RACE: Improve Multi-Agent Reinforcement Learning with Representation Asymmetry and Collaborative Evolution

Pengyi Li [1] **Jianye Hao** [1] **Hongyao Tang** [1] **Yan Zheng** [1] **Xian Fu** [1]

Abstract

Multi-Agent Reinforcement Learning (MARL) has demonstrated its effectiveness in learning collaboration, but it often struggles with low-quality reward signals and high non-stationarity. In contrast, Evolutionary Algorithm (EA) has shown better convergence, robustness, and signal quality insensitivity. This paper introduces a hybrid framework, Representation Asymmetry and Collaboration Evolution (RACE), which combines EA and MARL for efficient collaboration. RACE maintains a MARL team and a population of EA teams. To enable efficient knowledge sharing and policy exploration, RACE decomposes the policies of different teams controlling the same agent into a *shared* nonlinear observation representation encoder and *individual* linear policy representations. To address the partial observation issue, we introduce Value-Aware Mutual Information Maximization to enhance the shared representation with useful information about superior global states. To facilitate coordination, EA evolves the population using novel *agent-level* crossover and mutation operators, offering diverse experiences for MARL. Concurrently, MARL optimizes its policies and injects them into the population for evolution. The experiments on challenging continuous and discrete tasks demonstrate that RACE significantly improves the basic algorithms, consistently outperforming other algorithms. Our code is available at https://github.com/yeshenpy/RACE.

1. Introduction

Multi-Agent Reinforcement Learning (MARL) has emerged as a powerful approach for tackling complex real-world problems across various domains, such as robot control (Johannink et al., 2019; Yuan et al., 2023), game AI (Vinyals et al., 2019; HAO et al., 2023b), and transportation (Li et al., 2019; Ni et al., 2021). In MARL, individual agents interact with the environment and with each other, collecting samples and receiving reward signals to evaluate their decisions. By leveraging value function approximation, MARL optimizes policies through gradient updates. However, the reward signals are often of low quality (e.g., deceptive, sparse, delayed, and team-level), making it challenging to obtain accurate value estimates (Yang et al., 2020a;b). Consequently, the gradient-based optimization approach may struggle to efficiently explore the multi-agent policy space and facilitate collaboration (Christianos et al., 2020; Li et al., 2022). Furthermore, MARL algorithms suffer from the problem of non-stationarity (Papoudakis et al., 2019), as the agents learn concurrently and continuously influence each other, breaking the Markov assumption on which most single-agent RL algorithms are based (Majid et al., 2021). To address this, the Centralized Training with Decentralized Execution (CTDE) (Lowe et al., 2017) paradigm has been proposed. During centralized training, agents can access to other agents' information and the global state, while during decentralized execution, agents execute decisions independently based on their individual policies. However, the non-stationarity problem persists, especially when agents only have partial observations of their environment, making policy optimization even more challenging. These issues pose significant hurdles to MARL in effectively learning collaboration (Majid et al., 2021; Papoudakis et al., 2019).

Evolutionary Algorithm (EA) (Bäck & Schwefel, 1993; Gomez et al., 2006) simulates the natural process of genetic evolution and does not rely on gradient information for policy optimization, which has been demonstrated to be competitive with RL (Salimans et al., 2017) in single-agent settings. Unlike RL which typically maintains only one policy, EA maintains a population of individuals and performs iterative evolution according to policy fitness. EA possesses several key strengths: 1) EA does not require RL value function approximation and directly evolves individuals within the population according to fitness, i.e., the cumulative rewards. This makes EA more robust to reward signals (Sigaud, 2022; Khadka & Tumer, 2018). 2) EA is not reliant on the Markov property in problem formulation

[1]College of Intelligence and Computing, Tianjin University, China. Correspondence to: Jianye Hao <jianye.hao@tju.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

and evolves policies from the team perspective, thereby circumventing the non-stationarity problem encountered in MARL (Majid et al., 2021). 3) EA has strong exploration ability, good robustness, and stable convergence (Sigaud, 2022).

As described above, EA offers numerous strengths that can complement the weaknesses of MARL. However, the efficient integration of both approaches in complex multi-agent collaborative tasks has not been thoroughly investigated. To this end, we propose a novel framework called **R**epresentation **A**symmetry and **C**ollaborative **E**volution (RACE), which combines EA with MARL to achieve efficient collaboration. RACE introduces an additional population of teams alongside the MARL team. However, independently maintaining and optimizing each team's policy is highly inefficient, as it fails to leverage the valuable knowledge acquired by other teams. Moreover, exploring collaboration in a vast nonlinear policy space is also inefficient. To address these challenges, we propose a two-level team policy construction, which decomposes the policies of different teams controlling the same agent into a nonlinear *shared* observation representation encoder and independent *linear* policy representations. We refer to the different representation scopes of the policy construction as *representation asymmetry*. The observation representation encoder, responsible for sharing the task-related and collaboration-related knowledge, is optimized using an integrated update direction derived from value function maximization involving all the EA teams and the MARL team collectively.

However, relying solely on value information is inadequate for efficient representation due to the issue of partial observability. To address the problem, RACE maximizes the correlation, i.e., Mutual information (MI), between the shared observation representations and the global states. However, maximizing in inferior states (i.e., states with low values) may induce negative influences on shared representations from poorly coordinated global information, resulting in suboptimal collaborations (Li et al., 2022). To overcome this limitation, we propose Value-Aware MI Maximization, which incorporates the normalized state values as weights to extract the superior global state information into shared observation representations. Building upon the useful knowledge conveyed by the shared observation representation, each team can be viewed as a collection of policy representations and searches for superior collaboration within the linear policy representation space rather than in the nonlinear parameter space as the convention does. Within the more compact and favorable linear space, policies explore more efficiently, thus facilitating collaboration.

Furthermore, RACE synergistically combines the strengths of EA and MARL by leveraging their respective advantages. Specifically, EA evolves populations and generates diverse experiences, which are then utilized by MARL. Conversely, the MARL team is optimized based on collected samples and periodically incorporated into the population for evolution. To facilitate efficient evolution, we introduce the *agent-level* crossover and mutation. The agent-level crossover only exchanges the corresponding individual policy representations in the two selected teams, enabling the exploration of better team composition. The agent-level mutation perturbs the policy representation for the specific agent in the team, driving the discovery of better individual policies for agent control. Finally, we evaluate RACE on 16 challenging tasks, comprising both complex continuous control and discrete micromanipulation scenarios. The experiment results demonstrate that RACE can significantly improve the basic MARL methods and outperform other baselines in various challenging tasks. Notably, our work demonstrates, for the first time, that EA has the capability to significantly enhance MARL performance in complex collaborative tasks.

2. Background

2.1. Preliminaries

We consider a fully cooperative multi-agent task where a team of agents operates in a stochastic, partially observable environment. The task can be modeled as a *decentralized partially observable Markov decision process* (Dec-POMDP) (Oliehoek & Amato, 2016), which is defined as a tuple: $\langle \mathcal{N}, \mathcal{S}, \mathcal{U}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. Here, $\mathcal{N} = \{1, \cdots, N\}$ denotes the set of N agents. In a Dec-POMDP, the complete state of the environment $s_t \in \mathcal{S}$ is not directly observable to the agents at each time step t. Instead, each agent $i \in \mathcal{N}$ can only observe its individual observation $o_t^i \sim \mathcal{O}(s_t, i)$. Each agent i uses a stochastic policy π_i to select actions $u_t^i \sim \pi^i(\cdot|o_t^i) \in \mathcal{U}^i$, resulting in a joint action $u_t = \{u_t^i\}_{i=1}^N \in \mathcal{U}$. After executing the joint action u_t in state s_t, the environment transitions to the next state s_{t+1} according to the transition function $\mathcal{T}(s_t, u_t)$, and the policies receive a team reward r_t from the reward function $\mathcal{R}(s_t, u_t)$, $\gamma \in [0, 1)$ is a discount factor. We denote the joint policy as $\pi = \{\pi^1, \cdots, \pi^N\} \in \Pi$, where Π is the joint policy space. In cooperative MARL, the collaborative team aims to find a joint policy that maximizes the total expected discounted return, denoted as $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. For instance, MADDPG (Lowe et al., 2017) and MATD3 (Ackermann et al., 2019) learn a centralized value function $Q_\psi(s_t, u_t)$ to optimize decentralized policies using Q-value maximization. To evaluate the contributions of individual policies to the team, QMIX (Rashid et al., 2018) and FACMAC (Peng et al., 2021) maintain a factored value function $Q_{tot}(s_t, \{Q^i\}_{i=1}^N)$ for credit assignment.

In addition, we adopt the Policy-extended Value Function Approximator (Tang et al., 2020) (PeVFA) in RACE to pre-

serve the values of multiple policies using a single value function. PeVFA, parameterized by θ, incorporates the policy representation χ_π as an additional input, resulting in the form $\mathbb{Q}_\theta(s, u, \chi_\pi)$. By explicitly including the policy representation χ_π, PeVFA exhibits an appealing characteristic of value generalization across policies within the policy space.

Evolutionary Algorithm (EA) (Bäck & Schwefel, 1993; Gomez et al., 2006) is a class of population-based gradient-free optimization methods. In EA, a population of policies $\mathbb{P} = \{\pi_1, \cdots, \pi_n\}$ is maintained, and policy search is performed iteratively. During each iteration, all policies interact with the environment, obtaining their fitness, denoted as $\{f(\pi_1), \cdots, f(\pi_n)\}$. The fitness is typically defined as the average Monte Carlo (MC) return over e episodes, formalized as $f(\pi_i) = \frac{1}{e}\sum_{i=1}^{e}[\sum_{t=0}^{T} r_t \mid \pi_i]$. Subsequently, the population is improved based on fitness using various EAs, including Genetic Algorithm (GA) and Evolutionary Strategy (ES). Taking GA as an example, the next generation is produced by selecting parents from the population based on some selection criteria (e.g., selecting individuals with high fitness) and applying crossover and mutation operators. Specifically, parents π_i and π_j generate offspring π_i' and π_j' using the crossover operator, denoted as $\pi_i', \pi_j' = \texttt{Crossover}(\pi_i, \pi_j)$, or using the mutation operator, denoted as $\pi_i' = \texttt{Mutation}(\pi_i)$. General crossover and mutation operate on policy parameters, such as the k-point crossover that randomly exchanges segment-wise (network) parameters between parents, and Gaussian mutation that adds Gaussian noise to the parameters. In this paper, we utilize GA for subsequent research.

2.2. Related Work

Centralized Training & Decentralized Execution (Lowe et al., 2017; Liu et al., 2022; 2023) (CTDE) is a widely adopted paradigm in MARL that offers improved scalability and deployability. During the training phase, agents can utilize global information for optimization. During the execution phase, agents make individual decisions based on their policies. For instance, MADDPG (Lowe et al., 2017) and MATD3 (Ackermann et al., 2019) employ a centralized value function to optimize decentralized policies. Other approaches such as VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2018), and FACMAC (Peng et al., 2021) achieve value function factorization under CTDE. In this paper, our focus is on investigating whether RACE can further enhance the performance of MARL in complex collaborative tasks. We select MATD3 and FACMAC as representative algorithms for our evaluation.

Evolutionary Reinforcement Learning (ERL) (Khadka & Tumer, 2018) is a hybrid method thats combine GA and DDPG (Lillicrap et al., 2016) for policy optimization. EA and RL simultaneously optimize the ultimate objective. In the concurrent optimization process, EA provides diverse experiences to the replay buffer for RL optimization. If the RL policy is superior to the individuals of the population, it is injected into the population to dictate the direction of population evolution, if it is inferior, it is eliminated. This hybrid approach is more capable of exploration, convergence, and robustness than RL algorithms, and is less sensitive to sparse, deceptive, and delayed rewards. Subsequent works (Pourchot & Sigaud, 2019; Bodnar et al., 2020; HAO et al., 2023a) have built upon the foundational framework of ERL and further improved its performance in single-agent settings. Notably, ERL-Re^2 (HAO et al., 2023a) addresses the inefficiency of maintaining population individuals separately by decomposing the population policy into shared state representation and linear policy representation. This decomposition promotes knowledge sharing among the population and has achieved state-of-the-art performance in benchmark MuJoCo tasks. However, there remains a large gap in multi-agent collaborative tasks.

Multi-Agent Evolutionary Reinforcement Learning (MERL) (Majumdar et al., 2020) is a method that incorporates GA into MARL. It aims to enhance sample efficiency by leveraging both sparse team rewards and dense agent-specific rewards. The optimization process in MERL involves two parts: GA maximizes the sparse team reward, and MARL utilizes gradient optimization to maximize the dense agent-specific rewards. However, the practical application of MERL faces challenges due to the lack of agent-specific rewards in many tasks. Furthermore, MERL is only evaluated on some easy tasks such as Predator Prey and the efficacy of MERL in solving complex control tasks and micromanipulation tasks has not been thoroughly validated.

3. MARL with Representation Asymmetry and Collaborative Evolution

This section introduces our framework **R**epresentation **A**symmetry and **C**ollaborative **E**volution (RACE). We begin by introducing the concept of Representation Asymmetry for team construction. Then, we detail how to learn the shared observation representation encoders. Next, we describe how to improve MARL with Collaborative Evolution. Finally, we provide an overview of RACE.

3.1. Representation-Asymmetry Team Construction

Beyond the MARL team, RACE introduces a population of teams. These teams learn collaboration through evolution and reinforcement. Typically, each team maintains separate policies for decision-making and optimization. However, this independent policy construction limits knowledge sharing across teams and makes exploration in large policy spaces inefficient. Inspired by ERL-Re^2 (HAO et al., 2023a), we propose **R**epresentation-**A**symmetry **T**eam **C**onstruction

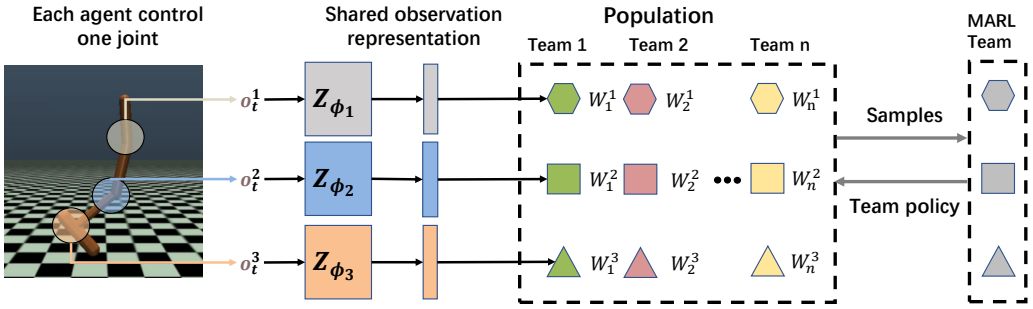

Figure 1. The conceptual illustration of Representation-Asymmetry Team Construction (RATC) on 3-Agent Hopper task. All policies are composed of the nonlinear shared observation representation encoder Z_{ϕ_i} and an individual linear policy representation W_j^i where j denotes the team index and i denotes the policy index in the team. RACE maintains a population of teams (denoted by green, red, and yellow) and a MARL team (grey). By sharing the observation representation encoders, each team is composed of multiple policy representations (denoted by hexagon, square, and triangle). In the population, each row controls the same agent (joint) and each column constructs the team policy representation $W_j = \{W_j^i\}_{i=1}^N$.

(RATC) to enable efficient knowledge sharing and policy exploration. The illustration of RATC is shown in Fig. 1. Specifically, the policies that control the same agent in different teams are composed of a *shared* nonlinear observation representation encoder $z_t^i = Z_{\phi_i}(o_t^i) \in \mathbb{R}^d$ (given the observation o_t^i of the agent i) and an *individual* linear policy representation $W_j^i \in \mathbb{R}^{(d+1)\times|\mathcal{U}^i|}$, where j denotes the team index and i denotes the policy index in the team. The i-th policies in all teams share the same observation representation encoder $Z_{\phi_i}(o_t^i)$ and make decisions by combining the shared observation representation encoder and the policy representation. The decision-making process can be described as follows:

$$\pi_j^i(o_t^i) = \texttt{act}(Z_{\phi_i}(o_t^i)^\intercal W_{j,[1:d]}^i + W_{j,[d+1]}^i) \in \mathbb{R}^{|\mathcal{U}^i|},$$

where $W_{j,[m(:n)]}^i$ denotes the slice of matrix W_j^i that consists of row m (to n) and $\texttt{act}(\cdot)$ denotes the activation function. On the foundation of shared observation representations, only policy representations differ among teams. Thus the team j can be defined as $W_j = \{W_j^i\}_{i=1}^N$ and makes decisions by $\pi_j(s_t) = \{\pi_j^1(o_t^1), \cdots, \pi_j^N(o_t^N)\}$. Intuitively, we expect the shared observation representation encoders to provide rich task-related and collaborative knowledge that benefits all policies controlling the same agent. The shared observation representation encoder determines a more compact and favorable linear policy space for controlling the agent i denoted by $\Pi(\phi_i)$, where we conduct evolution and reinforcement. Formally, we summarize the construction of individual, team, and population in RACE below:

Team j policy i: $\pi_j^i(o_t^i) = \texttt{act}(Z_{\phi_i}(o_t^i)^\intercal W_{j,[1:d]}^i + W_{j,[d+1]}^i)$
Team policy of team j: $\pi_j(s_t) = \{\pi_j^1(o_t^1), \cdots, \pi_j^N(o_t^N)\}$
Construction of team j: $W_j = \{W_j^1, W_j^2, \cdots, W_j^N\}$
Team Population: $\mathbb{P} = \{W_1, W_2, \cdots, W_n\}$

3.2. Shared Observation Representation Learning

Thanks to the team construction, all policies learn collaboration in the linear policy space $\Pi(\phi_i)$, which poses two demands: (1) The shared observation representation encoder Z_{ϕ_i} should provide useful knowledge about collaboration and tasks; (2) The knowledge is required to be beneficial to all teams, not just one particular team. To achieve this, we propose to learn the shared observation representation encoders with *value function maximization* regarding the corresponding policies in all teams. Specifically, we learn a centralized Policy-extended Value Function Approximator (PeVFA) (introduce in Section 2.1) $\mathbb{Q}_\theta(s, u, W_j)$ to estimate the value for the team policy representations $W_j \sim \mathbb{P}$; For the MARL team, the conventional centralized critic $Q_\psi(s, u)$ is maintained. The loss functions of $\mathbb{Q}_\theta$ and Q_ψ are formulated below:

$$\begin{aligned}\mathcal{L}_\theta &= \mathbb{E}_\mathcal{D}\left[\left(r + \gamma \mathbb{Q}_{\theta'}\left(s', \pi_j(s'), W_j\right) - \mathbb{Q}_\theta\left(s, u, W_j\right)\right)^2\right], \\ \mathcal{L}_\psi &= \mathbb{E}_\mathcal{D}\left[\left(r + \gamma Q_{\psi'}\left(s', \pi'_{marl}(s')\right) - Q_\psi\left(s, u\right)\right)^2\right],\end{aligned} \tag{1}$$

where W_j is sampled form $\mathbb{P}$, s, u, r, s' are sampled from the experience buffer D collected by all teams, $Q_{\theta'}, Q_{\psi'}$ denote the target networks of the PeVFA and the MARL critic, respectively. π'_{marl} denote the target policies (actors).

For all teams, the individual update direction of the shared encoder associated with a specific agent i can be obtained. This is done by computing $\nabla_{\phi_i}\mathbb{Q}_\theta(s, \pi_j(s), W_j)$ for any $W_j \in \mathbb{P}$ or $\nabla_{\phi_i} Q_\psi(s, \pi_{marl}(s))$ through π_j and π_{marl} respectively. This is the value function maximization principle where we adjust Z_{ϕ_i} to induce superior policy (space) for controlling the corresponding agents i. However, it is crucial to ensure that the optimization direction of Z_{ϕ_i} is beneficial for all individuals in the team. Instead of relying solely on individual update directions, it is necessary to integrate the

update directions from all teams. Therefore, the *Value Function Maximization* loss for shared observation representation encoder is defined:

$$\mathcal{L}_{\phi_i}^{\text{VFM}} = -\mathbb{E}_{\mathcal{D},\mathbb{P}}\Big[Q_\psi\left(s, \pi_{marl}\left(s\right)\right) + \mathbb{Q}_\theta\left(s, \pi_j\left(s\right), W_j\right)\Big]. \tag{2}$$

By minimizing Eq. 2, the shared observation representation encoder Z_{ϕ_i} is optimized towards a superior policy space Π_{ϕ_i} about all policies that control the same agent iteratively. In practice, it is feasible to maintain only one centralized PeVFA for both MARL and EA teams, but to make RACE a plug-and-play component, we maintain the original MARL critic as the convention does, thus avoiding the performance impact introduced by modifications to the value function.

However, only using the value information is inadequate since most tasks in Multi-Agent systems are partially observable, where agents can not access global information, and thus non-stationarity throughout the execution and learning phases is exacerbated. Thus we propose to maximize the mutual information (MI) between the shared observation representations $z_i = Z_{\phi_i}(o^i)$ and global state s to make z_i reflect global information thus alleviating the problem of partial observations. However, maximizing MI with inferior states may induce a negative influence on shared observation representations from the global information of poor collaboration, leading to suboptimality (Li et al., 2022). To this end, we propose a novel Value-Aware MI Maximization to extract the superior global information into z_i. Specifically, we first approximate the MI lower bound of representations z_i and states s by Mutual Information Neural Estimation (MINE) (Belghazi et al., 2018) as follows:

$$I(z_i; s) \geq \sup_{\omega\in\Omega} \underbrace{\mathbb{E}_{\mathbb{P}_{\mathcal{S}\mathcal{Z}_i}}\left[-sp\left(-T_\omega(s_t, z_{i,t})\right)\right] - \mathbb{E}_{\mathbb{P}_{\mathcal{S}}\otimes\mathbb{P}_{\mathcal{Z}_i}}\left[sp\left(T_\omega(s_t, z_{i,k})\right)\right]}_{I_{lb}(z_i;s)}, \tag{3}$$

where $z_{i,t}$ is the shared observation representation of the agent i at time t, $\mathbb{P}_{\mathcal{S}\mathcal{Z}_i}$ is the joint probability distribution, $\mathbb{P}_{\mathcal{S}}$ and $\mathbb{P}_{\mathcal{Z}_i}$ are the marginals. T_ω is a neural network with parameters $\omega \in \Omega$, $sp(z) = \log(1 + \exp(z))$. We can use the lower bound $I_{lb}(z_i, s)$ in Eq. 3 to approximate the MI and maximize it to extract the global information into z_i. It is worth noting that $I_{lb}(z_i, s)$ is equivalent to the expectation of $I_t(z_{i,t}, s_t) = -sp\left(-T_{\omega_1}(s_t, z_{i,t})\right) - \mathbb{E}_{\mathbb{P}_{\mathcal{S}}\otimes\mathbb{P}_{\mathcal{Z}_i}}\left[sp\left(T_{\omega_1}(s_t, z_{i,k})\right)\right]$, thus $I_t(z_{i,t}, s_t)$ can be considered as the per-step signals of MI, which are then selectively maximized to extract the superior states into z_i. To achieve the ultimate objective, we employ a value function $V_\zeta(s)$ that estimates the best return of state s on all teams. We implement it by minimizing the following loss:

$$\mathcal{L}_\zeta = \mathbb{E}_{s\sim D}\left[\left(V(s) - F(s)\right)^2\right]. \tag{4}$$

$F(s)$ should be defined as the maximum target value of actions taken by all teams under s. To reduce computational expenses, we approximate it by considering only the maximum target value obtained from the actions taken by the MARL team and a randomly selected team from the population. Thus $F(s)$ can be defined as $r + \max(Q_{\psi'}(s', \pi_{marl}(s')), \mathbb{Q}_{\theta'}\left(s', \pi_j(s'), W_j\right))$, where $Q_{\psi'}(s', \pi_{marl}(s'))$ and $\mathbb{Q}_{\theta'}\left(s', \pi_j(s'), W_j\right)$ can directly utilize the intermediate results obtained during the optimization of Eq. 1. Subsequently, we use the normalized values of $V_\zeta(s)$ as the weights of I_t . The loss of Value-Aware MI Maximization can be defined as follows:

$$\mathcal{L}_{\phi_i}^{\text{VMM}} = -\mathbb{E}_D\left[\frac{V_\zeta(s_t) - \min\limits_{s_j\sim D}(V_\zeta(s_j))}{\max\limits_{s_j\sim D}(V_\zeta(s_j)) - \min\limits_{s_j\sim D}(V_\zeta(s_j))} I_t(z_i^t, s_t)\right]. \tag{5}$$

Intuitively, the shared observation representations capture more of the global information with high values rather than the ones with low values by minimizing Eq. 5.

Finally, the loss function of Z_{ϕ_i} is defined as:

$$\mathcal{L}_{\phi_i} = \mathcal{L}_{\phi_i}^{\text{VFM}} + \beta\mathcal{L}_{\phi_i}^{\text{VMM}}, \tag{6}$$

where β is the hyperparameter to balance the impact of Value-Aware MI Maximization loss.

3.3. Improving MARL with Collaborative Evolution

Thanks to the *Value Function Maximization* and *Value-Aware MI Maximization*, shared observation representations not only provide the collaboration-related and task-related knowledge to build favorable policy space for efficient exploration but also capture high-quality global information, thus mitigating the challenges posed by partial observations. Based on the shared observation representation encoder Z_{ϕ_i}, the policies of different teams controlling the same agents optimize their policy representations more efficiently in the linear policy space $\Pi(\phi_i)$ than in the original non-linear policy space. In the following, we detail how to achieve collaborative evolution in the linear policy space.

For evolution, RACE first evaluates n teams in the population and selects the best-performing team as the elite team. Then crossover and mutation are performed. For crossover, two teams should be selected. The elite team serves as one parent to produce offspring. The other parent is selected by the tournament mechanism (Khadka & Tumer, 2018) (filtering the best-performing team in 3 randomly selected teams). Teams not selected as parents are replaced by the offspring. Additionally, all non-elite teams have a certain probability of mutation.

To achieve more efficient evolution, we design the *agent-level* crossover and mutation for both team and individual exploration. For team exploration, we randomly exchange the individual policy representations that control the same

agent in the two selected teams, promoting the exploration of better team compositions. For individual exploration, we introduce random parameter perturbation to some policy representations for the selected team, driving the discovery of better individual policies for agent control. These operations are formulated as follows:

$$\begin{aligned} W_i', W_j' &= ((W_i - W_i^{d_i}) \cup W_j^{d_i}, (W_j - W_j^{d_j}) \cup W_i^{d_j}) \\ &= \texttt{Crossover}(W_i, W_j), \\ W_j' &= (W_j - W_j^{d_j}) \cup P(W_j^{d_j}) = \texttt{Mutation}(W_j), \end{aligned}$$

where W_i and W_j are two selected teams, d_i, d_j are the randomly sampled subsets of agents indices $\{1, \cdots, N\}$, and P is the perturbation function that adds Gaussian noise to (or reset) certain parameters. We use W^d to denote the subset of the policy representations of the team with indices d. Thanks to the *agent-level* operators, the population can achieve more efficient and stable evolution, as well as more intuitive semantic meaning on teams and individuals.

During the evolution process, populations efficiently explore the policy space to develop collaboration policies. Additionally, the samples generated throughout population evolution can be utilized for training the MARL team. The learning process of the MARL team, denoted as W_{marl}, follow the standard policy optimization approach of MARL, with two notable distinctions: 1) policy optimization occurs in a linear policy space, 2) the optimization utilizes samples collected by all teams. Taking MADDPG (Lowe et al., 2017) as a representative example, the loss function for W_{marl} is defined below, based on the centralized critic Q_ψ (learned using Eq. 1):

$$\mathcal{L}_{\text{MARL}}(W_{marl}) = -\mathbb{E}_{s\sim\mathcal{D}}\Big[Q_{\psi_i}\left(s, \pi_{marl}(s)\right)\Big], \quad (7)$$

where $\mathcal{D}$ stores the off-policy experiences collected by both the MARL team and the EA teams. Furthermore, at the end of each iteration, the population incorporates the MARL policy representation W_{marl} for evolution. This reciprocal interaction enables populations to provide high-quality samples to MARL for optimization, while MARL, in turn, offers potentially superior policies to assist population evolution.

3.4. The Algorithm Framework of RACE

In principle, RACE is a general framework that can be implemented with most MARL algorithms. In this paper, we employ MATD3 (Ackermann et al., 2019) and FACMAC (Peng et al., 2021) as the fundamental MARL algorithms. The general pseudo-code of RACE is presented in Algorithm 1. Each iteration of RACE consists of three distinct phases, indicated in blue. First, each team of the population and the MARL team interact with the environment and collect experiences. The teams in the population $\mathbb{P}$ obtain the cumulative rewards of one episode as the fitness for evolution

Algorithm 1: RACE

1 **Initialize:** Replay buffer $\mathcal{D}$, the population size n, the team size N, the shared observation representation encoders $Z_{\phi_1}, \cdots, Z_{\phi_N}$, the MARL team W_{marl}, the population $\mathbb{P} = \{W_1, \cdots, W_n\}$, a state value function V_ζ, a MINE T_ω for each agent, the MARL centralized critic Q_ψ and the centralized PeVFA $\mathbb{Q}_\theta$ (target networks are omitted here)
2 **repeat**
3 # Rollout both the teams in the population $\mathcal{P}$ and MARL team with $\{Z_{\phi_1}, \cdots, Z_{\phi_N}\}$ and obtain the fitness
4 Rollout each team in $\mathbb{P}$ for one episode and evaluate its fitness $\{f(W_1), \cdots, f(W_n)\}$ by summing the undiscounted reward $f(W_i) = \sum_{t=0}^{T}[r_t \mid W_i]$
5 Rollout the MARL team for one episode
6 Store the experiences generated by $\mathbb{P}$ and W_{marl} to $\mathcal{D}$
7 # Evolution and reinforcement in the linear policy space
8 Train $\mathbb{Q}_\theta$, Q_ψ and V_ζ with D ▷ see Eq. 1 and Eq. 4
9 **Optimize the population:** perform the genetic operators (i.e., selection, crossover, and mutation)
10 **Optimize the MARL policies:** update W_{marl} (by e.g., MADDPG, MATD3) according to Q_ψ ▷ see Eq. 7
11 Inject MARL team policy to the population $\mathbb{P}$
12 # Optimize the shared representations $\{Z_{\phi_1}, \cdots, Z_{\phi_N}\}$
13 Update the shared observation representation encoders Z_{ϕ_i} with *Value Function Maximization* and *Value-Aware MI Maximization* ▷ see Eq. 6
14 **until** *reaching maximum training steps*;

(Line 4-6). Subsequently, evolution and reinforcement occur in the linear policy space provided by the currently shared observation representation encoders $\{Z_{\phi_1}, \cdots, Z_{\phi_N}\}$. The teams in the population $\mathbb{P}$ are optimized using genetic operators (Line 9). The MARL team learns from the experiences collected by all teams and periodically injects policies to $\mathbb{P}$ (Line 10-11). Finally, the shared observation representation encoders are updated to enhance the policy space for the next iteration (Line 13). Overall, we provide the technical details of RACE and describe how it can be combined with MARL algorithms.

4. Experiments

This section empirically evaluates RACE to answer the following research questions (1) **RQ1**: Can RACE improve MARL and outperform the related baselines in complex multi-agent cooperative tasks? (2) **RQ2**: Are the shared observation representation encoders optimized by Eq. 6 efficient? Are the *agent-level* crossover and mutation more efficient? (3) **RQ3**: How do the hyperparameters α and β impact the performance of RACE?

4.1. Experimental Setups

For a comprehensive comparative study, we evaluate RACE on tasks with both continuous and discrete action spaces. For continuous tasks, we integrate RACE with MATD3 (Ackermann et al., 2019) and evaluate it on eight

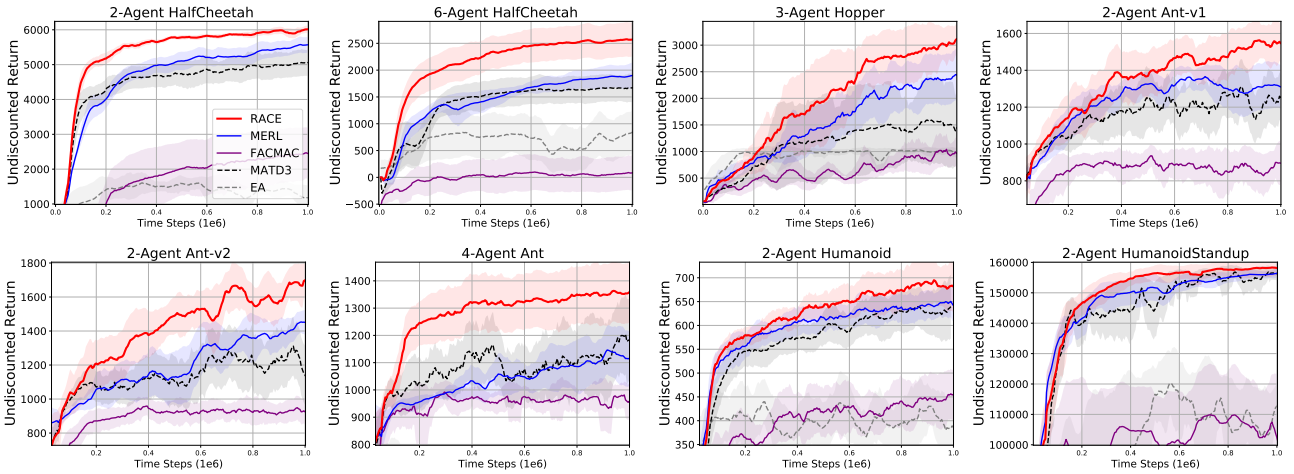

Figure 2. Performance comparison between RACE and baselines in Multi-Agent MuJoCo (All in MATD3 version).

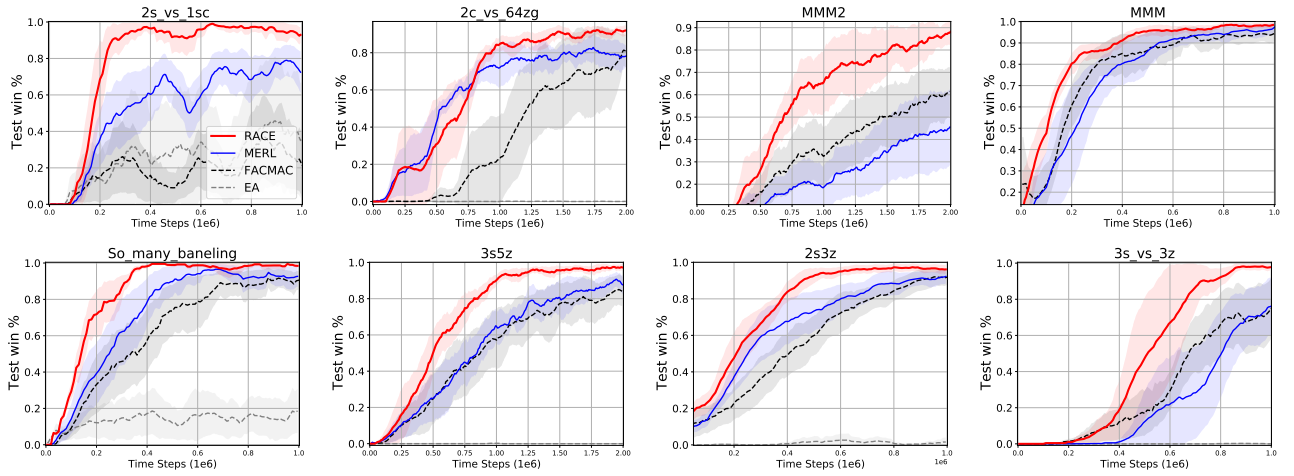

Figure 3. Test win rate comparison between RACE and baselines in SMAC (All in FACMAC version).

cooperative continuous control tasks from the Multi-Agent MuJoCo benchmark (Peng et al., 2021). These tasks involve controlling different joints of the robot with various morphologies to complete tasks such as standing or walking. Each agent can only observe its own joint information. For discrete tasks, we integrate RACE with FACMAC and evaluate it in the StarCraft II micromanagement environments (Samvelyan et al., 2019) (SMAC), which feature high control complexity and require learning policies in large discrete action space. We compare RACE with the following baselines: MATD3 (Ackermann et al., 2019), MERL (Majumdar et al., 2020), EA (Khadka & Tumer, 2018), and FACMAC (Peng et al., 2021). We use the official implementation of these algorithms for comparison. MATD3 is an extension of the official TD3 (Fujimoto et al., 2018) implementation in the CTDE framework. We implement RACE on the official code of EA and basic MARL algorithms while keeping the other hyperparameters and processes consistent. We fine-tune all baselines to provide their best performance. It is worth noting that since only one team reward is available for these tasks, MERL cannot be directly applied. Therefore, we optimize the team reward through EA and MARL collectively in the MERL baseline.

To ensure statistical significance, we conduct 5 independent runs with the same seeds from 1 to 5 and report the average results with 95% confidence intervals. The hyperparameters specific to RACE are set as follows: the population size of 5 for all tasks, α selected from $\{0.2, 0.5, 0.7, 1.0\}$ for Multi-Agent MuJoCo, α selected from $\{0.005, 0.005, 0.01, 0.05, 0.1\}$ for SMAC, and β selected from $\{0.1, 0.01, 0.001, 0.0001\}$. Further implementation details are provided in Appendix A.

4.2. Performance

To answer RQ1, our focus is on experimentally verifying whether RACE can significantly enhance the performance of MARL algorithms. We begin by integrating RACE with

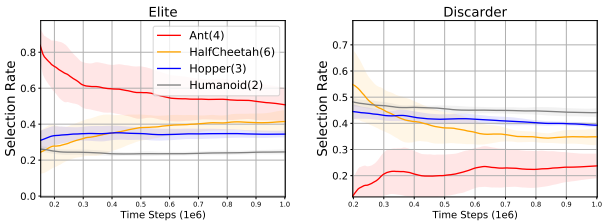

Figure 4. Elite rate and discarded rate on different tasks.

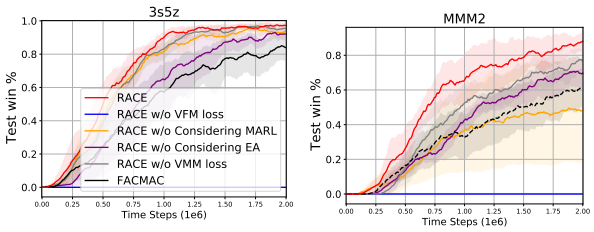

Figure 5. Experiments about how to update the shared observation representation encoders. RACE w/o Considering MARL/EA means only maximizing $\mathbb{Q}_\theta$/Q_ψ in VFM.

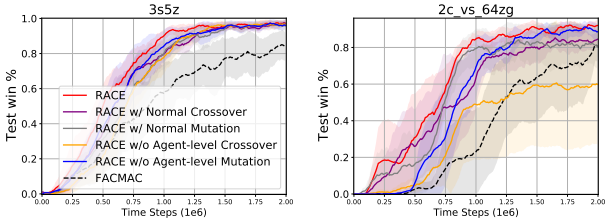

Figure 6. Ablation study on *Agent-Level* operators.

MATD3 (Ackermann et al., 2019) and evaluating its performance (RACE (MATD3)) alongside other related baselines in the Multi-Agent MuJoCo environment. The results shown in Figure 2 demonstrate that RACE (represented in red) significantly improves upon MATD3 (represented in black) and outperforms other baselines in most tasks. These findings clearly demonstrate the superiority of RACE in tackling challenging continuous control tasks. Notably, our experiments reveal that RACE consistently yields performance gains as the number of agents increases from 2 to 6 on the HalfCheetah task and 2 to 4 on the Ant task.

To further assess the generality of RACE, we integrate it with FACMAC (Peng et al., 2021) and evaluate its performance (RACE (FACMAC)) in SMAC. The results depicted in Fig.3 indicate that RACE (represented in red) can further enhance FACMAC (represented in black) and outperform other baselines. Specifically, RACE exhibits faster convergence and achieves higher performance. Overall, these experiments highlight RACE as an efficient framework that can be seamlessly integrated with multiple MARL algorithms, offering substantial improvements in both challenging continuous and discrete tasks.

To investigate the impact of EA and MARL on collaboration, we analyze the elite rate and discarded rate of the MARL team within the population, as depicted in Figure 4. We observe that, in most environments, both the elite rate and discarded rate hover around 40%. Notably, the teams maintained by EA are more likely to be selected as the elite compared to those guided solely by Reinforcement Learning (RL). This finding underscores the vital role played by EA in exploring efficient collaboration. However, on the 4-Agent Ant task, MARL achieves a higher elite rate and lower discarded rate, indicating that MARL assumes a leading role while EA plays a supporting role in this scenario.

4.3. Analysis of Components and Hyperparameter

To answer RQ2, we study whether jointly optimizing the shared observation representation encoders with *Value Function Maximization* (VFM) and *Value-Aware MI Maximization* (VMM) yields better results compared to optimizing them individually. Additionally, we investigate whether only maximizing Q_ψ (MARL) or maximizing $\mathbb{Q}_\theta$ (EA) is superior to VFM.

The results in Fig. 5 reveal the following findings: (1) Jointly optimizing the shared observation representation encoders with VFM and VMM is more efficient than using VFM/VMM alone. (2) Maximizing the value function of EA and MARL is preferable to maximizing EA/MARL alone, indicating that constructing shared representations that favor only MARL or EA does not promote efficient learning for all teams.

It's worth noting that RACE demonstrates greater sample efficiency and faster convergence. However, using the encoder with only VMM significantly damages performance (0% win rate on SMAC tasks). This result indicates that the performance of RACE is more dependent on VFM than VMM. This can be attributed to the model's core architecture, which includes team policy construction involving a nonlinear observation representation encoder and a linear policy representation. The linear form of the policy representation helps compact the policy search space and facilitate policy search. However, it necessitates the observation encoder to provide rich information about the task and collaboration, as it determines the policy space. While VMM can only provide some global information about the good states. The linear policy space constructed by VMM-based observation encoders relies on the linear policy representation to extract task-relevant information, leading to limited expressiveness and performance collapse. Thus, Eq.2 plays a critical role.

Additional experiments conducted on all 16 tasks confirm that using the observation encoder with VFM improves the basic algorithm's performance by 27.1% on all tasks,

including 8 MAMUJOCO tasks (26.7%) and 8 SMAC tasks (27.4%). Furthermore, using the encoder with VFM and VMM enhances performance by 34.4% compared to the basic algorithm. While using only the EA part or the MARL part of VFM leads to degraded performance of 9% and 11.5%, respectively.

To verify the superiority of the agent-level operators, we perform an ablation study on the *Agent-level* crossover and mutation and compare them with the normal crossover and mutation operators. The normal operators operate directly on the parameter values without considering the individual agent characteristics. The results in Fig. 6 demonstrate that removing or replacing either agent-level crossover or agent-level mutation degrades performance. This illustrates the efficiency of team search and individual search. Team search helps find a better team composition, while individual search promotes the discovery of superior individuals. Both *Agent-level* operators exhibit greater stability and efficiency compared to the normal operators.

To further support this claim, we conduct a comparison between RACE with and without the agent-level operators on all 8 SMAC tasks. The results show that using agent-level crossover and mutation outperforms the use of normal crossover and mutation by 6.8%, with improved stability in performance and faster convergence.

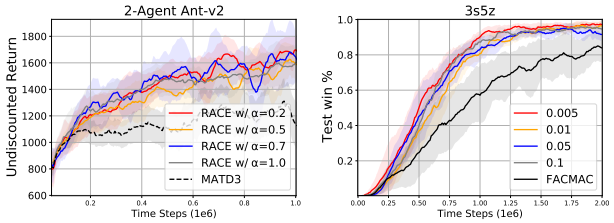

Figure 7. Analysis on hyperparameter α.

To answer RQ3, we first analyze the hyperparameter α, which controls the proportion of mutated parameters in the mutation process. The results in Fig. 7 indicate that the performance across different α is comparable, and proper adjustment of α can provide better results. For the continuous control tasks (Multi-Agent MuJoCo), α is chosen from $\{0.2, 0.5, 0.7, 1.0\}$. These tasks are generally insensitive to small policy perturbations, large perturbations are efficient for exploration. For the micromanipulation tasks (SMAC), α is chosen from $\{0.005, 0.005, 0.01, 0.05, 0.1\}$. These micromanagement tasks are highly sensitive to small policy changes, which can result in significant behavioral differences.

We further analyze the hyperparameter β to balance the impact of Value-Aware MI Maximization in Eq. 6. The results in Fig. 8 demonstrate that RACE is sensitive to β which influences the quality of the shared representations. When

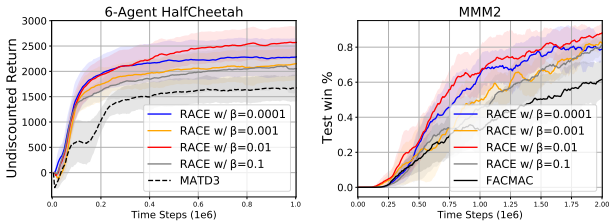

Figure 8. Analysis on hyperparameter β.

β is set to a large value, the representations contain more global information than task-related information. Since the importance of global and task-related information varies across tasks, it is necessary to appropriately adjust β for each task.

5. Conclusion

To address the reward sensitivity and non-stationarity imposed by MARL, we propose a hybrid framework RACE, which uses EA to further improve MARL. For efficient knowledge sharing and policy exploration, RACE decomposes the team policy into shared observation representation encoders and individual policy representations. With the shared observation representations optimized by *Value Function Maximization* and *Value-Aware MI Maximization*, knowledge of tasks and global information can be efficiently conveyed across different teams, and collaboration is more easily formed in linear policy space. To achieve efficient evolution, the *agent-level* crossover and mutation are proposed to facilitate team policy (composition) exploration and individual exploration. Finally, we integrate RACE with different MARL algorithms and demonstrate that RACE can further improve MARL in a wide range of challenging cooperative environments with both continuous action space and discrete action space.

Acknowledgments

This work is supported by the National Key R&D Program of China (Grant No. 2022ZD0116402), the National Natural Science Foundation of China (Grant No. 62106172), and the Natural Science Foundation of Tianjin (No. 22JCQNJC00250).

References

Ackermann, J., Gabler, V., Osa, T., and Sugiyama, M. Reducing overestimation bias in multi-agent domains using double centralized critics. *CoRR*, 2019.

Bäck, T. and Schwefel, H. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1993.

Belghazi, I., Rajeswar, S., Baratin, A., Hjelm, R. D., and Courville, A. C. MINE: mutual information neural estimation. *CoRR*, 2018.

Bodnar, C., Day, B., and Lió, P. Proximal distilled evolutionary reinforcement learning. In *AAAI*, 2020.

Christianos, F., Schäfer, L., and Albrecht, S. V. Shared experience actor-critic for multi-agent reinforcement learning. In *NeurIPS*, 2020.

Fujimoto, S., v. Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *ICML*, 2018.

Gomez, F. J., Schmidhuber, J., and Miikkulainen, R. Efficient non-linear control through neuroevolution. In *ECML*, 2006.

HAO, J., Li, P., Tang, H., ZHENG, Y., Fu, X., and Meng, Z. ERL-re$ˆ2$: Efficient evolutionary reinforcement learning with shared state representation and individual policy representation. In *ICLR*, 2023a.

HAO, J., X.Hao, Mao, H., Wang, W., Yang, Y., Li, D., Zheng, Y., and Wang, Z. Boosting multiagent reinforcement learning via permutation invariant and permutation equivariant networks. In *ICLR*, 2023b.

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J. A., Solowjow, E., and Levine, S. Residual reinforcement learning for robot control. In *ICRA*, 2019.

Khadka, S. and Tumer, K. Evolution-guided policy gradient in reinforcement learning. In *NeurIPS*, 2018.

Li, M., Qin, Z., Jiao, Y., Yang, Y., Wang, J., Wang, C., Wu, G., and Ye, J. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *WWW*, 2019.

Li, P., Tang, H., Yang, T., Hao, X., Sang, T., Zheng, Y., Hao, J., Taylor, M. E., Tao, W., and Wang, Z. PMIC: improving multi-agent reinforcement learning with progressive mutual information collaboration. In *ICML*, 2022.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

Liu, Z., Zhu, Y., Wang, Z., Gao, Y., and Chen, C. MIXRTs: Toward interpretable multi-agent reinforcement learning via mixing recurrent soft decision trees. *arXiv preprint*, 2022.

Liu, Z., Zhu, Y., and Chen, C. NA^2Q: Neural attention additive model for interpretable multi-agent q-learning. *arXiv preprint*, 2023.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS*, 2017.

Majid, A. Y., Saaybi, S., Rietbergen, T., François-Lavet, V., Prasad, R. V., and Verhoeven, C. J. M. Deep reinforcement learning versus evolution strategies: A comparative survey. *CoRR*, 2021.

Majumdar, S., Khadka, S., Miret, S., McAleer, S., and Tumer, K. Evolutionary reinforcement learning for sample-efficient multiagent coordination. In *ICML*, 2020.

Ni, F., Hao, J., Lu, J., Tong, X., Yuan, M., Duan, J., Ma, Y., and He, K. A multi-graph attributed reinforcement learning based optimization algorithm for large-scale hybrid flow shop scheduling problem. In *KDD*, 2021.

Oliehoek, F. A. and Amato, C. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

Papoudakis, G., Christianos, F., Rahman, A., and Albrecht, S. V. Dealing with non-stationarity in multi-agent deep reinforcement learning. *CoRR*, 2019.

Peng, B., Rashid, T., de Witt, C. S., Kamienny, P., Torr, P., Boehmer, W., and Whiteson, S. FACMAC: factored multi-agent centralised policy gradients. In *NeurIPS*, 2021.

Pourchot, A. and Sigaud, O. CEM-RL: combining evolutionary and gradient-based methods for policy search. In *ICLR*, 2019.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J. N., and Whiteson, S. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, 2018.

Salimans, T., Ho, J., Chen, X., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, 2017.

Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C., Torr, P. H. S., Foerster, J. N., and Whiteson, S. The starcraft multi-agent challenge. In *AAMAS*, 2019.

Sigaud, O. Combining evolution and deep reinforcement learning for policy search: a survey. *CoRR*, 2022.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V. F., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*, 2018.

Tang, H., Meng, Z., Hao, J., Chen, C., Graves, D., Li, D., Liu, W., and Yang, Y. What about taking policy as input of value function: Policy-extended value function approximator. *CoRR*, 2020.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gülçehre, Ç., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 2019.

Yang, Y., Hao, J., Chen, G., Tang, H., Chen, Y., Hu, Y., Fan, C., and Wei, Z. Q-value path decomposition for deep multiagent reinforcement learning. In *ICML*, 2020a.

Yang, Y., Hao, J., Liao, B., Shao, K., Chen, G., Liu, W., and Tang, H. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint*, 2020b.

Yuan, Y., HAO, J., Ni, F., Mu, Y., ZHENG, Y., Hu, Y., Liu, J., Chen, Y., and Fan, C. EUCLID: Towards efficient unsupervised reinforcement learning with multi-choice dynamics model. In *ICLR*, 2023.

A. Method Implementation Details

All experiments are carried out on NVIDIA GTX 2080 Ti GPU with Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz.

A.1. Network Architecture

For EA, we use ERL-Re^2 codebase[1], and all the processes and hyperparameters remain the same.

For the basic MARL algorithms, MATD3 is a simple extension of TD3 in the CTDE paradigm and we follow the official code [2], FACMAC follows the official implementation[3]. To reiterate, the above process and the hyperparameters introduced by these algorithms remain unchanged.

For structures specific to RACE, the centralized RACE is the same as the MARL critic except for taking policy representations as extra inputs. In Multi-Agent MuJoCo, the shared observation representation encoders are constructed by two fully connected layers with 400 and 300 units. The policy representation is the final layer that controls one agent's actions. In SMAC, the shared observation representation encoders use the first n-1 layers of policy networks in FACMAC. The policy representation is the final layer. In the following, we detail the structures which are specific to RACE.

In RACE (MATD3), the centralized PeVFA takes the state, actions, and the team policy representation W_j as inputs and maintains double Q networks which are similar to MATD3. The team policy representation can be regarded as a combination of a matrix with shape $[300, |\mathcal{U}|] = [300, \sum_{i=1}^{N} |\mathcal{U}^i|]$ (i.e., weights) and a vector with shape $[|\mathcal{U}|]$ (i.e., biases) which can be concatenated as a matrix with shape $[300 + 1, |\mathcal{U}|]$. We first encode each vector with shape $[300 + 1]$ of the team policy representations with 3 fully connected layers with units 64 and `leaky_relu` activation function. Thus we can get an embedding list with shape $[64, |\mathcal{U}|]$ and get the final team policy embedding with shape $[64]$ by taking the mean value of the embedding list in the action dimension. With the team policy embedding, we concatenate it with states and actions as the input to an MLP with 2 fully connected layers with units 400 and 300 and get the predicted value by the centralized PeVFA. The activation functions in the centralized PeVFA all use `leaky_relu`. We list structures in Table 1 and 2.

In RACE (FACMAC), the overall process is the same as RACE (MATD3) except that we use the structures of FACMAC. FACMAC maintains a shared policy network and a shared critic network, in addition to a Mixer Net for credit assignments. We introduce an extra Critic network with team policy representations as the inputs (i.e., Critic PeVFA) and an extra Mixer network with team policy representations as the inputs (i.e., Mixer PeVFA). The policy representations are processed in the same way as in Table. 2 and subsequently spliced with observation and action/Q value as inputs. To better confirm that the performance improvement is brought by RACE, we do not maintain a separate observation representation encoder for each agent, which means the observation representation encoder is shared for different agents in the same team. In fact, inter-team sharing is orthogonal to intra-team sharing. To be consistent and fair comparisons (without changing the basic MARL structure), the MARL team uses the shared observation representation encoders and a shared policy representation. But for each EA team, RACE maintains independent policy representations for each policy.

To calculate the Mutual Information, we leverage MINE (Belghazi et al., 2018) adopted in PMIC[4] (Li et al., 2022). In general, the number of MINE maintained corresponds to the number of shared observation representation encoders. Consequently, we maintain N MINEs in Multi-Agent MuJoCo tasks and one MINE is maintained in SMAC tasks. The structure of the MINE network, i.e., T_ω, can be divided into two parts: One for encoding the global states, which consists mainly of two fully connected layers with 128 units with leaky_relu activation function. The other part is used to encode the shared observation representations, which consists of one 128-cell fully connected layer. The hidden variables encoded by states and shared representations are finally calculated by the equation (3) to obtain the loss, and minimizing this loss maximizes the mutual information. To calculate *Value-Aware MI Maximization*, we maintain a value network to estimate the expected value of states to determine whether a state is a good state. The structure of the value network consists of two fully connected layers with 400 and 300 units with leaky_relu activation function in MAMUJOCO and 32 units with relu in SMAC.

[1] https://github.com/yeshenpy/ERL-Re2
[2] https://github.com/sfujim/TD3
[3] https://github.com/oxwhirl/facmac
[4] https://github.com/yeshenpy/PMIC

Table 1. The structures of the shared observation representation encoder and policy representations in MATD3.

Shared Observation Representation Encoder	Policy Representation
$(\texttt{obs\_dim}, 400)$	$(300, \lvert\mathcal{U}\rvert)$
tanh	tanh
(400,300)	
tanh	

Table 2. The structure of the centralized PeVFA in RACE (MATD3)

PeVFA	
$(\lvert\mathcal{S}\rvert + \lvert\mathcal{U}\rvert + 64, 400)$	$(301, 64)$
`leaky_relu`	`leaky_relu`
$(400, 300)$	$(64, 64)$
`leaky_relu`	`leaky_relu`
$(300, 1)$	$(64, 64)$

A.2. Hyperparameters

This section details the hyperparameters across different tasks. Two hyperparameters α and β need to tune across tasks. We list hyperparameters α and β which are various across tasks in Table 3 and Table 4.

Value-Aware MI Maximization in Eq. 6. The results in Fig. 8 show that RACE is sensitive to β which affects the quality of the shared representations. When the β is large, the representations contain more global information than task-related information. Since the importance of global and task-related information is different for different tasks, it needs to be adjusted appropriately.

Table 3. Details of the hyperparameter α and β of RACE (MATD3) in Multi-Agent MuJoCo.

Env name	α	β
2-Agent HalfCheetach	0.7	1e-1
6-Agent HalfCheetach	0.5	1e-2
2-Agent Ant	0.7	1e-2
2-Agent Ant-v2	0.2	1e-2
4-Agent Ant	0.2	1e-3
3-Agent Hopper	1.0	1e-1
2-Agent Humanoid	0.7	1e-2
2-Agent HumanoidStandup	1.0	1e-3

Table 4. Details of the hyperparameter α and β of RACE (FACMAC) in SMAC.

Env name	α	β
so_many_baneling	0.1	1e-3
2s3z	0.05	1e-3
2c_vs_64zg	0.05	1e-2
2s_vs_1sc	0.05	1e-2
MMM	0.05	1e-1
MMM2	0.01	1e-2
3s5z	0.005	1e-3
3s_vs_3z	0.005	1e-3