Generating Diverse Cooperative Agents by Learning Incompatible Policies

Anonymous Authors[1]

Abstract

Effectively training a robust agent that can cooperate with unseen agents requires diverse training partner agents. Nonetheless, obtaining cooperative agents with diverse behaviors is a challenging task. Previous work proposes learning a diverse set of agents by diversifying the state-action distribution of the agents. However, without information about the task's goal, the diversified behaviors are not motivated to find other important, albeit non-optimal, solutions, resulting in only local variations of a solution. In this work, we propose to learn diverse behaviors by looking at *policy compatibility* while using state-action information to induce local variations of behaviors. Conceptually, policy compatibility measures whether policies of interest can collectively solve a task. We posit that incompatible policies can be behaviorally different. Based on this idea, we propose a novel objective to learn diverse behaviors. We theoretically show that our novel objective can generate a unique policy, which we incorporate into a population-based training scheme. Empirically, the proposed method outperforms the baselines in terms of the number of discovered solutions given the same number of agents.

1. Introduction

Cooperating with unseen agents in multi-agent systems is a challenging problem. Current state-of-the-art multi-agent reinforcement learning techniques can produce highly competent agents in cooperative settings (Lowe et al., 2017; Sunehag et al., 2017; Mahajan et al., 2019; Peng et al., 2021). Nonetheless, those agents are often overfitted to their training partners and cannot coordinate with unseen agents effectively (Lanctot et al., 2017; Carroll et al., 2019; Bard et al., 2020; Hu et al., 2020).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

The problem of working with unseen partners (i.e., ad-hoc teamwork problem (Stone et al., 2010)) has been tackled in many different ways (Albrecht & Stone, 2018; Grover et al., 2018; Carroll et al., 2019; Shih et al., 2020; Rahman et al., 2021; Strouse et al., 2021; Xie et al., 2021; Mirsky et al., 2022; Parekh et al., 2022; Wang et al., 2022). These methods allow the agents to learn how to coordinate with unseen agents and, sometimes, humans. However, the success of these methods relies heavily on the quality of the pool of training partner agents. It has been shown that the diversity of the training partners is crucial to the generalization of cooperative agents (Charakorn et al., 2020; 2021; McKee et al., 2021). In spite of its importance, obtaining a diverse set of partners is still an open problem.

The simplest way to generate training partners is to use handcrafted policies (Leibo et al., 2021; Papoudakis et al., 2021; Xie et al., 2021; Wang et al., 2022) or agents produced from multiple runs of self-play training process (Grover et al., 2018; Charakorn et al., 2020; Strouse et al., 2021). These methods, however, are not scalable nor guaranteed to produce diverse behaviors. Many prior works recognize the importance of diverse training partner agents in the cooperative domain and propose techniques aiming to generate diverse agents by changing the state visitation and action distributions (Lucas & Allen, 2022), or joint trajectory distribution of the agents (Mahajan et al., 2019; Lupu et al., 2021). However, Lupu et al. (2021) discuss a potential drawback of using such information from trajectories to diversify the behaviors. Specifically, agents that make locally different decisions do not necessarily exhibit different high-level behavior.

To avoid this potential pitfall, we propose an alternative way to diversify behaviors using information about the task's objective. In contrast to previous work that uses joint trajectory distribution as a representation of behavior, we use policy compatibility instead. In multi-agent systems, policy compatibility evaluates whether policies of interest can collaboratively complete a task. Since cooperative environments commonly require all agents to *coordinate on the same solution*, if the policies have learned different solutions, then they cannot coordinate effectively. Consequently, if an agent discovers a solution that is incompatible with all other agents in a population, then the solution must be unique relative to the population. Motivated by this intu-

ition, we introduce a novel training objective that regularizes agents in a population to find unique solutions that are compatible with their partner agents while not compatible with any other agents in the population. We call this method *"Learning Incompatible Policies"* (LIPO).

We theoretically show that optimizing the proposed objective will yield a unique policy, which we extend further for a population-based training scheme. Furthermore, we utilize a mutual information objective to diversify the local behaviors of each joint policy. Empirically, LIPO can discover more solutions than previous methods, given the same population size.

2. Preliminaries

Our main focus lies in fully cooperative environments which are modeled as decentralized partially observable Markov decision processes (Dec-POMDP) (Bernstein et al., 2002). In this work, we start our investigation in the two-player variant. A two-player Dec-POMDP is defined by a tuple $(\mathcal{S}, \mathcal{A}^1, \mathcal{A}^2, \Omega^1, \Omega^2, T, O, r, \gamma, H)$ where $\mathcal{S}$ is the state space, $\mathcal{A} \equiv \mathcal{A}^1 \times \mathcal{A}^2$ and $\Omega \equiv \Omega^1 \times \Omega^2$ are the joint-action the joint-observation space of player 1 and player 2. The transition probability from state s to s' after taking a joint action (a^1, a^2) is given by $T(s'|s, a^1, a^2)$. $O(o^1, o^2|s)$ is the conditional probability of observing a joint observation (o^1, o^2) under state s. All players share a common reward function $r(s, a^1, a^2)$, γ is the reward discount factor and H is the horizon length.

Player 1 and 2, with potentially different observation and action spaces, are controlled by policy π^1 and π^2. At each timestep t, the players observe $o_t = (o_t^1, o_t^2) \sim O(o_t^1, o_t^2|s_t)$ under state $s_t \in \mathcal{S}$ and produces a joint action $a_t = (a_t^1, a_t^2) \in \mathcal{A}$ sampled from the joint policy $\pi(a_t|\tau_t) = \pi^1(a_t^1|\tau_t^1)\pi^2(a_t^2|\tau_t^2)$ where τ_t^1 and τ_t^2 contain a trajectory history until timestep t from the perspective of each agent. All players receive a shared reward $r_t = r(s_t, a_t^1, a_t^2)$. The return of a joint trajectory $\tau = (o_0, a_0, r_0, ..., r_{H-1}, o_H) \in \mathcal{T} \equiv (\Omega \times \mathcal{A} \times \mathbb{R})^H$ can be written as $G(\tau) = \sum_{t=0}^{H} \gamma^t r_t$. The expected return of a joint policy (π^1, π^2) is $J(\pi^1, \pi^2) = \mathbb{E}_{\tau \sim \rho(\pi^1, \pi^2)} G(\tau) = \int_\tau P(\tau|\pi^1, \pi^2) G(\tau)$ where $\rho(\pi^1, \pi^2)$ is the distribution over trajectories of the joint policy (π^1, π^2) and $P(\tau|\pi^1, \pi^2)$ is the probability of τ being sampled from a joint policy (π^1, π^2). Since we will be working with multiple joint policies, we use subscript to denote different joint policies e.g., $\pi_A = (\pi_A^1, \pi_A^2)$ is a different joint policy from $\pi_B = (\pi_B^1, \pi_B^2)$. Furthermore, we will use $i, j \in \{1, 2\}$ where $i \neq j$ as superscripts to refer to different player roles. Finally, policies (π_A^1, π_A^2) from a joint policy π_A aim to maximize $J(\pi_A^1, \pi_A^2)$.

Since we are interested in creating dissimilar policies for any given Dec-POMDP, it is useful to have an environment-agnostic measure that captures the similarity of policies. First, we consider a measure that can compute the similarity between policies of the same role i, e.g., π_A^i and π_B^i. We can measure this with the probability of a joint trajectory τ produced by either π_A^i or π_B^i. However, in the two-player setting, we need to pair these policies with a reference policy π_{ref}^j. Specifically, π_A^i and π_B^i are considered similar if they are likely to produce the same trajectories when paired with a reference policy π_{ref}^j. We define similar policies as follows:

Definition 2.1 (Similar policies)**.** Considering two policies of the same role i, π_A^i and π_B^i, and a reference policy π_{ref}^j of a different role j, π_A^i is similar to π_B^i up to ϵ if and only if $|1 - \frac{P(\tau|\pi_A^i, \pi_{\text{ref}}^j)}{P(\tau|\pi_B^i, \pi_{\text{ref}}^j)}| \leq \epsilon; \forall \tau \in \mathcal{T}$, where $0 \leq \epsilon \leq 1$.

Next, we consider an alternate view on measuring similarity between policies using policy compatibility (Section 3). Policy compatibility measures the change in performance of a joint policy π_B before and after one of its policies π_B^i is substituted by another policy π_A^i. We define compatibility between a policy π_A^i and a joint policy π_B as follows:

Definition 2.2 (Compatible policies)**.** Given a policy π_A^i and a joint policy π_B, we define the compatibility ratio between π_A^i and π_B as: $C(\pi_A^i, \pi_B) := \frac{J(\pi_A^i, \pi_B^j)}{J(\pi_B^1, \pi_B^2)}$. We say that π_A^i is compatible with π_B if and only if $C(\pi_A^i, \pi_B) \geq 1 - \epsilon$.

3. Learning Incompatible Policies (LIPO)

The goal of LIPO is to learn several joint policies that are dissimilar to each other and, therefore, create a pool of diverse partner agents. We theoretically show that policy compatibility can be used to identify whether two policies are different. Based on this observation, we propose a novel training objective that produces behaviorally diverse policies. Finally, we incorporate a mutual information objective that encourages each policy to learn local variations.

3.1. Identifying Distinct Policies with Policy Compatibility

In this section, we motivate our objective by looking at two joint policies: $\pi_A = (\pi_A^1, \pi_A^2)$ and $\pi_B = (\pi_B^1, \pi_B^2)$. The goal is for π_A to learn a different solution from π_B via the compatibility criterion. Importantly, the compatibility criterion can be computed without direct access to the trajectory distribution, which can be difficult to estimate.

Under the following assumptions, we can simplify the setting such that a simple relationship between similarity measure and compatibility criterion emerges.

Assumption 3.1. A common ϵ is used for Def. 2.1 and 2.2

Assumption 3.2. $P(\tau|\pi_B^i, \pi_B^j) > 0; \forall \tau \in \mathcal{T}$

Assumption 3.3. $G(\tau) > 0; \forall \tau \in \mathcal{T}$

By reasoning about the expected return under different pairs of policies, we derive our main result. (The proof can be found in Appendix A.)

Theorem 3.4. *If π_A^i is similar to π_B^i with π_B^j as a a reference, then π_A^i is compatible with π_B.*

Corollary 3.5 (Contrapositive of Theorem 3.4)**.** *If π_A^i is not compatible with π_B , then π_A^i is not similar to π_B^i.*

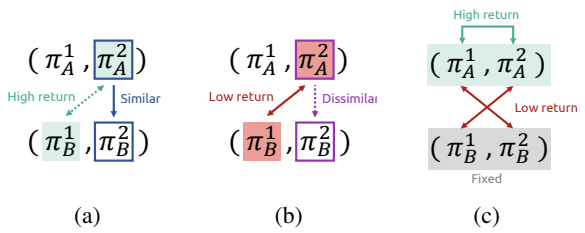


Figure 1: Conceptual illustration of Theorem 3.4 (a) and Corollary 3.5 (b). Solid lines represent given relationships and dotted lines represent implied relationships. The objective of π_A (Eq. 1) in relation to π_B is shown by (c).

The result from Corollary 3.5 shows that we can find a policy π_A^i that is not similar to π_B^i by decreasing its compatibility ratio with π_B until they are incompatible, i.e., $C(\pi_A^i, \pi_B) < 1 - \epsilon$. As a result, we can create π_A that is not similar to π_B by minimizing $C(\pi_A^1, \pi_B)$ and $C(\pi_A^2, \pi_B)$. Additionally, we can ensure that π_A learns a meaningful solution by maximizing $J(\pi_A^1, \pi_A^2)$. Assuming that π_B has learned a solution and is fixed, the optimization objective of π_A can be written as

$$\max_{\pi_A^1, \pi_A^2} J(\pi_A^1, \pi_A^2) - \lambda_{\text{XP}}(C(\pi_A^1, \pi_B) + C(\pi_A^2, \pi_B)),$$

where $\lambda_{\text{XP}} > 0$ is a hyper-parameter controlling importance of the compatibility term. Since the denominators of the compatibility ratios do not depend on π_A, therefore, the objective becomes

$$\max_{\pi_A^1, \pi_A^2} J(\pi_A^1, \pi_A^2) - \lambda_{\text{XP}}(J(\pi_A^1, \pi_B^2) + J(\pi_B^1, \pi_A^2)) \quad (1)$$

In practice, we use the parameter sharing technique for better sample efficiency and faster convergence (Tan, 1993; Foerster et al., 2018; Rashid et al., 2018). Assuming that a policy π_A^i is a neural network parameterized by θ_A^i, this means that for a joint policy (π_A^1, π_A^2), we have $\theta_A^1 = \theta_A^2$. Still, π_A^1 and π_A^2 can behave differently as they observe different parts of the environment and have a different player indicator concatenated with their local observations. We denote the expected joint return of self-play (SP) trajectories – where a policy would interact with a clone of itself – as $J_{\text{SP}}(\pi_A) := J(\pi_A^1, \pi_A^2)$ and expected joint return of cross-play (XP) trajectories – where policy for each agent will be chosen from a different joint policy – as $J_{\text{XP}}(\pi_A, \pi_B) := J(\pi_A^1, \pi_B^2) + J(\pi_B^1, \pi_A^2)$. By incorporating the parameter sharing technique, we can rewrite Eq. (1) as

$$\max_{\pi_A} J_{\text{SP}}(\pi_A) - \lambda_{\text{XP}} J_{\text{XP}}(\pi_A, \pi_B) \quad (2)$$

3.2. Learning a Population of Diverse Policies

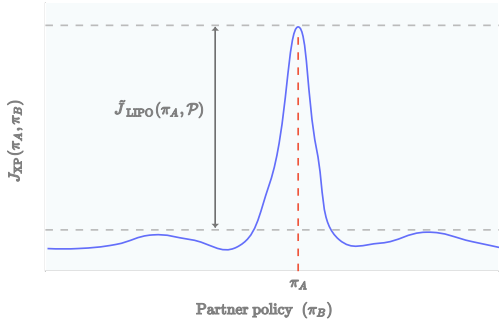


Figure 2: Hypothetical expected return of π_A trained with J_{LIPO} when paired with different partner policies. The red dashed line shows the expected return when the partner policy is π_A, i.e., $\pi_B = \pi_A$.

We propose to use a population of agents $\mathcal{P} = \{\pi_A | 1 \leq A \leq N\}$ where each policy is represented by a neural network with a set of parameters θ_A and N is the population size. Ultimately, we want each member of the population to have a different behavior relative to *the rest of the population*. We can write such an objective by expanding the cross-play term in Eq. (2) to cover the entire population. For a policy π_A in a population $\mathcal{P}$, its objective becomes

$$\max_{\pi_A} J_{\text{LIPO}}(\pi_A, \mathcal{P}) = J_{\text{SP}}(\pi_A) - \lambda_{\text{XP}} \tilde{J}_{\text{XP}}(\pi_A, \mathcal{P}), \quad (3)$$

$$\text{where} \quad \tilde{J}_{\text{XP}}(\pi_A, \mathcal{P}) = \max_{\pi_B \in \mathcal{P}_{-A}} J_{\text{XP}}(\pi_A, \pi_B), \quad (4)$$

$$\mathcal{P}_{-A} = \mathcal{P} \backslash \{\pi_A\}$$

Intuitively, π_A should behave differently from every policy in the population. So, we use the max operation as an aggregation function in Eq. 4. We refer to J_{LIPO} as the *compatibility gap* between a policy π_A and a population $\mathcal{P}$. Fig. 2 shows graphical interpretation of this objective.

We can see that the compatibility gap objective only uses the expected return (J_{SP} and J_{XP}), and therefore it is insensitive to the state and action information from trajectories. We argue that this distinction of LIPO helps the agents discover diverse solutions in situations in which previous methods might be ineffective (Sec. 4.1). Fig. 3 shows the difference between the training objective of our method and other works.

3.3. Inducing Variations in Each Policy

In the previous section, we propose to find a unique policy π_A by maximizing the compatibility gap. It is im-

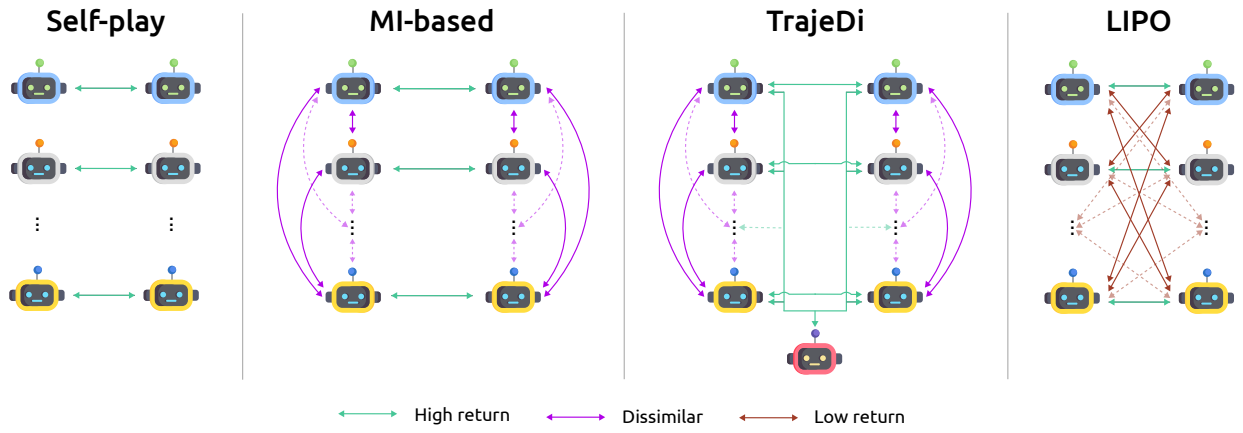


Figure 3: An overview of LIPO compared to previous methods.

portant to note that π_A could also have different local variations to its solution that still satisfy the compatibility gap, $J_{\text{LIPO}}(\pi_A, P)$. Similar to (Kumar et al., 2020; Osa et al., 2022; Lucas & Allen, 2022), we propose to use a mutual information objective to induce such variations. Specifically, we condition the policy on a latent variable z such that a policy π_A has the form of $\pi_A(a|\tau) = \mathbb{E}_{z^1\sim p(z^1), z^2\sim p(z^2)}\pi_A^1(a^1|\tau^1, z^1)\pi_A^2(a^2|\tau^2, z^2)$ where $p(z^1, z^2)$ is a pre-defined prior distribution. We can promote solution variety by maximizing $I(\{o^1, a^1\}; z^1)$ and $I(\{o^2, a^2\}; z^2)$, where $I(\cdot;\cdot)$ is the mutual information between two random variables. Intuitively, this objective encourages each policy to observe different observation and perform different actions given different values of the latent variable. However, maximizing $I(\{o^i, a^i\}; z^i)$ directly is intractable, instead we optimize the lower bound of the mutual information (Kingma & Welling, 2013; Mahajan et al., 2019; Osa et al., 2022):

$$\begin{aligned} I(\{o^i, a^i\}; z^i) &= H(z^i) - H(z^i|\{o^i, a^i\}) \\ &= H(z^i) + \mathbb{E}_{z^i,(o^i,a^i)}[\log p(z^i|o^i, a^i)] \\ &\geq H(z^i) + \mathbb{E}_{z^i,(o^i,a^i)}[\log q_{\phi_A}(z^i|o^i, a^i)] \end{aligned}$$

where $q_{\phi_A}(z^i|o^i, a^i)$ is an approximation of the true posterior $p(z^i|o^i, a^i)$ parameterized by ϕ_A. Thus, maximizing $I(\{o^1, a^1\}; z^1)$ and $I(\{o^2, a^2\}; z^2)$ is an optimization problem that can be written as

$$\max_{\pi_A, \phi_A} \frac{1}{2}\sum_{i=1}^{2} H(z^i) + \mathbb{E}_{z^i,(o^i,a^i)} \log q_{\phi_A}(z^i|o^i, a^i) \quad (5)$$

In previous work, shared z (i.e., $z^1 = z^2$) allows all policies to *collectively* switch between different modes of behavior (Mahajan et al., 2019). However, LIPO uses independently sampled z as it utilizes z for a different purpose. Specifically, LIPO maximizes J_{LIPO} to learn diverse solutions and optimizes the MI objective to learn variations of each solution. That is, the MI objective does not impact the diversity *between* different policies but increases variations of a learned solution of each individual policy.

3.4. Implementation

Algorithm 1 Training process of LIPO (on-policy)

This pseudocode is based on self-play. Blue text is related to the MI objective. LIPO specific code is highlighted in green.

Input: A Population $\mathcal{P} = \{\pi_A \mid 1 \leq A \leq N\}$, number of self-play and cross-play episodes per iteration E_{SP} and E_{XP}.
while *not done* **do**
  for $A \in \{1, ..., N\}$ **do**
    $\mathcal{B}^{sp} \leftarrow$ GetEpisodeRollouts($\pi_A, \pi_A, E_{\text{SP}}$)
    Compute $J_{\text{SP}}(\pi_A)$ using $\mathcal{B}^{sp}$
    $\mathcal{B}^{xp} \leftarrow$ GetCrossPlayRollouts($\pi_A, \mathcal{P}, E_{\text{XP}}$)
    Compute $\tilde{J}_{\text{XP}}(\pi_A, \mathcal{P})$ using $\mathcal{B}^{xp}$ (Eq. 4)
    Compute L_{MI} (Eq. 6) using $\mathcal{B}^{sp}$ and $\mathcal{B}^{xp}$
    $\theta_A \leftarrow \theta_A - \nabla_{\theta_A}[-J_{\text{SP}} + \lambda_{\text{XP}}\tilde{J}_{\text{XP}} + \lambda_{\text{MI}}L_{\text{MI}}]$
    $\phi_A \leftarrow \phi_A - \lambda_{\text{MI}}\nabla_{\phi_A}L_{\text{MI}}$
Function `GetEpisodeRollouts`(π_A, π_B, E):
  $\mathcal{B} \leftarrow \{\}$
  for *episode* $\in \{1, .., \frac{E}{2}\}$ **do**
    $z^1, z^2 \sim p(z^1, z^2)$
    $\tau_{\text{AB}} \sim \rho(\pi_A^1(\cdot|\cdot, z^1), \pi_B^2(\cdot|\cdot, z^2))$
    $\tau_{\text{BA}} \sim \rho(\pi_B^1(\cdot|\cdot, z^1), \pi_A^2(\cdot|\cdot, z^2))$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\tau_{\text{AB}}, \tau_{\text{BA}}\}$
  return $\mathcal{B}$
Function `GetCrossPlayRollouts`($\pi_A, \mathcal{P}, E_{XP}$):
  $\mathcal{B}^{xp} \leftarrow \{\}$
  for $\pi_B \in \mathcal{P}_{-A}$ **do**
    $\mathcal{B} \leftarrow$ `GetEpisodeRollouts`($\pi_A, \pi_B, \frac{E_{XP}}{|\mathcal{P}_{-A}|}$)
    $\mathcal{B}^{xp} \leftarrow \mathcal{B}^{xp} \cup \mathcal{B}$
  return $\mathcal{B}^{xp}$

We implement LIPO on top of the multi-agent version of PPO (MAPPO) (Schulman et al., 2017; Yu et al., 2021), which is an on-policy algorithm. For simplicity, we use a feed-forward architecture for all neural networks. Each member $A \in \{1, ..., N\}$ has a joint policy (π_A^1, π_A^2) and a discriminator q_{ϕ_A}. MAPPO is used for both maximizing

J_{SP} and minimizing J_{XP}. We train all joint policies in the population concurrently.

In practice, we modify the MI objective (Eq. 5) such that it is differentiable with respect to the policy π_A^i. Specifically, we modify the variational posterior q_{ϕ_A} such that, instead of a sampled action a^i, it takes the whole distribution $\pi_A^i(\cdot|o^i, z^i)$ as an input, i.e., $q_{\phi_A}(z|o, \pi_A^i(\cdot|o^i, z^i))$. In contrast to previous MI-based approaches (Eysenbach et al., 2018; Sharma et al., 2019; Jiang & Lu, 2021; Lucas & Allen, 2022), we can optimize $I(\{o^i, a^i\}; z^i)$ directly without computing an auxiliary reward (Mahajan et al., 2019; Osa et al., 2022). The loss function of the modified MI objective is

$$L_{\text{MI}}(\pi_A, \phi_A) = -\frac{1}{2}\sum_{i=1}^{2}\mathbb{E}_{z^i,(o^i,a^i)}\log q_{\phi_A}(z^i|o^i, \pi_A^i(\cdot|o^i, z^i))) \quad (6)$$

We set z as a discrete variable and use uniform distribution for $p(z^1)$ and $p(z^2)$. At the beginning of each episode, each policy is given an independently sampled z that will be used until the end of the episode. The overall objective of a policy π_A in a population $\mathcal{P}$ becomes

$$\max_{\pi_A,\phi_A} J_{\text{SP}}(\pi_A) - \lambda_{\text{XP}} J_{\text{XP}}(\pi_A, \mathcal{P}) - \lambda_{\text{MI}} L_{\text{MI}}(\pi_A, \phi_A)$$

In each training iteration, LIPO collects self-play and cross-play trajectories of all policies combinations to compute J_{SP}, $\tilde{J}_{\text{XP}}$ and L_{MI}. Algorithm 1 shows the pseudocode for the training process of LIPO.

4. Experiments

In this section, we study the effectiveness of LIPO under three cooperative environments to answer the following questions: (i) Can LIPO discover diverse solutions? (ii) How does the MI objective affect the behavior of LIPO agents? In this work, we compared LIPO with cooperative MARL methods that do not require domain knowledge. Our baselines include:

Multiple runs of self-play (Multi SP): A simple but effective way to produce multiple agents that could learn different solutions by training multiple SP agents that have different neural network initializations and random seeds (Charakorn et al., 2020; Strouse et al., 2021). Specifically, each run produces a joint policy π_A that maximizes $J_{\text{SP}}(\pi_A)$ using MAPPO.

Self-play with MI (SP_{MI}): A single run of SP agent trained with added MI objective $I(z|o^i, a^i)$. SP_{MI} uses a shared z for both policies and consider each z as a different joint policy. We train SP_{MI} using the same training procedure as LIPO by setting $N = 1$, $\lambda_{\text{XP}} = 0$ and $z^1 = z^2$.

MAVEN (Mahajan et al., 2019): An algorithm designed specifically for learning diverse solutions in cooperative multi-agent environments. A joint policy is represented as $\pi(\cdot|\tau, z)$ and each mode of behavior is represented by the latent variable z. Similar to SP_{MI}, MAVEN uses a shared z for all policies.

Multi SP_{MI} and Multi MAVEN: A population containing joint policies from multiple runs of a corresponding method. Like Multi SP, each run has different neural network initializations and random seeds.

TrajeDi (Lupu et al., 2021): A method that produces a population of diverse agents that also maximize the expected return in cooperative environments. The diversity measure of this method is based on the Jensen-Shannon divergence (JSD) between the trajectory distribution of each policy. More details of all environments and baselines can be found in the appendix.

4.1. Can LIPO discover diverse solutions?

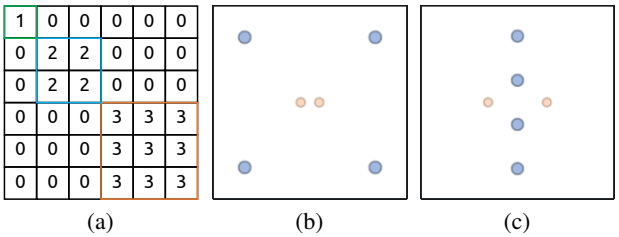
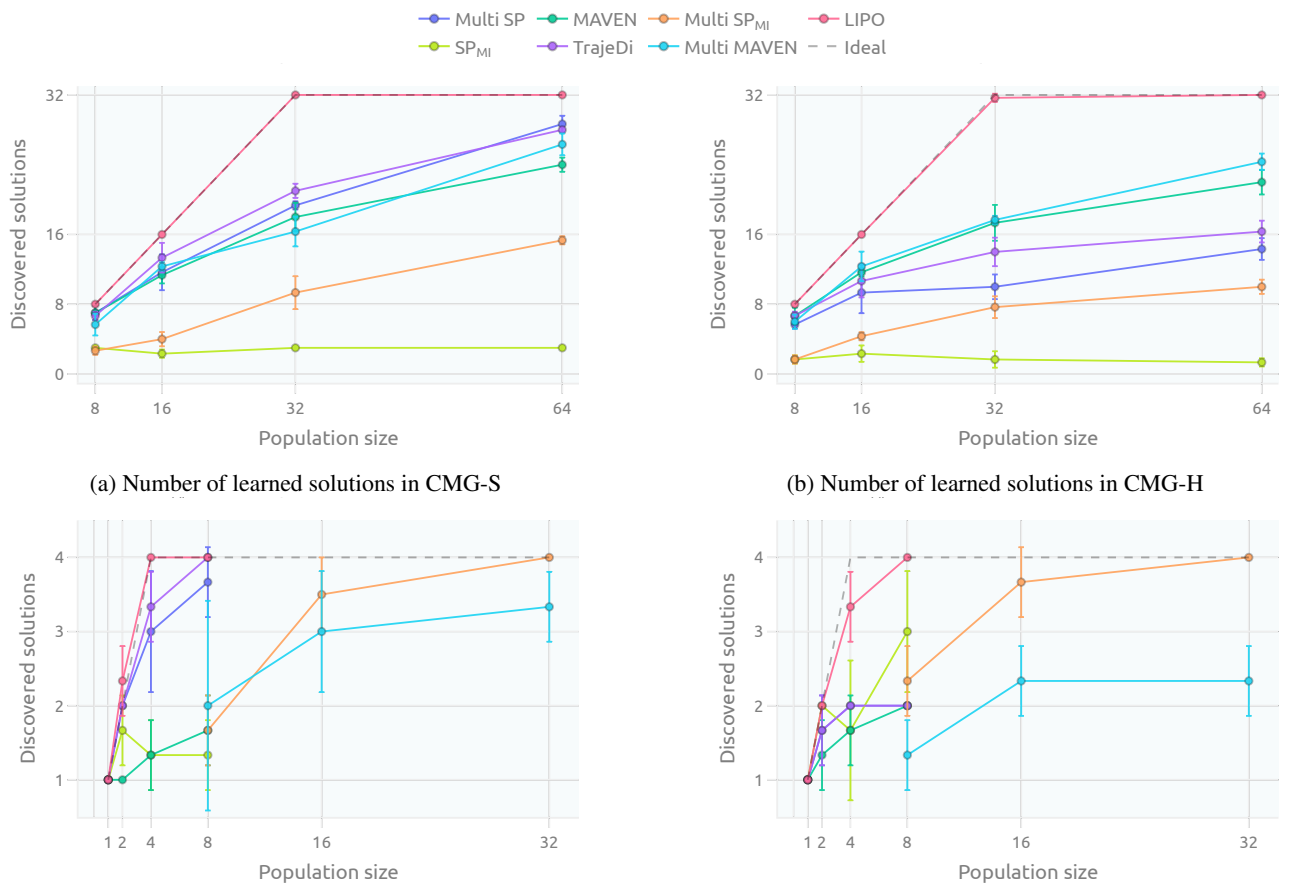


Figure 4: Visualization of the environments: (a) An example payoff matrix of a CMG game with ($M = 3, k_m = m, r_m = m$) (b, c) The agents (orange) and landmark positions (blue) of PMR-C and PMR-L.

We start our investigation in (i) *One-Step Cooperative Matrix Game* (CMG) and (ii) *Point Mass Rendezvous* (PMR)

One-Step Cooperative Matrix Game (CMG): A game of CMG is defined by a tuple $(M, \{k_m\}, \{r_m\})$ where M is the number of solution. For $m \in \{1, ..., M\}$, k_m is the number of compatible actions and r_m is the reward of a solution m. The game is stateless (i.e., there is only one state) and terminate immediately after both players simultaneously choose an action. By choosing the same solution, both player get a reward r_m associated with the chosen solution. We consider two setups of CMG: sub-optimal solutions (CMG-S) and hard-to-find solutions (CMG-H). We set ($M = 32$, $k_m = 8$, $r_m = 0.5 * (1 + \frac{m-1}{M-1})$) and ($M = 32$, $k_m = m$, $r_m = 1$) for CMG-S and CMG-H respectively. An example payoff matrix is shown in Fig. 4a.

Point Mass Rendezvous (PMR): The environment is based on the Multi-Agent Particle Environment (Lowe et al., 2017; Terry et al., 2020). The goal of this environment is for the



(a) Number of learned solutions in CMG-S

(b) Number of learned solutions in CMG-H

(c) Number of learned solutions in PMR-C

(d) Number of learned solutions in PMR-L

Figure 5: Number of discovered solutions in different population sizes. The error bars represent the standard errors of three random seeds.

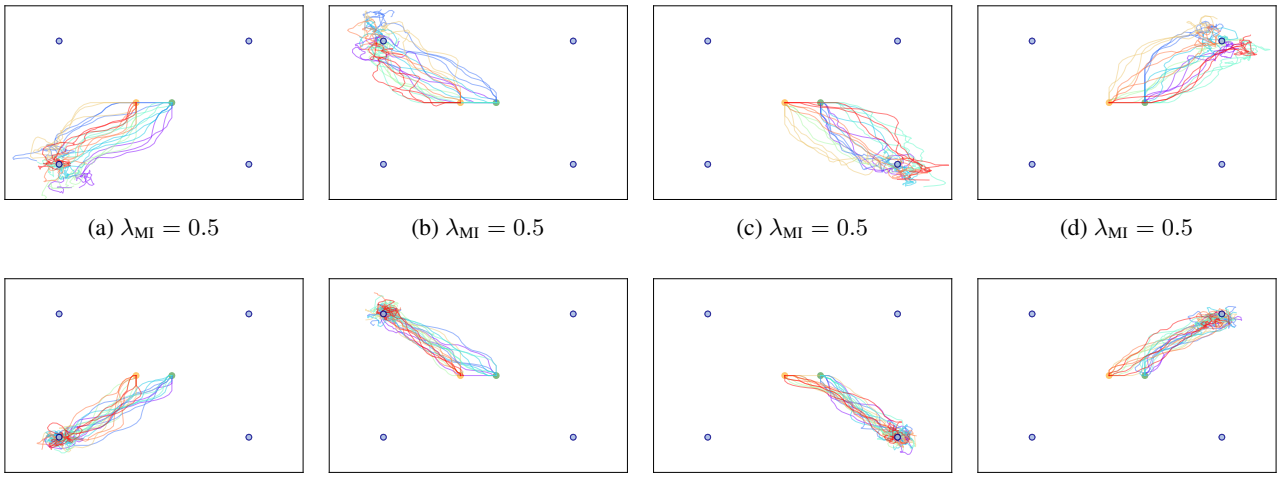
two agents to navigate to a landmark together. Doing so will reward both agents as long as they stay on top of a landmark. We consider each landmark as a solution in this environment. There are two modes in this environment: `Circle` (PMR-C) and `Line` (PMR-L). In `Circle`, landmarks are distributed evenly on a circumference of a circle. Thus, all landmarks are equally optimal. In `Line`, landmarks are placed in a vertical line. In this scenario, closer landmarks are easier to be found, while further landmarks are harder to be discovered.

The population size of MI-based methods (SP_{MI} and MAVEN) is equal to the number of dimensions of the latent variable $|z|$. In methods that require multiple runs (Multi SP_{MI} and Multi MAVEN), the population size is $n \times |z|$ where n is the number of runs. For Multi SP_{MI} and Multi MAVEN, we use $|z| = 8$ in all environments. For population-based methods (Multi SP, TrajeDi, and LIPO), the population size is the number of joint policies in the population.

Results: Fig 5 shows the number of learned solutions as the population size increases. Ideally, if the population size increases by one, one more solution should be discovered, as depicted by the dashed line in the figure. In all environments, LIPO can discover more solutions than the baselines given the same population size. LIPO is also better at discovering sub-optimal policies as illustrated in CMG-S and PMR-L (Fig. 5a,5d). Also, we have experimented with large λ_{MI} for the baselines. This helps the baselines discover more solutions. However, if λ_{MI} is too large, we find that the baselines fail to produce capable policies. Hyperparameters of all methods can be found in the appendix.

4.2. How does the MI objective effect the behavior of LIPO agents?

Here, we investigate further into how the MI objective affects the behaviors of produced policies. Fig 6 shows the behaviors of policies produced by LIPO with and without the MI objective in PMR-C. We can see the effect of the



(a) $\lambda_{\text{MI}} = 0.5$ (b) $\lambda_{\text{MI}} = 0.5$ (c) $\lambda_{\text{MI}} = 0.5$ (d) $\lambda_{\text{MI}} = 0.5$

(e) $\lambda_{\text{MI}} = 0$ (f) $\lambda_{\text{MI}} = 0$ (g) $\lambda_{\text{MI}} = 0$ (h) $\lambda_{\text{MI}} = 0$

Figure 6: Trajectories of agents trained by LIPO with (a,b,c,d) and without (e,f,g,h) the MI objective in PMR-C. Each row shows four joint policies produced with a single run of LIPO training. Different colors of the trajectories correspond to different values of the latent variable z. The starting positions of both players are shown by the orange and green circles. The blue circles represent the landmarks.

MI objective in the distributions of the trajectories, which exhibit larger variations given a small MI regularization $\lambda_{MI} = 0.5$. With or without the MI regularization, LIPO discovers all the landmarks with $N = 4$.

5. Related Work

Mutual information objectives have been used in reinforcement learning to learn diverse behaviors. In the single-agent domain, (Kumar et al., 2020; Osa et al., 2022) propose to optimize the RL and MI objective simultaneously. The MI objective helps the agent learn diverse solutions for a specific task and increases generalization to unseen environment instances. A similar idea has also been applied to the cooperative multi-agent domain. For instance, MAVEN (Mahajan et al., 2019) optimizes RL and an MI objective to encourage the agents to explore in a committed manner and discover diverse solutions. Also, Any-play (Lucas & Allen, 2022) uses a similar objective to produce training partners with many solutions for an adaptive agent. In contrast, our approach uses the MI objective to regularize each policy to learn local variations of each solution.

Population-based training is another technique that has been proposed to learn a population of diverse solutions. DvD (Parker-Holder et al., 2020) introduces a metric distance between policies using policy-embedding and learns diverse solutions by maximizing the distance between policies while also maximizing the expected return. In the multi-agent cooperative domain, (Canaan et al., 2019; 2020) use a Quality Diversity (QD) algorithm (Mouret & Clune, 2015; Pugh et al., 2016) to produce a population of agents with a different behavioral niche attached to each agent. QD, however, requires domain knowledge to encode different types of behaviors. RPG (Tang et al., 2021) obtain diverse policies by randomizing the factorized reward function, which is not applicable in general. Finally, TrajeDi (Lupu et al., 2021) produces a diverse population of agents, based on the trajectory distribution, that shares a common best response policy. Like TrajeDi, ours does not require domain-specific knowledge. However, LIPO uses state-action information for learning local variations and utilizes expected returns of different pairs of policies to promote behavioral diversity.

The idea of using the empirical return in multi-agent games as a diversity metric has been explored in the context of finding diverse solutions in non-transitive competitive games (Liu et al., 2021; Balduzzi et al., 2019; Perez-Nieves et al., 2021). In particular, Liu et al. share some similar ideas with this work. They propose to use both expected return when encountering different opponents and state-action information to promote diversity in a population of agents in zero-sum games. LIPO can be thought of as a special case designed specifically for cooperative environments.

6. Discussion and Conclusion

Discussion An agent trained with LIPO are incentivized to act adversarially toward agents that behave differently from itself. This behavior might not be desirable for certain downstream tasks. For example, agents produced by LIPO might not be suitable for interacting with humans as they

would refuse to conform with the user, which is the same downside found in Multi SP agents (Carroll et al., 2019; Bard et al., 2020; Hu et al., 2020). However, we believe that training an adaptive agent with these agents, which is the main motivation of this work, would have an opposite effect, in that the adaptive agent would try to comply with what its current partner is doing. In previous work, adaptive agents trained with Multi SP have shown such a behavior (Charakorn et al., 2020; Strouse et al., 2021).

Limitations Although LIPO can be fully parallelized, it requires more computation than the baselines to get an accurate approximation of $\tilde{J}_{\text{XP}}$, which makes it harder to scale up to bigger population size. Instead of collecting all policy pairs, sampling a portion of policy pairs to approximate $\tilde{J}_{\text{XP}}$ could reduce computation cost and training time. Additionally, LIPO requires an additional hyperparameter λ_{XP}. If λ_{XP} is too big, it is possible that a joint policy would focus on minimizing $\tilde{J}_{\text{XP}}$ and struggle to maximize J_{SP}, which will result in an incompetent joint policy. An adaptive mechanism that selects a suitable value for λ_{XP} at different stages of training could help increase training stability.

Future work Evaluating the effectiveness of produced policies in different downstream tasks could give additional metrics for *"goodness"* of the populations. For example, they can be used as training partners for an adaptive agent or training a neural network for agent modeling. These tasks will be included in our future work.

Conclusion We propose LIPO, a method that can create a population of diverse agents in cooperative multi-agent environments. Unlike previous work that uses state-action information from trajectories, LIPO utilizes the concept of compatible policies to create diverse policies. This alternative view of producing diverse agents makes LIPO more robust to the state and action space of the environments. On top of learning diverse solutions, LIPO uses the MI objective to learn local variations of each solution. Empirically, LIPO can produce more solutions than the baselines in various settings.

References

Albrecht, S. V. and Stone, P. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

Balduzzi, D., Garnelo, M., Bachrach, Y., Czarnecki, W., Perolat, J., Jaderberg, M., and Graepel, T. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning*, pp. 434–443. PMLR, 2019.

Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E., et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

Canaan, R., Togelius, J., Nealen, A., and Menzel, S. Diverse agents for ad-hoc cooperation in hanabi. In *2019 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE, 2019.

Canaan, R., Gao, X., Togelius, J., Nealen, A., and Menzel, S. Generating and adapting to diverse ad-hoc cooperation agents in hanabi. *arXiv preprint arXiv:2004.13710*, 2020.

Carroll, M., Shah, R., Ho, M. K., Griffiths, T., Seshia, S., Abbeel, P., and Dragan, A. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

Charakorn, R., Manoonpong, P., and Dilokthanakul, N. Investigating partner diversification methods in cooperative multi-agent deep reinforcement learning. In *International Conference on Neural Information Processing*, pp. 395–402. Springer, 2020.

Charakorn, R., Manoonpong, P., and Dilokthanakul, N. Learning to cooperate with unseen agents through meta-reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1478–1479, 2021.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2018.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Grover, A., Al-Shedivat, M., Gupta, J., Burda, Y., and Edwards, H. Learning policy representations in multiagent systems. In *International conference on machine learning*, pp. 1802–1811. PMLR, 2018.

Hu, H., Lerer, A., Peysakhovich, A., and Foerster, J. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pp. 4399–4410. PMLR, 2020.

Jiang, J. and Lu, Z. The emergence of individuality. In *International Conference on Machine Learning*, pp. 4992–5001. PMLR, 2021.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kumar, S., Kumar, A., Levine, S., and Finn, C. One solution is not all you need: Few-shot extrapolation via structured maxent rl. *Advances in Neural Information Processing Systems*, 33:8198–8210, 2020.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Leibo, J. Z., Dueñez-Guzman, E. A., Vezhnevets, A., Agapiou, J. P., Sunehag, P., Koster, R., Matyas, J., Beattie, C., Mordatch, I., and Graepel, T. Scalable evaluation of multi-agent reinforcement learning with melting pot. In *International Conference on Machine Learning*, pp. 6187–6199. PMLR, 2021.

Liu, X., Jia, H., Wen, Y., Yang, Y., Hu, Y., Chen, Y., Fan, C., and Hu, Z. Towards unifying behavioral and response diversity for open-ended learning in zero-sum games. *Advances in Neural Information Processing Systems*, 34, 2021.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Lucas, K. and Allen, R. E. Any-play: An intrinsic augmentation for zero-shot coordination. *arXiv preprint arXiv:2201.12436*, 2022.

Lupu, A., Cui, B., Hu, H., and Foerster, J. Trajectory diversity for zero-shot coordination. In *International Conference on Machine Learning*, pp. 7204–7213. PMLR, 2021.

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pp. 7611–7622, 2019.

McKee, K. R., Leibo, J. Z., Beattie, C., and Everett, R. Quantifying environment and population diversity in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.08370*, 2021.

Mirsky, R., Carlucho, I., Rahman, A., Fosong, E., Macke, W., Sridharan, M., Stone, P., and Albrecht, S. V. A survey of ad hoc teamwork: Definitions, methods, and open problems. *arXiv preprint arXiv:2202.10450*, 2022.

Mouret, J.-B. and Clune, J. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Osa, T., Tangkaratt, V., and Sugiyama, M. Discovering diverse solutions in deep reinforcement learning by maximizing state-action-based mutual information. *Neural Networks*, 2022.

Papoudakis, G., Christianos, F., and Albrecht, S. V. Agent modelling under partial observability for deep reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=QcwJmp1sTnk.

Parekh, S., Habibian, S., and Losey, D. P. Rili: Robustly influencing latent intent. *arXiv preprint arXiv:2203.12705*, 2022.

Parker-Holder, J., Pacchiano, A., Choromanski, K. M., and Roberts, S. J. Effective diversity in population based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18050–18062, 2020.

Peng, B., Rashid, T., Schroeder de Witt, C., Kamienny, P.-A., Torr, P., Böhmer, W., and Whiteson, S. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34, 2021.

Perez-Nieves, N., Yang, Y., Slumbers, O., Mguni, D. H., Wen, Y., and Wang, J. Modelling behavioural diversity for learning in open-ended games. In *International Conference on Machine Learning*, pp. 8514–8524. PMLR, 2021.

Pugh, J. K., Soros, L. B., and Stanley, K. O. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

Rahman, M. A., Hopner, N., Christianos, F., and Albrecht, S. V. Towards open ad hoc teamwork using graph-based policy learning. In *International Conference on Machine Learning*, pp. 8776–8786. PMLR, 2021.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4295–4304. PMLR, 2018.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Shih, A., Sawhney, A., Kondic, J., Ermon, S., and Sadigh, D. On the critical role of conventions in adaptive human-ai collaboration. In *International Conference on Learning Representations*, 2020.

Stone, P., Kaminka, G. A., Kraus, S., and Rosenschein, J. S. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Strouse, D., McKee, K. R., Botvinick, M., Hughes, E., and Everett, R. Collaborating with humans without human data. *arXiv preprint arXiv:2110.08176*, 2021.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

Tang, Z., Yu, C., Chen, B., Xu, H., Wang, X., Fang, F., Du, S., Wang, Y., and Wu, Y. Discovering diverse multi-agent strategic behavior via reward randomization. *arXiv preprint arXiv:2103.04564*, 2021.

Terry, J. K., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sulivan, R., Santos, L., Perez, R., Horsch, C., Dieffendahl, C., Williams, N. L., Lokesh, Y., Sullivan, R., and Ravi, P. Pettingzoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2020.

Wang, W. Z., Shih, A., Xie, A., and Sadigh, D. Influencing towards stable multi-agent interactions. In *Conference on Robot Learning*, pp. 1132–1143. PMLR, 2022.

Xie, A., Losey, D., Tolsma, R., Finn, C., and Sadigh, D. Learning latent representations to influence multi-agent interaction. In *Conference on Robot Learning*, pp. 575–588. PMLR, 2021.

Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

A. Proof for Theorem 3.4

Theorem. *If π_A^i is similar to π_B^i with π_B^j as a a reference, then π_A^i is compatible with π_B.*

Proof. Let $r(\tau) = \frac{P(\tau|\pi_A^i,\pi_B^j)}{P(\tau|\pi_B^i,\pi_B^j)}$. Because π_A^i is similar to π_B^i, then $1-\epsilon \leq r(\tau) \leq 1+\epsilon$; $\forall\tau$ (Def. 2.1).

From the definition of the expected return of a policy pair in Section 2, we have

$$J(\pi_B^i,\pi_B^j) = \mathbb{E}_{\tau\sim\rho(\pi_B^i,\pi_B^j)}G(\tau) = \int_\tau P(\tau|\pi_B^i,\pi_B^j)G(\tau)$$

Assume that Def. 2.1 and 2.2 share a common ϵ, and $P(\tau|\pi_B^i,\pi_B^j) > 0$ and $G(\tau) > 0$ for all $\tau \in \mathcal{T}$. Then, we can use importance sampling to write $J(\pi_A^i,\pi_B^j)$ in relation to $J(\pi_B^i,\pi_B^j)$:

$$\begin{aligned}
J(\pi_A^i,\pi_B^j) &= \mathbb{E}_{\tau\sim\rho(\pi_A^i,\pi_B^j)}G(\tau) \\
&= \int_\tau P(\tau|\pi_A^i,\pi_B^j)G(\tau) \\
&= \int_\tau \frac{P(\tau|\pi_B^i,\pi_B^j)}{P(\tau|\pi_B^i,\pi_B^j)}P(\tau|\pi_A^i,\pi_B^j)G(\tau) \\
&= \int_\tau r(\tau)P(\tau|\pi_B^i,\pi_B^j)G(\tau) \\
&= \mathbb{E}_{\tau\sim\rho(\pi_B^i,\pi_B^j)}r(\tau)G(\tau)
\end{aligned}$$

Because $1-\epsilon \leq r(\tau) \leq 1+\epsilon$, loose upper and lower bounds of $J(\pi_A^i,\pi_B^j)$ can be written as:

$$\begin{aligned}
(1-\epsilon)J(\pi_B^i,\pi_B^j) \leq & J(\pi_A^i,\pi_B^j) \leq (1+\epsilon)J(\pi_B^i,\pi_B^j) \\
1-\epsilon \leq & \frac{J(\pi_A^i,\pi_B^j)}{J(\pi_B^i,\pi_B^j)} \leq 1+\epsilon
\end{aligned}$$

This means that π_A^i is compatible with π_B (Def. 2.2).

$\therefore$ If π_A^i is similar to π_B^i, then π_A^i is compatible with π_B. □

B. Additional Environments Details

B.1. Point Mass Rendezvous (PMR)

PMR is based on the the Multi-Agent Particle Environment (Lowe et al., 2017; Terry et al., 2020). The observation of each agent is relative positive of the landmarks and the other agent to itself. In PMR-C, the start positions of the agents are {(0.3,0), (-0.3,0)} and the landmarks positions are {(1.59, 1.59), (1.59, -1.59), (-1.59, 1.59), (-1.59, -1.59)}. For PMR-L, the start and the landmark positions are {(1,0),(0,1)} and {(0,2.25),(0,0.75),(0,-0.75),(0,-2.25)}. An episode will be terminated after 50 timesteps. The agents are incentivized to go to the same landmark and stay close together with the reward function

$$r_t = 1 - d(p^i, c) - \min_{l\in L} d(l, c),$$

where $d(\cdot,\cdot)$ is the euclidean distance between two points, p^i is the 2-d coordinate of agent i, c is the average coordinate of all agents, and L is the set of all landmarks.

C. Implementation Details

Table 1: Hyperparameters used by the MAPPO algorithm.

Hyperparameters	Value
learning rate	0.003
Discount factor (γ)	0.99
GAE lambda	0.95
Batch size	100 (CMG), 2500 (PMR)
epochs	10
number of mini-batches	2
Entropy coefficient	0.0 (CMG), 0.03 (PMR)
Value loss coefficient	0.5
PPO clipping parameter	0.3
Gradient clipping	0.5
Adam epsilon	1e-5
Total timesteps (T)	30000 (CMG), 750000 (PMR)

All methods are implemented on top of MAPPO except MAVEN. The critic, policy, and discriminator q_ϕ are 3-layer feed-forward neural networks where each hidden layer has 64 units. For a fair comparison, we use the same or more environment steps in the policy update of the baselines compared to LIPO. Common hyperparameters of methods based on MAPPO are shown in Table. 1. For a fair comparison, the baselines collect additional data to have the same or more timesteps than LIPO.

C.1. MAPPO

MAPPO is the base MARL algorithm for all baselines except MAVEN. The policy parameters are shared among all policies. The critic takes a state of the environment and outputs an expected return of a given global state. The global state is provided by the environment and only used during training. For the full training objectives of MAPPO, we refer the reader to Appendix A of Yu et al. (2021).

C.2. Multi SP

A population of Multi SP is obtained by simply running multiple runs of the MAPPO algorithm. Each run collects a total of T timesteps.

C.3. SP_{MI}

SP_{MI} is based on the MAPPO algorithm. In addition to policy and critic networks, a discriminator network q_ϕ is implemented using the same feed-forward architecture. It takes a local observation o^i and action distribution $\pi^i(\cdot|o^i)$ as inputs and outputs the discrete probability of the latent variable. The latent variable of all policies is shared during an episode. It collects a total of $2|z|T$ timesteps and produces $|z|$ joint policies.

C.4. MAVEN

We use the same network architecture presented in (Mahajan et al., 2019) with recurrent neural networks. However, we do not use the hierarchical policy but sample z from the uniform distribution. The latent variable of all policies is shared during an episode. Similar to SP_{MI}, it collects a total of $2|z|T$ timesteps and produces $|z|$ joint policies.

C.5. Multi SP_{MI} and Multi MAVEN

A population is produced by multiple runs of each base algorithm. The final population size is $n|z|$, where n is the number of runs. Notably, this baseline uses the training data differently from the base algorithms. Instead of training a long single run, this approach allows the policy to "restart" by using different initialization of neural networks. For example, let $N = n|z|$, training a single run with $|z| = N, n = 1$ might not discover as many solutions as training n runs with $|z| = \frac{N}{n}$. Empirically, we find that multiple runs of a base algorithm can find more solutions compared to a longer run of the algorithm.

C.6. TrajeDi

TrajeDi is implemented on top of the MAPPO algorithm. Different from the original implementation, we remove the best-response (BR) policy from the population. Since the BR policy might work well with only a specific solution, removing BR potentially increase the number of variations in the population. Our modified loss is:

$$\mathcal{L} = -[\sum_{A=1}^{N}(J_{\text{SP}}(\pi_A)) + \alpha \text{JSD}_\gamma(\pi_1, ..., \pi_N)],$$

where JSD is the proposed diversity objective of TrajeDi. α and γ are the hyperparameters of TrajeDi.

C.7. LIPO

LIPO use the same implementation as SP_{MI} except LIPO uses independent latent variable for each policy and $\lambda_{\text{XP}} > 0$. Additionally, LIPO uses extra critics for the cross-play trajectories. In total, LIPO has a self-play critic $V_{sp}^{\pi_A}$ and $N-1$ cross-play critics $\{V_{xp}^{\pi_A,\pi_B} \mid \pi_B \in \mathcal{P}_{-A}\}$. It uses $2NT$ timesteps for policy updates and produces N policy. Specifically, each joint policy uses T timesteps from self-play trajectories and another T timesteps from $\max(\tilde{J}_{\text{XP}})$. We set λ_{MI} as 0.0 and 0.5 in CMG and PMR respectively.

D. Hyperparameters

We provide the possible search values of each method in Tables 2, 3, 4, 5, 6 and 7. The hyperparameters are searched individually for each population size. We use three random seeds for each set of hyperparameters. We do not use any validation method; instead, we present the results using the best parameters in the main paper.

E. Additional Results

E.1. Sensitivity of λ_{XP}

We provide additional results of LIPO with varying values of λ_{XP} in Fig. 7.

Table 2: The values of λ_{MI} used by SP_{MI} in all environments. The best values are shown in bold.

Population size	Hyperparameters	Values (CMG-S)	Values (CMG-H)	Values (PMR-C)	Values (PMR-L)
1		-	-	[0.5,1,5,**10**]	[0.5,**1**,5,10]
2		-	-	[0.5,1,5,**10**]	[0.5,1,**5**,10]
4		-	-	[0.5,**1**,5,10]	[0.5,1,**5**,10]
8	λ_{MI}	[1,5,**10**,50]	[1,5,10,**50**]	[0.5,**1**,5,10]	[0.5,**1**,5,10]
16		[1,5,10,**50**]	[1,5,10,**50**]	-	-
32		[1,5,10,**50**]	[1,5,10,**50**]	-	-
64		[1,5,10,**50**]	[1,5,10,**50**]	-	-

Table 3: The values of λ_{MI} used by MAVEN in all environments. The best values are shown in bold.

Population size	Hyperparameters	Values (CMG-S)	Values (CMG-H)	Values (PMR-C)	Values (PMR-L)
1		-	-	[1,5,**10**,50]	[**1**,5,10,50]
2		-	-	[1,5,**10**,50]	[1,5,**10**,50]
4		-	-	[**1**,5,10,50]	[1,**5**,10,50]
8	λ_{MI}	[**1**,5,10,50]	[1,**5**,10,50]	[**1**,5,10,50]	[**1**,5,10,50]
16		[1,**5**,10,50]	[1,5,**10**,50]	-	-
32		[1,**5**,10,50]	[1,5,10,**50**]	-	-
64		[1,5,**10**,50]	[1,5,10,**50**]	-	-

Table 4: The values of λ_{MI} used by Multi SP_{MI} in all environments. The best values are shown in bold.

Population size	Hyperparameters	Values (CMG-S)	Values (CMG-H)	Values (PMR-C)	Values (PMR-L)
1		-	-	-	-
2		-	-	-	-
4		-		-	-
8	λ_{MI}	[1,5,10,**50**]	[1,5,**10**,50]	[**1**,5,10,50]	[**1**,5,10,50]
16		[1,5,10,**50**]	[1,5,10,**50**]	[1,**5**,10,50]	[1,**5**,10,50]
32		[1,5,10,**50**]	[1,5,10,**50**]	[1,**5**,10,50]	[**1**,5,10,50]
64		[1,5,10,**50**]	[1,5,10,**50**]	-	-

Table 5: The values of λ_{MI} used by Multi MAVEN in all environments. The best values are shown in bold.

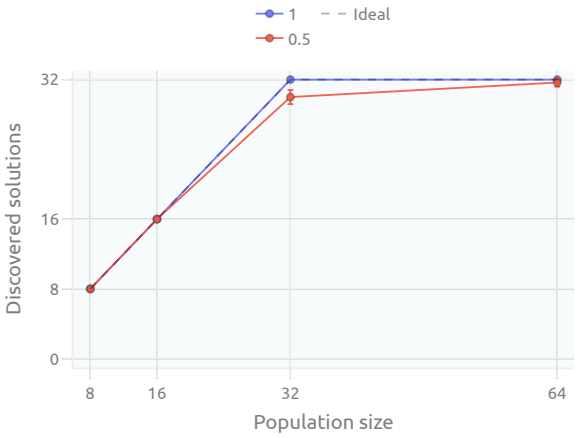
Population size	Hyperparameters	Values (CMG-S)	Values (CMG-H)	Values (PMR-C)	Values (PMR-L)
1		-	-	-	-
2		-	-	-	-
4		-		-	-
8	λ_{MI}	[1,**5**,10,50]	[**1**,5,10,50]	[1,5,**10**,50]	[1,5,**10**,50]
16		[**1**,5,10,50]	[1,**5**,10,50]	[1,5,**10**,50]	[1,5,**10**,50]
32		[**1**,5,10,50]	[**1**,5,10,50]	[1,5,**10**,50]	[1,**5**,10,50]
64		[**1**,5,10,50]	[1,**5**,10,50]	-	-

Table 6: The values of α and γ used by TrajDi in all environments. The best values are shown in bold.

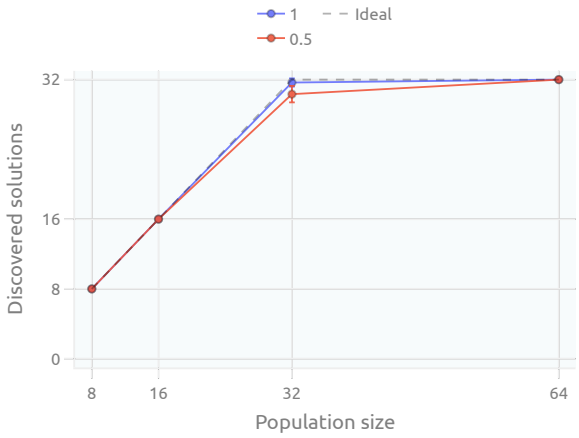
Population size	Hyperparameters	Values (CMG-S)	Values (CMG-H)	Values (PMR-C)	Values (PMR-L)
1		-	-	[1,5,**10**,50]	[1,5,10,**50**]
2		-	-	[1,5,10,**50**]	[1,5,10,**50**]
4		-		[**1**,5,10,50]	[1,**5**,10,50]
8	α	[**0.01**,0.05,0.1,0.2]	[0.01,0.05,**0.1**,0.2]	[1,5,**10**,50]	[1,**5**,10,50]
16		[0.01,**0.05**,0.1,0.2]	[0.01,0.05,**0.1**,0.2]	-	-
32		[**0.01**,0.05,0.1,0.2]	[0.01,0.05,0.1,**0.2**]	-	-
64		[**0.01**,0.05,0.1,0.2]	[0.01,0.05,**0.1**,0.2]	-	-
1		-	-	[0, **0.1**, 0.5]	[0, **0.1**, 0.5]
2		-	-	[0, **0.1**, 0.5]	[**0**, 0.1, 0.5]
4		-	-	[**0**, 0.1, 0.5]	[0, **0.1**, 0.5]
8	γ	0	0	[0, 0.1, **0.5**]	[0, **0.1**, 0.5]
16		0	0	-	-
32		0	0	-	-
64		0	0	-	-

Table 7: The values of λ_{XP} used by LIPO in all environments. The best values are shown in bold.

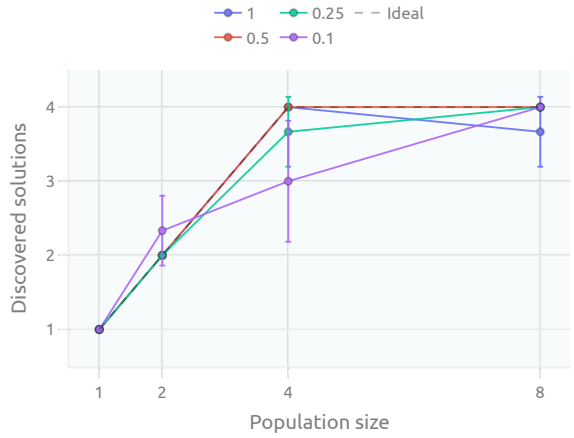
Population size	Hyperparameters	Values (CMG-S)	Values (CMG-H)	Values (PMR-C)	Values (PMR-L)
1		-	-	[0.1,0.25,**0.5**,1]	[0.1,0.25,**0.5**,1]
2		-	-	[**0.1**,0.25,0.5,1]	[0.1,**0.25**,0.5,1]
4		-	-	[0.1,0.25,**0.5**,1]	[0.1,**0.25**,0.5,1]
8	λ_{XP}	[**0.5**,1]	[0.5,**1**]	[0.1,**0.25**,0.5,1]	[0.1,**0.25**,0.5,1]
16		[0.5,**1**]	[0.5,**1**]	-	-
32		[0.5,**1**]	[0.5,**1**]	-	-
64		[0.5,**1**]	[0.5,**1**]	-	-



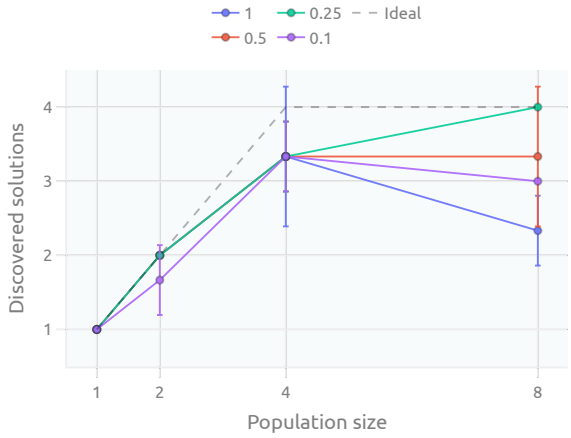
(a) Number of learned solutions in CMG-S



(b) Number of learned solutions in CMG-H



(c) Number of learned solutions in PMR-C



(d) Number of learned solutions in PMR-L

Figure 7: Number of discovered solutions in different population of LIPO with different values of λ_{XP}.