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

Anonymous Author(s)
Affiliation
Address
email

Abstract

In fully cooperative tasks, multi-agent credit assignment makes the value function approximation difficult, resulting in learning collaboration challenging in Multi-Agent Reinforcement learning (MARL). In contrast, Evolutionary Algorithm (EA), without requiring value function, has been demonstrated to achieve competitive performance with RL and further improve RL in single-agent settings. To develop the potential of EA to further improve MARL, we propose a novel learning framework called MARL with **R**epresentation **A**symmetry and **C**ollaboration **E**volution (RACE). Besides the MARL team, RACE maintains an additional population of collaborative teams. RACE decomposes the policies controlling the same member in different teams into the nonlinear *shared* observation representations and *individual* linear policy representations, i.e., Representation Asymmetry. The shared observation representations convey useful knowledge to control specific members learned by all teams of the population collectively. Based on the shared representations, each team can be considered as a composition of different policy representations instead of different nonlinear policy networks, which constructs a favorable space for collaboration. To achieve effective collaboration, RACE evolves the population through evolutionary algorithm and provides diverse samples to the MARL team. The MARL team trains based on the diverse samples and injects the optimized team into the population to participate in the evolution. Besides, we design the novel *agent-level* crossover and mutation operations that can be performed to promote team exploration and individual exploration. The experiments in complex continuous control tasks Multi-Agent MuJoCo and discrete micromanipulation control tasks SMAC show that RACE can significantly improve the MARL algorithms. To our knowledge, RACE has demonstrated for the first time that EA can assist MARL in achieving better collaboration in complex collaborative tasks.

1 Introduction

Multi-Agent Reinforcement learning (MARL) shows the potential to solve complex real-world problems and has been applied in many practical domains such as Robot Control [6], Game AI [21], Transportation [9] and etc. In MARL, the agents interact with the environment and other agents to collect samples. With function approximation like deep neural networks, the agents can be optimized with gradient updates. However, agents often receive a team reward for all the agents, which makes it difficult to approximate a value function to determine the contribution of each agent to the overall team [17]. Moreover, the MARL algorithms suffer from the problem of non-stationarity [15], since the agents learn concurrently and continuously affect other agents, which stems from breaking the Markov assumption that governs most single-agent RL algorithms. One way to deal with the non-stationary problem is to train all agents in a centralized fashion like single-agent RL. However, this paradigm is not scalable as the number of agents increases [10]. To balance the challenges

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

imposed by non-stationarity and scalability, the Centralized Training with Decentralized Execution (CTDE) [10] paradigm is proposed. During centralized training, agents are granted access to other agents' information and possibly the global state, while during decentralized execution, agents make decisions independently based on their individual policies. However, The problem of non-stationarity still remains. The above two issues make MARL learning collaboration challenging.

Evolutionary Algorithm (EA) [3, 5] is a bionic algorithm that simulates the natural law of genetic evolution. EA is a class of heuristic algorithms that do not rely on gradient information for policy search and optimization. which has been demonstrated to be competitive with RL in single-agent settings. EA maintains a population of individuals and searches for favorable solutions by iteration. In each iteration, three operations need to be performed: evaluation operation, selection operation, and genetic operation. Specifically, each individual needs to interact with the environment to get its fitness according to the evaluation metrics. Subsequently, we selected individuals as parents by fitness based on selection mechanisms (e.g., Select individuals with the highest fitness). Finally, parents generate the next generation through inheritance and mutation. Different from MARL, EA is heuristic and offers several strengths: 1) EA does not require an approximation function but directly inherits and varies individuals of the population according to fitness. 2) EA does not require formalizing the problem as Markov decision process (MDP) and thus does not suffer from the non-stationarity problem [11]. 3) EA has strong exploration ability, robustness and stable convergence [8]. Despite the advantages, one major drawback of EA is the low sample efficiency in evaluating each individual of the population. This issue becomes more acute When solving high-dimensional complex tasks [8].

Although there are many efforts to combine EA with RL under single-agent settings [8, 7, 4, 22], the potential of EA in MARL settings has not been fully exploited. In this paper, we propose a novel framework called MARL with **R**epresentation **A**symmetry and **C**ollaborative **E**volution (RACE). Specifically, RACE introduces an additional population of teams besides the MARL team. However, maintaining and optimizing independent nonlinear policy networks for each team is very inefficient, neglecting to share the useful knowledge learned across teams. To solve the problem, we decompose the policies controlling the same member in different teams into a nonlinear *shared* observation representation and independent *linear* policy representations. We refer to the different representation scopes (shared/individual + observation/policy representation) of the policy construction as representation asymmetry. The observation representations responsible for sharing the useful knowledge of controlling different members across teams are optimized towards an integrated update direction derived from value function maximization regarding all the EA teams and the MARL team collectively. Building on the foundation of abundant shared knowledge, each team can be considered as a collection of policy representations and searches for superior collaboration in the linear policy representation space rather than in the nonlinear parameter space as the convention. To facilitate collaboration through evolution, the EA teams with superior collaboration (i.e., high fitness) are selected to produce new teams, which explore better collaboration. Besides, the EA teams provide diverse samples generated during the evaluation phase for the MARL team, which optimizes based on the samples and injects the optimized team policy into the population periodically. To achieve effective evolution, we propose the novel agent-level crossover and mutation. The agent-level crossover only exchanges the corresponding policy representations in the two selected teams, which explores better team composition. The agent-level mutation perturbs the policy representation for the specific agent in the team, which facilitates individual exploration for discovering better collaboration. Importantly, RACE can be easily combined with most policy-based MARL algorithms. Our experiments show that RACE significantly accelerates the MARL algorithms, outperforming other baseline algorithms in continuous complex control tasks (Multi-Agent MuJoCo) and discrete micromanagement tasks (SMAC). We summarize our major contributions below:

- We propose a novel framework RACE to further exploit the potential of EA in MARL. To effective sharing knowledge across teams, the policies controlling the same member are composed of the nonlinear shared observation representations and linear individual policy representations, i.e., Representation Asymmetry, making each team formed by a collection of policy representations and effectively search for collaboration in linear policy representation space.
- To achieve collaboration by evolution, the EA teams with superior collaboration (i.e., high fitness) are retained and selected as parents to produce new teams, which can explore better collaboration. For effective evolution, we devise agent-level crossover and mutation: the agent-level crossover exchanges the policy in different teams, which can search for better team composition; the agent-level mutation adds perturbations for individual policies in the team to facilitate individual exploration.

- We empirically show that RACE significantly improves MARL algorithms and consistently outperforms related methods on both continuous control tasks MAMUJOCO and discrete control StarCraft II micromanagement environments.

2 Background

2.1 Preliminaries

We consider a fully cooperative multi-agent task where a team of agents are situated in a stochastic, partially observable environment, it can be modeled as a *decentralised partially observable Markov decision process* (Dec-POMDP) [14], which can be 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 Dec-POMDP, the full state of the environment $s_t \in \mathcal{S}$ cannot be observed by agents at each time step t. Each agent $i \in \mathcal{N}$ can only observe its individual observation o_t^i determined by observation function $\mathcal{O}(s_t, i)$, each agent i uses a stochastic policy π_i to choose actions $u_t^i \sim \pi_i(\cdot|o_t^i) \in \mathcal{U}^i$, yielding the joint action $u_t = \{u_t^i\}_{i=1}^N \in \mathcal{U}$. After executing u_t in state s_t, the environment transits to the next state s_{t+1} according to transition function $\mathcal{T}(s_t, u_t)$ and agents receive a common reward r_t from $\mathcal{R}(s_t, u_t)$, with a discount factor $\gamma \in [0, 1)$. 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 to maximize the total expected discounted return, denoted by $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. For example, MADDPG [10]/MATD3 [1] learns a centralized value function $Q_\psi(s_t, u_t)$ to optimize the decentralized policies with Q-value maximization. To evaluate the agent's contribution to the team, QMIX [17]/FACMAC [16] maintains a factored value function $Q_{tot}(s_t, \{Q_i\}_{i=1}^N)$ for credit assignment.

In addition to the MARL approaches, we introduce some necessary knowledge about Policy-extended Value Function Approximator [20] (PeVFA) which we adopt in RACE. PeVFA can preserve the values of multiple policies with only one value function. Concretely, given some representation χ_π of policy π, a PeVFA parameterized by θ takes as input χ_π additionally, i.e., $\mathbb{Q}_\theta(s, a, \chi_\pi)$. Through the explicit policy representation χ_π, one appealing characteristic of PeVFA is the value generalization among policies (or policy space).

Evolutionary Algorithm (EA) [3, 5] is a class of population-based black-box optimization methods. where a population of policies $\mathbb{P} = \{\pi_1, \cdots, \pi_n\}$ is maintained. EA performs policy search by iteration. In each iteration, all agents interact with the environment to obtain the estimates of policy fitness $\{f(\pi_1), \cdots, f(\pi_n)\}$ where the fitness can be defined as the Monte Carlo (MC) return for e episodes $f(\pi) = \frac{1}{e}\sum_{i=1}^{e}[\sum_{t=0}^{T} r_t \mid \pi]$ or other forms. With some selection criteria (e.g., select individuals with the highest fitness), the parents are selected from the population to produce the next generation in many ways such as genetic operators [13]. Specifically, the selected parents π_i and π_j produce offspring π_i' and π_j' by performing the crossover operator, i.e., $\pi_i', \pi_j' = \texttt{Crossover}(\pi_i, \pi_j)$ or the mutation operator $\pi_i' = \texttt{Mutation}(\pi_i)$. General crossover and mutation operate on the parameters of policies. Typically, k-point crossover is randomly exchange segment-wise (network) parameters of parents while Gaussian mutation adds Gaussian noises to the parameters. With the diversity brought by abundant candidates and consistent variation, EA has strong exploration ability and is more robust to local optima compared to RL.

2.2 Related Work

Centralized Training & **Decentralized Execution** [10] (CTDE) is a popular paradigm in MARL, which has better scalability & deployability. In the training phase, the global information can be utilized for optimization. In the execution phase, the agents make decisions based on their policies. For example, MADDPG [10]/MATD3 [1] uses a centralized value function to optimize decentralized policies. VDN [19], QMIX [17] and FACMAC [16] achieve CTDE through value function factorization.

Multi-Agent Evolutionary Reinforcement Learning (MERL) tries to introduce ERL into multi-agent settings. Specifically, ERL combines EA and RL to blend their complementary strengths in single-agent settings. ERL maintains a population and an RL agent. Individuals in the population interact with the environment to provide the RL with diverse samples, and the RL agent trains on the samples and periodically injects optimized policy into the population to participate in the evolution. ERL has better exploration ability, convergence and robustness than RL algorithms, and is not affected by deceptive and delayed rewards. Based on the interaction schema of ERL, MERL

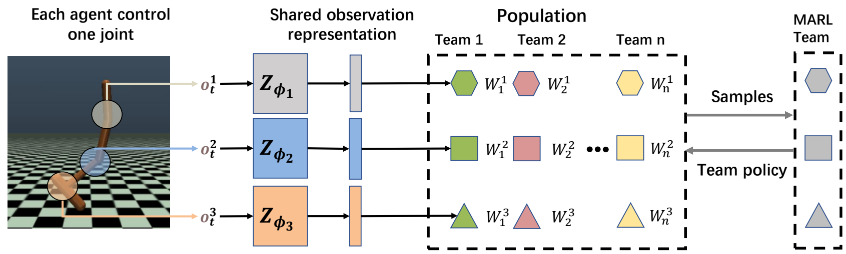

Figure 1: The conceptual illustration of Representation-based Team Construction (RTC) in 3-Agent Hopper task. All policies are composed of the nonlinear shared observation representation Z_{ϕ_i} and an individual linear policy representation W_j^i where j denotes the team index and i denotes the agent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

investigates how to make full use of reward signals in a task where both sparse team reward and dense agent-specific reward exist. MERL optimizes sparse team reward by evolution and optimizes dense agent-specific reward by gradient optimization. However, agent-specific rewards do not exist in many tasks, which makes it difficult to apply MERL in practice. Moreover, MERL is evaluated in some simple environments and does not validate the effectiveness in complex control tasks and micromanipulation tasks.

3 MARL with Representation Asymmetry and Collaborative Evolution

This section introduces our framework Representation Asymmetry and Collaborative Evolution to improve cooperative MARL. We start by describing the concept of Representation Asymmetry for team construction. Then we detail how to optimize the team's niche, i.e., the shared observation representations. Subsequently, we describe how to improve MARL with Collaborative Evolution. Finally, we introduce the overview of RACE framework.

3.1 Team Construction with Representation Asymmetry

Beyond the MARL team, RACE introduces an additional population of teams. The teams of the population and the MARL team explore superior collaboration through evolution and reinforcement, respectively. Typically, the teams in the population maintain individual policy networks for decision-making and optimization. However, independent team policy construction can prevent common knowledge from being shared across teams, resulting in inefficient learning. In the literature on Evolutionary Reinforcement learning, ERL-Re^2 [2] decomposes the policy into the shared state representation and policy representations in single-agent settings. The shared state representation can convey common knowledge in populations. Taking this inspiration, we introduce the Representation-based Team Construction (RTC) to enable efficient knowledge sharing and thus avoid duplication of learning. The illustration of RTC is shown in Fig. 1. Specifically, the policies that control the same member in different teams are composed of a *shared* nonlinear observation representation $z_t^i = Z_{\phi_i}(o_t^i) \in \mathbb{R}^d$ (given a observation o_t^i) and an *individual* linear policy representation $W_j^i \in \mathbb{R}^{(d+1)\times|\mathcal{U}^i|}$, where i donates agent index in one team and j denates the team index. The i-th agents in all teams share the same observation representations network $Z_{\phi_i}(o_t^i)$ and make decisions by combining the shared observation representation and the policy representation:

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

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, the team j is 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 representations to provide rich task-related and collaborative knowledge, which are favorable for all policies controlling the same member and not specific to any single policy. The shared observation representations determine the policy space for the member i denoted by $\Pi(\phi_i)$, where we conduct evolution and reinforcement. Formally, we summarize the team construction in

RACE below:

$$
\begin{aligned}
\textbf{Policy } i \textbf{ in team } j\textbf{:}\ & \pi_j^i(o_t^i) = \mathtt{act}(Z_{\phi_i}(o_t^i)^\mathsf{T} W_{j,[1:d]}^i + W_{j,[d+1]}^i) \\
\textbf{Construction of team } j\textbf{:}\ & W_j = \{W_j^1, W_j^2, \cdots, W_j^N\} \\
\textbf{Team policy of team } j\textbf{:}\ & \pi_j(s_t) = \{\pi_j^1(o_t^1), \pi_j^2(o_t^2), \cdots, \pi_j^N(o_t^N)\} \\
\textbf{Team Population:}\ & \mathbb{P} = \{W_1, W_2, \cdots, W_n\}
\end{aligned} \tag{1}
$$

3.2 Construct Shared Favorable Niches for All Teams

To effectively convey useful knowledge across teams, we learn the shared observation representations with *value function maximization* regarding the corresponding policies in all teams. Specifically, we learn a centralized PeVFA $\mathbb{Q}_\theta(s, u, W_j)$ based on the team policy representations W_j sampled from the population $\mathbb{P}$; For the MARL team, we maintain a centralized critic $Q_\psi(s, u)$ as convention. The optimization of the shared observation representations are illustrated in Fig. 2. The loss functions of $\mathbb{Q}_\theta$ and Q_ψ are formulated below:

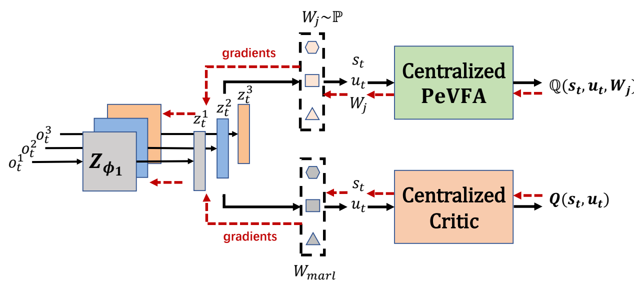

Figure 2: Optimization illustration of the shared observation representations.

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

where D is the experience buffer collected by all teams, θ', ψ' denote the target networks of the PeVFA and the MARL critic, π'_{marl} denote the target actors.

For each team in the population and MARL team, an individual update direction of the shared observation representation to control a specific member Z_{ϕ_i} is now ready to obtain by $\nabla_{\phi_i}\mathbb{Q}_\theta(s, \pi_j(s), W_j)$ 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 the corresponding agents. Z_{ϕ_i} should not take either individual update direction solely; instead, the natural way is to take an integrated update direction regarding all the agents. Finally, the loss function of Z_{ϕ_i} is defined:

$$
\mathcal{L}_Z(\phi_i) = -\mathbb{E}_{s\sim\mathcal{D}, W_j\sim\mathbb{P}}\Big[Q_\psi\left(s, \pi_{marl}\left(s\right)\right) + \mathbb{Q}_\theta\left(s, \pi_j\left(s\right), W_j\right)\Big], \tag{3}
$$

By minimizing Eq. 3, the shared observation representation Z_{ϕ_i} is optimized towards a superior policy space Π_{ϕ_i} pertaining to all policies that control the same member iteratively.

3.3 Improve MARL with Collaborative Evolution

Based on the shared observation representations Z_{ϕ_i}, all agents controlling the same member in different teams optimize their policy representations in the policy space Π_{ϕ_i}. The evolution and reinforcement occur in the linear policy representation space, which leads to more efficient optimization. We detail how to improve MARL with collaborative evolution.

For collaboration, RACE evolve the population with n teams with three phases: evaluation, selection and variation. 1): For evaluation, the teams interact with the environment for one episode to get the cumulative rewards as the fitness. 2): For selection, all teams will be divided into three categories: elite, winners, and discarders. The elite is the best performing team that participates in the construction of the next generation as a parent throughout. The elite will be completely preserved and will not participate

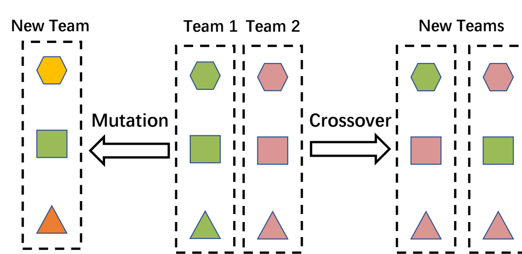

Figure 3: Agent-level genetic operators in RACE for team exploration and individual exploration.

in the subsequent mutation. The winners are selected among the remaining teams through a tournament mechanism for the crossover with the elite to produce offspring. Specifically, the winners are the best-performing team of a random subset of the population. The winners will be involved in the subsequent mutation to increase exploration. The discarders are the teams that are not selected as the elite and winners and will be replaced by new teams. 3): For variation, we design the agent-level crossover and mutation for both team and individual exploration. For individual exploration, we randomly exchange the policy representations which control the same member in the two selected teams, which helps explore better team composition. For agent exploration, we randomly add parameter perturbation to some policy representations for the selected team, which facilitates the exploration of individuals in the team. Formally, we formulate the two operations below:

$$\begin{aligned}(W_i^{new}, W_j^{new}) &= ((W_i - W_i^{d_i}) \cup W_j^{d_i}, (W_j - W_j^{d_j}) \cup W_i^{d_j}) = \texttt{Crossover}(W_i, W_j),\\ W_j^{new} &= (W_j - W_j^{d_j}) \cup P(W_j^{d_j}) = \texttt{Mutation}(W_j),\end{aligned} \tag{4}$$

where W_i and W_j are two selected teams, d_i, d_j are the randomly sampled subsets of agents indices (from 1 to N), P is the perturbation function which adds Gaussian noise to (or reset) some parameters. We use W^d to denote the subset of the policy representations of the team with indices d. RACE discards poor teams in the population and rebuilds the entire population based on the elite team. The mutated population contains three categories: the elite team in the previous generation, the reunited elite teams and the mutated elite teams where some teammates are mutated. Based on our proposed agent-level operators, the population can achieve more effective and stable evolution, as well as more informative in the sense of teams' and individuals' semantics.

Through our evolutionary operators, populations can fully explore the policy space to form collaboration policies that can be directly used for deployment. Besides, the samples generated during population exploration can be provided to the MARL team for training. As to the learning of the MARL agent W_{marl}, it resembles the conventional circumstance except done with respect to the linear policy representation and the shared replay buffer D. Taking MADDPG [10] for a typical example, the loss function of W_{marl} is defined below, based on the centralized critic Q_ψ (learned by Eq. 2):

$$\mathcal{L}_{\text{MARL}}(W_{marl}) = -\mathbb{E}_{s\sim\mathcal{D}}\Big[Q_{\psi_i}\left(s, \pi_{marl}(s)\right)\Big]. \tag{5}$$

The MARL team learn from the off-policy experience in the buffer D collected also by both the MARL team and the EA teams Meanwhile, the population incorporates the MARL policy representation W_{marl} at the end of each iteration. By such an interaction, the MARL team policy can participate in the evolution of the population, which in turn assists the population in finding collaborative policies.

3.4 The Algorithm Framework of RACE

In principle, RACE is a general framework that can be implemented with different policy based algorithms. In this paper, we use MATD3 [1] and FACMAC [16] as the basic MARL algorithms. A general pseudo-code of RACE is shown in Algorithm 1. In each iteration, the algorithm proceeds across three phases (denoted by blue). First, each teams of the population and the MARL team interact with the environment and collect the experiences. The teams in the population $\mathbb{P}$ obtain the cumulative rewards of one episode as the fitness for evolution (Line 4-6). Next, evolution and reinforcement occur in the linear policy space offered by the current shared observation representations $\{Z_{\phi_1}, \cdots, Z_{\phi_N}\}$. The teams in the population $\mathbb{P}$ are optimized with the genetic operators (Line 9-16). The MARL team learns with additional off-policy experiences collected by the teams in $\mathbb{P}$ and periodically injects policies to $\mathbb{P}$ (Line 17-18). Finally, the shared observation representations are updated to provide superior policy space for the following iteration (Line 19-20).

4 Experiments

This section empirically evaluates RACE to answer the following research questions:

RQ1 (Performance) Can RACE improve MARL and outperform other baselines in complex multi-agent cooperative tasks?
RQ2 (Superior of Components) Does the shared observation representation optimized by the centralized PeVFA and Critic better than only using PeVFA/Critic? Are the agent-level crossover and mutation effective?
RQ3 (Parameter Analysis) How much is RACE affected by the hyperparameter α?

Algorithm 1: Representation-based Collaborative Evolution (RACE)

Initialize: a replay buffer $\mathcal{D}$, the shared observation representation function $Z_{\phi_1}, \cdots, , Z_{\phi_N}$, the MARL team W_{marl}, the population $\mathbb{P} = \{W_1, \cdots, W_n\}$ with the population size n, the MARL centralized critic Q_ψ and the centralized PeVFA $\mathbb{Q}_\theta$ (target networks are omitted here)

repeat

  # Rollout both the teams in the population $\mathcal{P}$ and MARL team with $\{Z_{\phi_1}, \cdots, Z_{\phi_N}\}$ and obtain the fitness

  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 $f(W_i) = \sum_{t=0}^{T}[r_t \mid W_i]$.

  Rollout the MARL team for one episode

  Store the experiences generated by $\mathbb{P}$ and W_{marl} to $\mathcal{D}$

  # Evolution and reinforcement in the linear policy space

  Train PeVFA $\mathbb{Q}_\theta$ and MARL critic Q_ψ with D ▷ see Eq. 2

  Optimize the population: perform the genetic operators (i.e., selection, crossover and mutation).

  Based on fitness, the population $\mathbb{P}$ is divided into the `elite`, `winners`, and `discarders`.

  while *`discarders` is not completely replaced* **do**

    Randomly select a team from `winners` to crossover with the `elite`. Randomly swap teammates in the teams to get a new composition team which replaces the team in discarders

  for *Team W_j in `winners` and `discarders`* **do**

    for *Team member W_j^i in the team W_j* **do**

      if *random number* < mut_{prob} **then**

        Add minor (90%), drastic (5%) Gaussian perturbations, or reset parameters (5%) to randomly selected α parameters from W_j^i

  Optimize the MARL agent: update W_{marl} (by e.g., MADDPG, MATD3) according to Q_ψ ▷ see Eq. 5

  Inject MARL team policy representations to the population $\mathbb{P}$ periodically

  # Improving the policy space through optimizing $\{Z_{\phi_1}, \cdots, Z_{\phi_N}\}$

  Update the shared observation representation: optimize Z_{ϕ_i} with an integrated gradient direction derived from value function maximization regarding $\mathbb{Q}_\theta$ and Q_ψ ▷ see Eq. 3

until *reaching maximum training steps*;

4.1 Experimental Setups

For a comprehensive comparative study, we evaluate RACE in tasks with both continuous and discrete action spaces. For continuous tasks, we integrate RACE with MATD3 [1] and evaluate the Multi-Agent MuJoCo benchmark [16] on eight cooperative continuous control tasks, where each agent can only observe its own joints' information. For discrete tasks, we integrate RACE with FACMAC and evaluate it in the StarCraft II micromanagement environments [18] (SMAC) which has high complexity of control and requires learning policies in a large discrete action space. We compare RACE with the following baselines: MATD3 [1], FACMAC [16], MERL [12] and EA [13]. We use the official implementation for these methods and implement our method RACE based on the codebase of FACMAC and MATD3 with all settings following the original paper. Note that only one team rewards are available for these tasks, and MERL cannot be applied to these tasks directly. Thus we make MERL optimize team reward through EA and MARL collectively. All statistics are obtained based on 5 independent runs. We report the average with 95% confidence regions. For the hyperparameters specific to RACE, we set the population size to 5 in all tasks and select α from $[0.2, 0.5, 1.0]$ for Multi-Agent MuJoCo and α from $[0.01, 0.05, 0.2]$ for SMAC. All implementation details are provided in Appendix A.

4.2 Performance

We first evaluate RACE (MATD3) and other baselines in Multi-Agent MuJoCo. In these tasks, agents need to cooperate in robot control and different agents control different joints. In our experimental setting, agents can not get and observe other agents' information and global information of the robot, which is the most difficult setting in Multi-Agent MuJoCo. The results in Fig.4 show that RACE significantly improves MATD3 and outperforms other baselines in most tasks, which demonstrates the superior of RACE in challenging continuous control tasks.

To further verify the generality of the method, we integrate RACE with FACMAC and evaluate it in SMAC. The results in Fig.5 show that RACE can further improve FACMAC and outperform other baselines, reaching convergence faster and achieving higher performance. Overall, the experiments

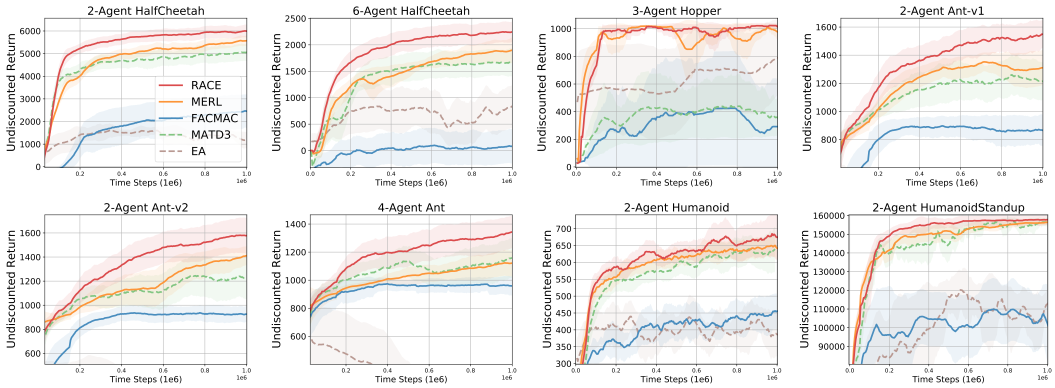

Figure 4: Performance comparison between RACE (MATD3) and baselines in Multi-Agent MuJoCo.

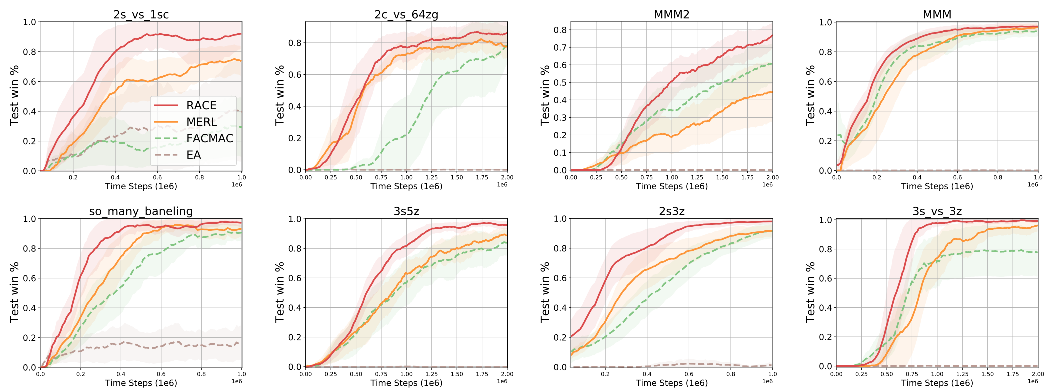

Figure 5: Performance comparison between RACE (FACMAC) and baselines in SMAC.

show that RACE is an effective and general framework that can be integrated with multiple MARL algorithms and provide significant improvement in both challenging continuous and discrete tasks.

4.3 Superior of Components

To answer RQ2, we first conduct analytical experiments about whether our proposed optimization approach for the shared observation representations is efficient. We considered three optimization ways: optimize with PeVFA and Critic (ours), optimize with Critic and optimize with PeVFA. The results in Fig. 6 show that optimizing the shared observation representations with PeVFA and Critic is more effective than only using PeVFA/Critic. The reason is that only optimizing with PeVFA/Critic only builds a policy space that is superior for EA teams or the MARL team, which does not take advantage of EA and even compromises the performance of the original MARL.

To verify the superiority of the agent-level operators, we perform ablation experiments on crossover and mutation at the agent level and compare it with the normal operators, i.e., operate directly in parameter space. The results in Fig. 7 demonstrate that removing either agent-level crossover or agent-level mutation degrades performance. This illustrates the importance of team search and

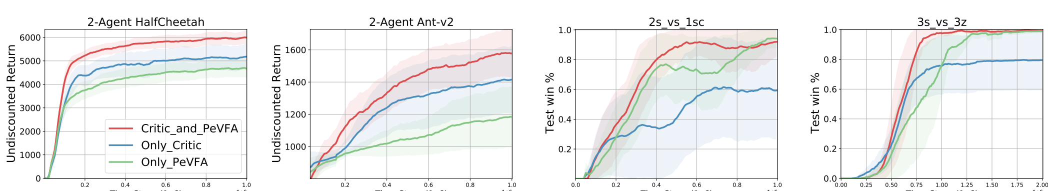

Figure 6: Experiments about how to update the shared observation representation. Only using Critic/PeVFA can not construct a superior policy space for all teams.

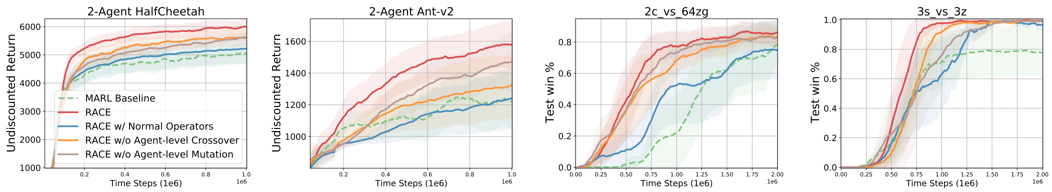

Figure 7: Ablation study on agent-level crossover and mutation operators.

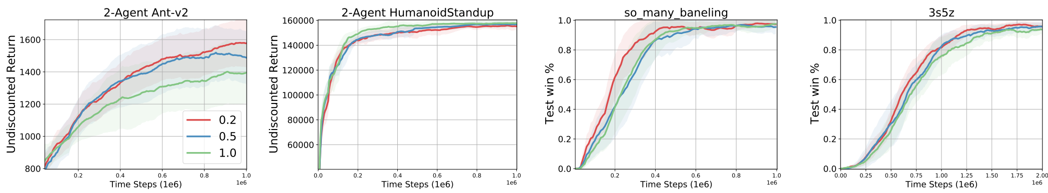

Figure 8: Analysis on hyperparameter α.

individual search, since team search can help find a better team composition, and individual search can help further promote the discovery of effective individuals. Besides, our operators are more effective than normal operators, which can deliver more performance gains.

4.4 Parameter Analysis

We analyze the only hyperparameter α which controls the degree of variation. The results in Fig. 8 show that the performance of different α is similar and proper adjustment of α can provide better results. For the continuous control tasks, i.e., Multi-Agent MuJoCo, α is chosen from $[0.2, 0.7 = 5, 1.0]$. This is mainly because the tasks are generally insensitive to perturbations of the policies, and large perturbations are useful for exploration. For the micromanipulation task, i.e., SMAC, α is chosen from $[0.01, 0.05, 0.2]$. This is mainly because these micromanagement tasks are very sensitive to small changes in the policies, which can lead to large behavioral differences, so we set a smaller value.

5 Conclusion

To fully exploit the potential of EA in MARL, we propose a novel framework RACE. In RACE, we design the representation-based team construction for effective knowledge sharing. Specifically, the policies controlling the same member in different teams are composed of shared observation representations and individual policy representations. With the shared observation representations, knowledge can be efficiently conveyed across different teams, and collaboration is more easily formed in linear policy space. Moreover, the EA teams with superior collaboration (i.e., high performance) are selected as parents to produce new teams for collaboration exploration. To achieve effective 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 cooperative environments with both continuous action space and discrete action space. To the best of our knowledge, we show for the first time that EA can further improve MARL in complex tasks, i.e., continuous control tasks Multi-Agent MuJoCo and complex discrete micromanipulation tasks SMAC.

References

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

[2] Anonymous. ERL-re^2: Efficient evolutionary reinforcement learning with shared state representation and individual policy representation. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. under review.

[3] T. Bäck and H. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1(1):1–23, 1993.

[4] C. Bodnar, B. Day, and P. Lió. Proximal distilled evolutionary reinforcement learning. In *AAAI*, pages 3283–3290, 2020.

[5] F. J. Gomez, J. Schmidhuber, and R. Miikkulainen. Efficient non-linear control through neuroevolution. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*. Springer, 2006.

[6] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *ICRA*, pages 6023–6029, 2019.

[7] S. Khadka, S. Majumdar, T. Nassar, Z. Dwiel, E. Tumer, S. Miret, Y. Liu, and K. Tumer. Collaborative evolutionary reinforcement learning. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3341–3350, 2019.

[8] S. Khadka and K. Tumer. Evolution-guided policy gradient in reinforcement learning. In *NeurIPS*, pages 1196–1208, 2018.

[9] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 2019.

[10] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6379–6390, 2017.

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

[12] S. Majumdar, S. Khadka, S. Miret, S. McAleer, and K. Tumer. Evolutionary reinforcement learning for sample-efficient multiagent coordination. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. PMLR, 2020.

[13] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, 1998.

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

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

[16] B. Peng, T. Rashid, C. S. de Witt, P. Kamienny, P. Torr, W. Boehmer, and S. Whiteson. FACMAC: factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

[17] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2018.

[18] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C. Hung, P. H. S. Torr, J. N. Foerster, and S. Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, 2019.

[19] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, 2018.

[20] H. Tang, Z. Meng, J. Hao, C. Chen, D. Graves, D. Li, W. Liu, and Y. Yang. What about taking policy as input of value function: Policy-extended value function approximator. *CoRR*, abs/2010.09536, 2020.

[21] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D Budden, Y. Sulsky, J Molloy, T. Le Paine, Ç. Gülçehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, pages 350–354, 2019.

[22] Y. Wang, T. Zhang, Y. Chang, B. Liang, X. Wang, and B. Yuan. A surrogate-assisted controller for expensive evolutionary reinforcement learning. *CoRR*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In the Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is on general Reinforcement Learning study. No specific practical application is considered.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will open the source code including the MATD3 and FACMAC versions when the paper is accepted.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

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 Implementation of Baselines

For all baseline algorithms, we use the official implementation. In our paper, there are several baselines: MATD3, MERL[1], EA[2] and FACMAC[3]. We use the official implementation for comparison. MATD3 is a simple extension of the official TD3[4] implementation in the CTDE framework. To integrate with MATD3 and FACMAC, we implemented RACE on the official code and keep the other hyperparameters and processes unchanged. We fine-tuned all baselines to provide the best performance.

A.2 Network Architecture

This section details the architecture of the networks. For the EA process, we use the implementation in ERL[5]. All the processes remain the same. For structures specific to RACE, in Multi-Agent MuJoCo, the shared observation representation networks are constructed by two fully connected layers with 400 and 300 units. The policy representation is the final layer which controls one agent's actions. In SMAC, the shared observation representation networks use the first n-1 layers of policy networks in FACMAC. The policy representation is the final layer.

In RACE (MATD3), PeVFA takes state, action and policy representation as inputs and maintains double Q networks which are similar to MATD3. The policy representation can be regarded as a combination of a matrix with shape $[300, \texttt{action\_dim}]$ (i.e., weights) and a vector with shape $[\texttt{action\_dim}]$ (i.e., biases) which can be concatenated as a matrix with shape $[300+1, \texttt{action\_dim}]$. We first encode each vector with shape $[300 + 1]$ of the policy representations with 3 fully connected layers with units 64 and `leaky_relu` activation function. Thus we can get an embedding list with shape $[64, \texttt{action\_dim}]$ and get the final policy embedding with shape $[64]$ by taking the mean value of the embedding list in the action dimension. With the policy embedding, we concatenate the policy embedding, 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 PeVFA. The activation functions in 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 framework structure in FACMAC. FACMAC maintains a shared policy network and a shared critic network, in addition to a Qmix Net for credit assignments. We introduce an extra Critic network with policy representation inputs and an extra Qmixer network with policy representation inputs. The policy representation is 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 shared observation representation for each agent. To be consistent, MARL team uses a shared observation representation network and a shared policy representation for all agents. But for each EA team, RACE maintains different policy representations for each agent.

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

Shared Observation Representation Network	Policy Representation
$(\texttt{obs\_dim}, 400)$	$(300, \texttt{action\_dim})$
tanh	tanh
(400,300)	
tanh	

[1]https://tinyurl.com/y6erclts
[2]https://github.com/ShawK91/Evolutionary-Reinforcement-Learning
[3]https://github.com/oxwhirl/facmac
[4]https://github.com/sfujim/TD3
[5]https://github.com/ShawK91/Evolutionary-Reinforcement-Learning

Table 2: The structure of PeVFA in RACE (MATD3)

PeVFA	
$(\texttt{state-action\_dim} + 64, 400)$	$(301, 64)$
`leaky_relu`	`leaky_relu`
$(400, 300)$	$(64, 64)$
`leaky_relu`	`leaky_relu`
$(300, 1)$	$(64, 64)$

A.3 Hyperparameters

This section details the hyperparameters across different tasks. Only one hyperparameter α need to tune across all tasks. Population size is 5 for both RACE (MATD3) and RACE (FACMAC). The synchronization period which controls the frequency of the RL policy injected into the population is set to 1. We list hyperparameters α which varied across tasks in Table 3 and Table 4.

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

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

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

Env name	α
2c_vs_64zg	0.2
2s_vs_1sc	0.2
MMM	0.2
MMM2	0.05
3s5z	0.05
2s3z	0.01
so_many_baneling	0.01
3s_vs_3z	0.01