number of objective updates. In the following, n denotes the population size, while T indicates the number of updates needed to train an individual population member. The precise meaning of n and T might vary with the algorithm, but is made clear in each description.

FCP: Let T denote the number of RL updates needed to train each IPPO team and let n denote the number of teams trained by FCP. Then, the computational complexity of FCP is $O(nT)$.

BRDiv/LIPO: Both BRDiv [49] and LIPO [11] require sampling trajectories from each pair of agents in the population, for each update. Thus, if the total number of updates is T and the population size is n, then the algorithm's time complexity is $O(n^2T)$. Due to the quadratic complexity in n, BRDiv and LIPO are typically run with smaller population sizes, with $n < 10$ for all non-matrix game tasks in both original papers.

CoMeDi: Recall that CoMeDi trains population members one at a time, such that each agent is distinct from the previously discovered teammates in the population. This necessitates performing evaluation rollouts of the currently trained agent against all previously generated teammates at each RL update step. Let T be the number of RL updates required to train the ith agent to convergence, and let n denote the population size. Then CoMeDi's time complexity is $O(n^2T)$—making it scale quadratically in n, similar to BRDiv and LIPO.

COLE: COLE trains a new teammate policy to convergence at each iteration of open-ended learning. The cooperative incompatibility score is based on the cross-play returns of the current population, which requires evaluating each new teammate against all previously generated teammates. Thus, similar to CoMeDi, the time complexity of COLE is $O(n^2T)$, where n is the population size and T is the number of RL updates required to train each teammate to convergence.

ROTATE: In ROTATE, a new teammate is trained to convergence for each iteration of open-ended learning. Thus, the number of open-ended learning iterations is equal to the population size n, where within each iteration, there are $O(T)$ RL updates performed. Therefore, the complexity of ROTATE is $O(nT)$, meaning that our method scales linearly in the population size n.

H Experimental Tasks

Experiments in the main paper are conducted on JAX re-implementations of Level-Based Foraging (LBF) [2, 7], five tasks from the Overcooked suite (Cramped Room (CR), Asymmetric Advantages (AA), Counter Circuit (CC), Coordination Ring (CoR), and Forced Coordination (FC)) [10, 53] and a sabotage matrix game. The JAX implementations of LBF and Overcooked are provided by Jumanji and JaxMARL, respectively, both released under the Apache-2.0 license. The human gameplay data used for the human proxy evaluation (App. C.2) is distributed in the MIT-licensed Overcooked-AI repository [10]. The sabotage matrix game is already fully described in the main paper. The other tasks are described below.

Level-Based Foraging (LBF) Originally introduced by Albrecht and Ramamoorthy [2], Level-Based Foraging is a cooperative logistics problem in which N players interact within a rectangular grid world to obtain k foods. All players and foods have a positive integer *level*, where groups of one to four players may only *load* (collect) a food if the sum of player levels is at least the food's level. A food's level is configured so that it is always possible to load it.

We use the JAX re-implementation of LBF by Bonnet et al. [7], which was based on the implementation by Christianos et al. [14]. The implementation permits the user to specify the number of players, number of foods, grid world size, level of observability, and whether to set the food level equal to the sum of player levels in order to force players to coordinate to load each food.

The experiments in this paper configured the LBF environment to a 7×7 grid, in which two players interact to collect three foods. Our LBF configuration is shown in Fig. 18. Each player observes the full environment state, allowing each player to observe the locations of other agents and all foods and the number of time steps elapsed in the current episode. Each player has six discrete actions: up, down, left, right, no-op, and *load*, the last of which is the special food collection action. A food may only be collected if the sum of player levels is at least the level of the food. Since this paper focuses on fully cooperative scenarios, we set the food level equal to the sum of

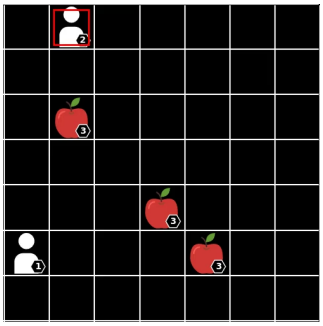


Figure 18: Level-based foraging environment. The apple icons denote food. The number on the icon indicates each player's and food's level. The AHT player is indicated by the red box.

the levels of both players, so all foods require cooperation in order to be collected. When a food is collected, both players receive an identical reward, which is normalized such that the maximum return in an episode is 0.5. Raw LBF returns are converted to the percent of food eaten, reported elsewhere in this paper, by multiplying by 200, so a return of 0.5 corresponds to 100% of food eaten. An episode terminates if an invalid action is taken, if players collide, or when 100 time steps have passed. Player and food locations are randomized for each episode.

Overcooked Introduced by Carroll et al. [10], the Overcooked suite is a set of two-player collaborative cooking tasks, based on the commercially successful Overcooked video game. Designed to study human-AI collaboration, the original Overcooked suite consists of five simple environment *layouts*, in which two agents collaborate within a grid world kitchen to cook and deliver onion soups. While Carroll et al. [10] introduced Overcooked to study human-AI coordination, Overcooked has become popularized for AHT research as well [11, 54, 21].

We use the JAX re-implementation of the Overcooked suite by Rutherford et al. [53], which is based on the original implementation by Carroll et al. [10]. Later versions of Overcooked include features such as multiple dish types, order lists, and alternative layouts, but this paper considers only the five original Overcooked layouts: Cramped Room (CR), Asymmetric Advantages (AA), Counter Circuit (CC), Coordination Ring (CoR), and Forced Coordination (FC).

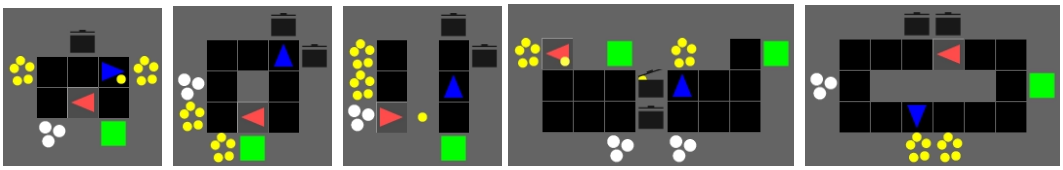
The objective for all five tasks is to deliver as many onion soups as possible, where the only difference between the tasks is the environment layout, as shown in Fig. 19. To deliver an onion soup, players must place three onions in a pot to cook, use a plate to pick up the cooked soup, and send the plated soup to the delivery location. Each player observes the state and location of all environment features (counters, pots, delivery, onions, and plates), the position and orientation of both players, and an urgency indicator, which is 1 if there are 40 or fewer remaining time steps, and 0 otherwise. Each player has six discrete actions, consisting of the four movement actions, interact, and no-op. The reward function awards both agents $+20$ upon successfully delivering a dish, which is the return reported in the experimental results. To improve sample efficiency, all algorithms are trained using a shaped reward function that provides each agent an additional reward of 0.1 for picking up an onion, 0.5 for placing an onion in the pot, 0.1 for picking up a plate, and 1.0 for picking up a soup from the pot with a plate. An episode terminates after 400 time steps. Player locations are randomized in each episode. In divided layouts such as AA and FC, we ensure that a player is spawned on each half of the layout.

H.1 Optimal Behavior in Asymmetric Advantages

Optimal behavior in AA still requires effective coordination due to layout asymmetry. In the “left" kitchen, the delivery zone is adjacent to the pots while the onions are farther, while in the “right" kitchen, the opposite is true. Thus, an optimal team consists of the “left" agent delivering finished soup, and the “right" agent placing onions in the pots—and indeed, we observe that teams of IPPO agents converge to this behavior. In the main paper, we observed that Minimax Return performed surprisingly well on this task. Visualizing the trained Minimax Return policies on AA reveals that the adversarial teammate trained by Minimax Return cannot drive the ego agent’s return to zero, and does not prevent the ego agent from learning how to perform the task independently.

I Implementation Details

As implementations of prior methods use PyTorch while this paper uses JAX, we re-implemented all methods in this paper, using PPO [56] with Generalized Advantage Estimation (GAE) [55] as a base RL algorithm and Adam [28] as the default optimizer. The code for this paper is released for reproducibility at `https://github.com/carolinewang01/rotate`. A detailed discussion of ROTATE’s losses, update procedure, and pseudocode is provided in App. F. This section discusses implementation details such as training time choices, agent architectures, and key hyperparameters for ROTATE and all baselines.



(a) Cramped Room. (b) Coord. Ring. (c) Forced Coord. (d) Asymmetric Advantages. (e) Counter Circuit.

Figure 19: The five classic Overcooked layouts. Each yellow circle is an onion, while white circles are plates. Grid spaces with multiple yellow (resp. white) circles are onion (resp. plate) piles, which agents must visit to pick up an onion (or plate). The green square is the delivery location, where finished dishes must be sent to receive a reward. Black squares denote free space, while adjacent gray spaces are empty counters. A black pot icon indicates pots, while agents are shown as red and blue pointers. The AHT agent is highlighted.

I.1 Training Compute

For fair comparison, every method was trained with the same total environment-interaction budget on each task, counting both the teammate generation stage and ego agent training: 360M steps on LBF, 240M steps on CR and AA, and 440M steps on CoR, CC, and FC. This budget applies to all methods except for FCP, whose population consists of independently trained IPPO agents. Rather than scaling its population to consume the shared budget, we used a similar population size to the original paper (ours is $\sim$110 agents per task, while the original used 96 agents per task). Population sizes for the remaining two-stage baselines were chosen as follows. BRDiv was trained with a population size of 3-4 agents. While we attempted training BRDiv with a larger population size, the algorithm was prone to discovering degenerate solutions in which only 2-3 agents in the population could discover solutions with high SP returns, and all other agents in the population would have zero returns. CoMeDi was trained with a population size of 10 agents. We attempted to train CoMeDi with a larger population size, but due to the algorithm's quadratic complexity in the population size, its runtime surpassed the available time budget. Nevertheless, the population size of 10 forms a reasonable comparison to ROTATE because (1) the original paper used a population size of 8 for all Overcooked tasks, and (2) the configuration of CoMeDi in this paper runs for a similar wall-clock time as ROTATE.

I.2 Compute infrastructure

Experiments were performed on two servers, each with the following specifications:

- **CPUs**: two Intel(R) Xeon(R) Gold 6342 CPUs, each with 24 cores and two threads per core.
- **GPUs**: four NVIDIA A100 GPUs, each with 81920 MiB VRAM.

The experiments in this paper were implemented in JAX and parallelized across seeds. On the servers above, each method took approximately 4-6 hours of wall-clock time to run.

I.3 Agent Architectures

For all methods considered in this paper, agents are implemented using neural networks and an actor-critic architecture, as is standard for PPO-based RL algorithms. All AHT methods implement policies without parameter sharing [14], to enable greater behavioral diversity. Specifics for ego agents, teammates, and best response agents are described below.

As mentioned in the main paper, ego agents are history-conditioned. Thus, ego agents are implemented with the S5 actor-critic architecture, a recently introduced recurrent architecture shown to have stronger long-term memory than prior types of recurrent architectures. Another advantage of the S5 architecture over typical recurrent architectures (e.g. LSTMs) is that it is parallelizable during training, allowing significant speedups in JAX [36].

On the other hand, teammates and best response agents are state-based. Best response agents are implemented with fully connected neural networks. Teammates are also based on fully connected neural networks, but the precise architecture varies based on the algorithm. For methods in which

the teammate only interacts with itself (FCP) or with the ego agent (Minimax Return), a standard actor-critic architecture is used. However, for open-ended learning methods that optimize regret (ROTATE and PAIRED), or for teammate generation methods that optimize adversarial diversity (CoMeDi and BRDiv), teammates must estimate returns when interacting with multiple agents. Thus, for these methods, the teammate architecture includes a critic for each type of interaction.

For ROTATE and PAIRED, the teammate must estimate returns when interacting with the ego agent and its best response, and so it maintains a critic network for each partner type. For CoMeDi and BRDiv, given a population with n agents, each teammate must estimate the return when interacting with the other $n-1$ agents in the population. As it would be impractical to maintain $n-1$ critics for each teammate, the teammate instead uses a critic that conditions on the agent ID of a candidate partner agent—in effect, implementing the $n-1$ critics via *parameter sharing* [14].

Table 11: Hyperparameters for IPPO.

Task	LBF	CR	AA	CC	CoR	FC
Timesteps	**5e5**, 1e6	1e6	1e6	1e6, **3e6**	**3e6**	1e6, **3e6**, 1e7
Number envs	**8**, 16	**8**, 16	8	**8**, 16	8	8
Epochs	7, **15**	15	15	**15**, 30	15	15
Minibatches	**4**, 8	4, 8, **16**, 32	16	16	16	16
Clip-Eps	**0.03**, 0.05	0.03, 0.05, 0.10, 0.15, **0.2**, 0.3	0.2, **0.3**	**0.1**, 0.2	**0.1**, 0.2, 0.3	**0.1**, 0.2
Ent-Coef	5e-3, **0.01**, 0.03, 0.05	5e-3, **0.01**, 0.03, 0.05	**0.01**, 0.02	0.01, 0.03, **0.05**	0.001, 0.01, **0.05**	0.01, **0.05**
LR	1e-4	1e-4	**1e-4**, 1e-3	1e-4, **1e-3**	1e-4, 5e-4, **1e-3**	1e-4, 5e-4, **1e-3**
Anneal LR	**true**, false	**true**, false	true	**true**, false	true	true

Table 12: Hyperparameters for the teammate generation stage of BRDiv.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps	—	3.3e8	2.1e8	2.1e8	3.8e8	3.8e8	3.8e8
XP Coefficient	0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1	**0.1**	**0.5**	**0.5**	**0.01**	**0.005**	**0.01**
Population size	—	**3**, 4, 5, 10	2, 3, **4**, 5	**3**, 4	**3**, 4	**3**, 4	**3**, 4
Num Envs	—	64	64	64	128	128	128
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	**5e-5**	**1e-5**	**1e-3**	**1e-5**	**1e-5**	**5e-4**
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	**0.005**	**0.05**	**0.05**	**0.01**	**0.05**	**0.01**
Clip-Eps	0.05, 0.1, 0.2, 0.3, 0.5	**0.2**	**0.3**	**0.05**	**0.01**	**0.1**	**0.05**

I.4 Hyperparameters

This section presents the hyperparameters for ROTATE (Table 13), baseline methods (Tables 12 and 14 to 19), and IPPO (Table 11), which served as the base algorithm for all methods and was used to train evaluation teammates. For all methods, hyperparameters were first tuned manually, and then tuned by automated hyperparameter search for the most important hyperparameters. The procedure is described below.

Manual hyperparameter tuning began with the base IPPO hyperparameters for each task (Table 11), typically varying one parameter over a range while holding the others fixed, and varying parameters jointly only when varying one at a time did not yield the desired result. These manually-tuned IPPO settings served as the starting point for the IPPO training hyperparameters used by all algorithms. Next, hyperparameters for the shared ego agent training portion of each teammate generation

Table 13: Hyperparameters for ROTATE. Hyperparameters specific to the teammate training process are marked by "(T)".

	Search Grid	LBF	CR	AA	CC	CoR	FC
OEL Iterations	—	30	30	30	20	20	20
Num Envs	—	16	16	16	16	16	16
λ_1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	**4**	**4**	**9**	**2**	**10**	**3**
λ_2	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10	**4**	**4**	**0**	**4**	**10**	**4**
Minibatches	—	**4**, 8	8	8	8	8	8
Timesteps per Iter (Ego)	—	2e6	2e6	2e6	6e6	6e6	6e6
Epochs (Ego)	—	5, **10**, 20	**10**, 15	10	10	10	**5**, 10
Ent-Coef (Ego)	—	**1e-4**, 1e-3, 0.01, 0.05	1e-4, **1e-3**, 1e-2	**1e-3**, 0.01	**1e-3**, 0.05	**1e-3**, 0.05	**1e-4**, 1e-3, 1e-2
LR (Ego)	—	**5e-5**, 1e-4, 1e-3	1e-5, 3e-5, **5e-5**, 1e-4	1e-5, 3e-5, **5e-5,** 1e-4	3e-5, **5e-5,** 1e-3	1e-5, **3e-5**, 5e-5, 1e-3	8e-6, **1e-5**, 3e-5, 5e-5, 1e-4
Eps-Clip (Ego)	—	0.05, **0.1**	**0.1**, 0.2	**0.1**, 0.3	0.1	0.1	0.1
Anneal LR (Ego)	—	true, **false**	true, **false**	true, **false**	true, **false**	true, **false**	true, **false**
Timesteps per Iter (T)	—	1e7	6e6	6e6	1.6e7	1.6e7	1.6e7
Epochs (T)	—	20	20	20	20	20	20
Ent-Coef (T)	0.005, 0.01, 0.03, 0.05, 0.1	**0.01**	**0.01**	**0.05**	**0.05**	**0.01**	**0.05**
LR (T)	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	**1e-4**	**1e-4**	**5e-5**	**1e-3**	**1e-3**	**1e-3**
Clip-Eps (T)	0.05, 0.1, 0.2, 0.3, 0.5	**0.1**	**0.2**	**0.5**	**0.1**	**0.5**	**0.1**
Anneal LR (T)	—	**true**, false	**true**, false	false	false	false	true, **false**

Table 14: Hyperparameters for PPO ego agent for all teammate generation methods.

	LBF	CR	AA	CC	CoR	FC
Total Timesteps	6e7	6e7	6e7	1.2e8	1.2e8	1.2e8
Num Envs	8	8	8	8	8	8
LR	5e-5	5e-5	5e-5	5e-5	3e-5	1e-5
Epochs	10	10	10	10	10	5
Minibatches	4	8	8	8	8	8
Ent-Coef	1e-4	1e-3	1e-3	1e-3	1e-3	1e-4
Clip-Eps	0.1	0.1	0.1	0.1	0.1	0.1
Anneal LR	false	false	false	false	false	false

Table 15: Hyperparameters for teammate generation stage of FCP.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps Per Agent	—	2e6	2e6	2e6	4e6	4e6	4e6
Num Seeds	—	23	23	23	22	22	22
Num Checkpoints	—	5	5	5	5	5	5
Num Envs	—	8	8	8	8	8	8
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	**1e-5**	**1e-4**	**1e-5**	**1e-3**	**1e-3**	**5e-4**
Epochs	—	15	15	15	15	15	15
Minibatches	—	4	16	16	16	16	16
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	**0.005**	**0.01**	**0.1**	**0.03**	**0.01**	**0.005**
Eps-Clip	0.05, 0.1, 0.2, 0.3, 0.5	**0.5**	**0.2**	**0.2**	**0.05**	**0.3**	**0.2**
Anneal LR	—	true	true	true	true	true	true

Table 16: Hyperparameters for the teammate generation stage of CoMeDi.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps Per Iteration	—	3.3e7	2.1e7	2.1e7	3.8e7	3.8e7	3.8e7
Population Size	—	10	10	10	10	10	10
Num Envs	—	48	48	48	48	48	48
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	**5e-4**	**5e-4**	**5e-5**	**1e-3**	**1e-3**	**5e-4**
Epochs	—	15	15	15	15	15	15
Minibatches	—	8	8	8	8	8	8
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	**0.03**	**0.005**	**0.1**	**0.03**	**0.005**	**0.01**
Eps-Clip	0.05, 0.1, 0.2, 0.3, 0.5	**0.3**	**0.3**	**0.1**	**0.5**	**0.3**	**0.5**
Anneal LR	—	false	false	false	false	false	false
α	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0	**0.6**	**0.4**	**0.4**	**0.1**	**0.7**	**0.3**
β	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0	**1.0**	**0.7**	**0.4**	**0.1**	**0.4**	**0.4**

algorithm (Table 14) were tuned on a *per-task* basis, and held fixed for fair comparison. Finally, the hyperparameters were tuned for each algorithm on a per-task basis, focusing on the teammate generation portion.

The automated hyperparameter search consisted of random search over predefined hyperparameter ranges. For each algorithm-task combination, 15 configurations were sampled uniformly at random, where each was evaluated with 2 seeds at a reduced training budget. The final hyperparameter configuration was selected from the searched configurations and the configuration resulting from manual tuning. The hyperparameters tuned by the random search are the learning rate, entropy coefficient, and PPO clip epsilon common to all methods, together with each method's own diversity or regret coefficients (λ_1, λ_2 for ROTATE, α, β for CoMeDi, and the XP-return coefficient for BRDiv).

For each hyperparameter, the searched values are listed in the tables, and the selected values are bolded. The hyperparameters for the two-stage teammate generation methods are presented in separate tables, where those corresponding to the shared ego agent training stage are presented in

Table 17: Hyperparameters for PAIRED.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps	—	3.6e8	2.4e8	2.4e8	4.4e8	4.4e8	4.4e8
Num Seeds	—	3	3	3	3	3	3
Num Checkpoints	—	10	10	10	10	10	10
Num Envs	—	16	16	16	16	16	16
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	**1e-4**	**5e-4**	**1e-5**	**5e-5**	**1e-4**	**5e-5**
Epochs	—	15	15	15	15	15	15
Minibatches	—	4	8	8	8	8	8
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	**0.03**	**0.005**	**0.05**	**0.01**	**0.005**	**0.1**
Eps-Clip	0.05, 0.1, 0.2, 0.3, 0.5	**0.2**	**0.05**	**0.1**	**0.05**	**0.05**	**0.3**
Anneal LR	—	false	false	false	false	false	false

Table 18: Hyperparameters for Minimax Return. Hyperparameters specific to the teammate training process are marked by "(T)".

	Search Grid	LBF	CR	AA	CC	CoR	FC
OEL Iterations	—	30	30	30	20	20	20
Num Envs	—	16	16	16	16	16	16
Timesteps Per Iter (Ego)	—	2e6	2e6	2e6	6e6	6e6	6e6
Timesteps Per Iter (T)	—	1e7	6e6	6e6	1.6e7	1.6e7	1.6e7
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	**1e-3**	**1e-4**	**1e-5**	**1e-5**	**1e-4**	**5e-5**
Epochs	—	15	15	15	15	15	15
Minibatches	—	4	8	8	8	8	8
Ent-Coef	0.005, 0.01, 0.03, 0.05, 0.1	**0.03**	**0.1**	**0.05**	**0.005**	**0.005**	**0.01**
Eps-Clip	0.05, 0.1, 0.2, 0.3, 0.5	**0.03**	**0.1**	**0.2**	**0.05**	**0.1**	**0.05**
Anneal LR	—	false	true	false	false	false	false

Table 14. All experiments in the paper were performed with a discount factor of $\gamma = 0.99$ and $\lambda^{\text{GAE}} = 0.95$.

J Evaluation Teammates

Evaluation teammates were constructed using three strategies: training IPPO teammates in self-play using varied seeds and reward shaping, training teammates with BRDiv, and manually programming heuristic agents. The evaluation teammates trained using IPPO and BRDiv were trained using different seeds than those used for training ROTATE and baseline methods. The teammate construction procedure results in distinct teammate archetypes, which are summarized in Table 20 and detailed below. The precise set of evaluation teammates used for each task may be read from the axes of the XP matrices shown in fig. 20.

Generally, IPPO agents execute straightforward, return-maximizing strategies. On the other hand, since BRDiv agents are trained to maximize self-play returns with their best response partner and to reduce cross-play returns with all other best response policies in the population, the generated teammates display more adversarial behavior compared to IPPO and heuristics. Coefficients on

Table 19: Hyperparameters for COLE.

	Search Grid	LBF	CR	AA	CC	CoR	FC
Timesteps Per Iteration	—	1.8e7	1.85e7	1.85e7	3.38e7	3.38e7	3.38e7
Population Size	12, 15, 18, 20 (LBF); 13, 15, 17 (CR, CoR)	**20**	**13**	13	13	**13**	13
Num Envs	—	64	64	64	64	64	64
LR	1e-5, 5e-5, 1e-4, 5e-4, 1e-3	**1e-3**	**1e-3**	1e-3	1e-3	**1e-3**	1e-3
Epochs	—	15	15	15	15	15	15
Minibatches	—	8	8	8	8	8	8
Ent-Coef	0.1, 0.05, 0.03, 0.01, 0.005	**0.05**	**0.05**	0.01	0.01	**0.01**	0.01
Eps-Clip	—	0.05	0.05	0.3	0.01	0.1	0.05
Anneal LR	—	false	false	false	false	false	false
α	0.1, 0.5, 1, 1.5	**1.5**	**0.1**	0.1	0.1	**0.1**	0.1
c_{UCT}	0.01, 0.1, 0.2, 0.3, 0.4, 0.5	**0.4**	**0.1**	0.3	0.3	**0.3**	0.3

Table 20: Summary of evaluation teammate types.

Type	Tasks	Description
ippo_mlp (k)	LBF and Overcooked	Teammate trained by IPPO in self-play to maximize return; k indexes the training seed. On CC, ippo_mlp_cc agents navigate counterclockwise around the counter.
ippo_mlp (k, c)	LBF	Intermediate checkpoint of a teammate trained by IPPO, where k indexes the training seed and c the checkpoint.
ippo_mlp_pass (k)	Overcooked	Teammate trained by IPPO with reward shaping to pass onions across the counter (CC); k as above.
brdiv-conf$j$ (k)	LBF and Overcooked	Confederate teammate trained by BRDiv, where j indexes the BRDiv training run and k the member of its confederate population.
independent_agent_p	Overcooked	Programmed agent that cooks and delivers soups on its own. The parameter p is the probability that the agent places a held item on a nearby counter.
onion_agent_p	Overcooked	Programmed agent that places onions in non-full pots; p as above.
plate_agent_p	Overcooked	Programmed agent that plates finished soups and delivers them; p as above.
seq_agent_$*$	LBF	Planning agents that collect food in a fixed order: lexicographic (lexi), reverse lexicographic (rlexi), column-major (col), reverse column-major (rcol), or nearest/farthest-first by Manhattan distance from the agent's initial position (nearest, farthest).

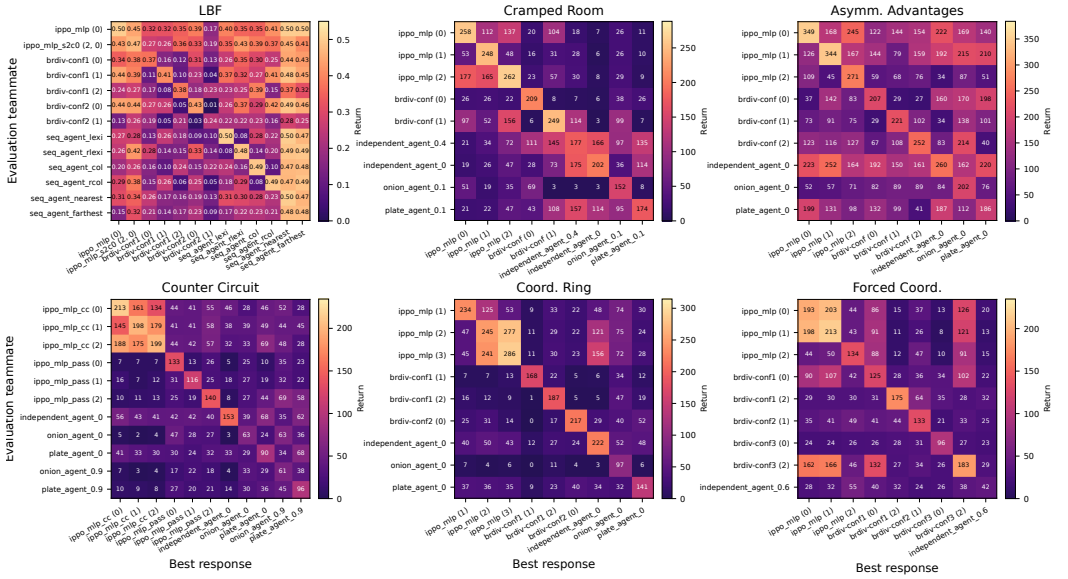


Figure 20: Cross-play matrices among the evaluation teammates and their best responses, per task, in unnormalized return units. Each diagonal entry is the raw empirical return of the teammate paired with its own RL-trained best response; the diagonal/off-diagonal gap evidences the diversity of the evaluation set. The per-teammate upper bounds used for return normalization are reported separately in Table 21.

the SP and XP returns were carefully tuned to ensure that the behavior was not too adversarial, which we operationalized as teammates whose SP returns were high but whose XP returns were near zero. Finally, the manually programmed heuristic agents have a large range of skills and levels of determinism. The LBF heuristics are planning-based agents that deterministically attempt to collect the apples in a specific order. Given a best response partner, the LBF heuristics can achieve the optimal task return in LBF. The Overcooked heuristics execute pre-programmed roles that are agnostic to the layout, along with some basic collision-avoidance logic. The “onion" heuristic collects onions and places them in non-full pots. The “plate" heuristic plates soups that are ready, and delivers them. The “independent" heuristic attempts to fulfill both roles by itself. All three heuristic types have a user-specified parameter that defines the probability that the agent places whatever it is holding on a nearby counter. The feature serves two purposes: first, it creates a larger space of behaviors, and second, it allows the heuristics to work for the FC task, where the agent in the left half of the kitchen must pass onions and plates to the right, while the agent in the right half must pick up resources from the dividing counter, cook soup, and deliver.

To quantify the diversity of Π^{eval}, we compute the cross-play (XP) matrix among evaluation teammates for each task, visualized in fig. 20. Averaging each matrix, the mean diagonal return (where each teammate is paired with its own best response) substantially exceeds the mean off-diagonal return (where teammates are paired with another teammate’s best response). The large gap between diagonal and off-diagonal returns provides evidence that the evaluation teammates employ diverse conventions, so the normalized evaluation return does capture robustness to conventions. Candidate teammates whose diagonal and off-diagonal returns were similar added no diversity and were pruned, which is why the per-source mix of evaluation teammates is uneven across tasks.

J.1 Evaluation Return Normalization Procedure

Normalizing evaluation returns is necessary in AHT because AHT asks whether the ego agent is a good partner *for a given teammate*, relative to what is achievable with that teammate. Because evaluation teammates vary in skill and therefore have different maximum team returns, unnormalized returns conflate ego agent quality with teammate skill. Normalization is also necessary to report returns aggregated across evaluation teammates, and is in line with best-practice recommendations for both RL [1] and AHT [69].

Concretely, each method is evaluated by the min-max normalized team return with each evaluation teammate. The lower return bound is set to zero since a poor teammate could always cause a zero return in all tasks considered. Ideally, the upper return bound for each evaluation teammate would be the return achieved with its theoretically optimal best response partner, but this bound is computationally intractable to obtain. Instead, initial estimates are obtained by training BR policies via RL, and each estimate is then revised upward whenever *any* partner policy exceeds it, including evaluated methods. This correction procedure yields tighter bounds than using the RL-trained best responses alone or the compared methods alone. The resulting per-teammate upper bounds are reported in Table 21.

LBF	
Teammate	Bound
ippo_mlp (0)	100.0
ippo_mlp_s2c0 (2, 0)	96.4
brdiv-conf1 (0)	97.4
brdiv-conf1 (1)	100.0
brdiv-conf1 (2)	89.6
brdiv-conf2 (0)	100.0
brdiv-conf2 (1)	62.5
seq_agent_lexi	100.0
seq_agent_rlexi	100.0
seq_agent_col	100.0
seq_agent_rcol	100.0
seq_agent_nearest	100.0
seq_agent_farthest	100.0

Cramped Room	
Teammate	Bound
ippo_mlp (0)	258.4
ippo_mlp (1)	253.8
ippo_mlp (2)	262.2
brdiv-conf (0)	214.1
brdiv-conf (1)	248.8
independent_agent_0.4	197.2
independent_agent_0	202.2
onion_agent_0.1	151.9
plate_agent_0.1	191.2

Asymm. Advantages	
Teammate	Bound
ippo_mlp (0)	382.5
ippo_mlp (1)	369.4
ippo_mlp (2)	299.7
brdiv-conf (0)	283.4
brdiv-conf (1)	345.3
brdiv-conf (2)	333.1
independent_agent_0	308.1
onion_agent_0	299.4
plate_agent_0	284.7

Counter Circuit	
Teammate	Bound
ippo_mlp_cc (0)	213.1
ippo_mlp_cc (1)	200.9
ippo_mlp_cc (2)	199.4
ippo_mlp_pass (0)	136.6
ippo_mlp_pass (1)	115.9
ippo_mlp_pass (2)	170.0
independent_agent_0	153.4
onion_agent_0	95.0
plate_agent_0	90.0
onion_agent_0.9	91.9
plate_agent_0.9	105.6

Coord. Ring	
Teammate	Bound
ippo_mlp (1)	249.7
ippo_mlp (2)	246.6
ippo_mlp (3)	285.9
brdiv-conf1 (1)	168.4
brdiv-conf1 (2)	187.2
brdiv-conf2 (0)	216.9
independent_agent_0	221.6
onion_agent_0	96.9
plate_agent_0	141.2

Forced Coord.	
Teammate	Bound
ippo_mlp (0)	220.0
ippo_mlp (1)	214.4
ippo_mlp (2)	225.6
brdiv-conf1 (0)	131.6
brdiv-conf1 (2)	184.7
brdiv-conf2 (1)	143.8
brdiv-conf3 (0)	95.6
brdiv-conf3 (2)	183.1
independent_agent_0.6	81.2

Table 21: Per-teammate upper bounds used for min-max normalization of evaluation returns (the lower bound is always 0). Each bound is obtained from the RL-trained best-response return estimate, revised upward whenever any evaluated policy exceeded it. Units are percent of food eaten for LBF and the unshaped base return for the Overcooked tasks.

As described in Section 6, our normalized return metric is a member of the class of BRProx metrics recommended by Wang et al. [69], with the *mean* as the main aggregation metric. The main comparison between ROTATE and baselines is also reported with IQM and median as aggregation functions (Fig. 5).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Contributions are clearly stated in the introduction, with corresponding sections of the paper providing evidence.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The final section of the paper (Discussion and Conclusion) addresses multiple limitations of our paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All theoretical results include full assumptions and complete proofs, provided in App. B.2.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Algorithm pseudocode and implementation details are specified in App. F and App. I, respectively. Baselines, tasks, and evaluation teammates are described in App. G, App. H, and App. J.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes, the code is publicly released at https://github.com/carolinewang01/rotate (App. I), with instructions for reproducing the experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, the appendix provides full training and evaluation details, including implementation details and hyperparameters (App. I and App. I.4), and evaluation teammates and return normalization (App. J).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, all core results are reported with bootstrapped 95% confidence intervals (Sec. 6).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, App. I states our compute infrastructure and the amount of time needed to run experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: No new human-subject data were collected. The human proxy evaluation (App. C.2) uses behavior-cloned policies trained on the public, anonymized dataset of Carroll et al. [10]. The study proposes a novel algorithm for ad hoc teamwork, which is unlikely to have negative social impacts.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: This paper proposes a novel ad hoc teamwork algorithm, that has been validated only on simulated gridworld environments. As foundational algorithmic research, it has no direct path to negative applications.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper does not use generative models or scrape datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All existing assets are cited, and their licenses are stated in App. H: Jumanji (Apache-2.0) for LBF, JaxMARL (Apache-2.0) for Overcooked, and the MIT-licensed Overcooked-AI repository, which distributes the human gameplay data of Carroll et al. [10].

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The only new asset released by this work is the codebase, which is MIT-licensed and documented in the repository README to allow others to reproduce the work.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects. No new human-subject data were collected. The human proxy evaluation (App. C.2) uses behavior-cloned policies trained on the public, anonymized dataset of Carroll et al. [10].

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or research with human subjects. No new human-subject data were collected. The human proxy evaluation (App. C.2) uses behavior-cloned policies trained on the public, anonymized dataset of Carroll et al. [10].

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The paper does not use LLMs in any components of the method.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.