Constructive Conflict-Driven Multi-Agent Reinforcement Learning for Strategic Diversity

Yuxiang Mai[1,2,3] , **Qiyue Yin**[1,2,3] , **Wancheng Ni**[1,2,3,*] , **Pei Xu**[2,3] , **Kaiqi Huang**[1,2,3,*]

[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]CRISE, Institute of Automation, Chinese Academy of Sciences
[3]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences
{maiyuxiang2020, wancheng.ni, pei.xu}@ia.ac.cn, {qyyin, kqhuang}@nlpr.ia.ac.cn

Abstract

In recent years, diversity has emerged as a useful mechanism to enhance the efficiency of multi-agent reinforcement learning (MARL). However, existing methods predominantly focus on designing policies based on individual agent characteristics, often neglecting the interplay and mutual influence among agents during policy formation. To address this gap, we propose Competitive Diversity through Constructive Conflict (CoDiCon), a novel approach that incorporates competitive incentives into cooperative scenarios to encourage policy exchange and foster strategic diversity among agents. Drawing inspiration from sociological research, which highlights the benefits of moderate competition and constructive conflict in group decision-making, we design an intrinsic reward mechanism using ranking features to introduce competitive motivations. A centralized intrinsic reward module generates and distributes varying reward values to agents, ensuring an effective balance between competition and cooperation. By optimizing the parameterized centralized reward module to maximize environmental rewards, we reformulate the constrained bilevel optimization problem to align with the original task objectives. We evaluate our algorithm against state-of-the-art methods in the SMAC and GRF environments. Experimental results demonstrate that CoDiCon achieves superior performance, with competitive intrinsic rewards effectively promoting diverse and adaptive strategies among cooperative agents.[1]

1 Introduction

Due to the advancement of deep multi-agent reinforcement learning, many real-world problems have been modeled and addressed as cooperative multi-agent problems [Samvelyan *et al.*, 2019; Sunehag *et al.*, 2017; Rashid *et al.*, 2020], such as traffic signal control [Wiering, 2000], autonomous driving [Hu *et al.*, 2019], and robot control [Haarnoja *et al.*,

2019]. The goal of these cooperative multi-agent problems is to maximize the reward from a team perspective [Colby *et al.*, 2015; Rashid *et al.*, 2020]. However, learning effective strategies for such complex multi-agent systems remains a significant challenge. One key problem is that relying on a single feedback signal often leads to homogeneous agent policies, resulting in inefficient exploration and hindering the emergence of complex cooperative behaviors [Li *et al.*, 2021].

Providing additional reward signals [Du *et al.*, 2019; Li *et al.*, 2021; Jiang and Lu, 2021] to individual agents has proven to be an effective approach for diversifying cooperative strategies and improving performance. Existing methods for designing such rewards primarily rely on the mutual information between agent policies and agent identities (IDs) as intrinsic rewards [Li *et al.*, 2021; Jiang and Lu, 2021], but such methods do not consider the influence of other agents and lack intuitive interpretability. LIIR [Du *et al.*, 2019] introduces learnable intrinsic rewards with shared parameters to facilitate information exchange, enabling agents to develop independent strategies. However, the process of learning intrinsic rewards overlooks the important role of competitive mechanisms in cooperative scenarios, as evidenced in sociological research.

There is a unique phenomenon observed in sociological research called Constructive Conflict [Kirchmeyer and Cohen, 1992; King *et al.*, 2009]. Constructive Conflict describes a situation in which the collective intelligence of a cooperative group is stimulated by the positive competition or clash of views between individuals and subgroups. Unlike destructive conflict that hinder the achievement of optimization goals, constructive conflict does not lead to strained relationships or reduced efficiency. Instead, it fosters diverse perspectives and strategy refinement, positively contributing to group goals. Inspired by the concept of constructive conflict, we propose competitive intrinsic rewards to enhance agent learning and team performance. In particular, unlike intrinsic rewards, the competitive intrinsic rewards provide cooperative agents with competitive incentives, thereby stimulating strategic communication and strategy diversification among agents.

In this paper, we propose an algorithm called CoDiCon, which is designed for competitive intrinsic rewards. Specifically, ranking is an effective method to encourage mutual competition, so we design an intrinsic reward that incorpo-

---

*Corresponding authors
[1]Codes are available at https://github.com/YuxiangMai/CoDiCon.

rates ranking property. The overall algorithm adopts an actor-critic structure [Lowe *et al.*, 2017; Schulman *et al.*, 2017] based on MAPPO [Yu *et al.*, 2022], where each agent has a separate set of parameters. The global critic evaluates the current action values, and a centralized intrinsic reward generation module produces a ranked intrinsic reward for each agent's current action to foster competition. The agent's policy is jointly optimized using both extrinsic and intrinsic rewards, with the intrinsic reward serving merely as a training signal to distinguish rankings rather than having any inherent meaning. Mathematically, the optimization problem can be modeled as a constrained bilevel optimization problem, where the constraints in the outer optimization ensure the equivalence between the optimization objective and the original maximization of the environmental reward. We summarize our contributions as follows:

- We design an effective intrinsic reward mechanism based on the principles of constructive conflict, introducing competitive incentives to cooperative multi-agent systems to enhance team performance.

- The ranking module is proposed to provide agents with intrinsic rewards possessing ranking property. The optimization objective is formulated as a constrained bilevel problem, ensuring that optimizing the policy based on the ranking rewards aligns with the original goal of maximizing environmental rewards.

- Experimental results demonstrate that our algorithm outperforms existing methods. Visualizations of the agents' intrinsic rewards and state-reward space indicate that the learned intrinsic rewards produce distinct signals, enabling agents to take diverse actions collaboratively.

2 Related Work

In recent years, deep multi-agent reinforcement learning has made significant advancements. Research efforts such as COMA [Foerster *et al.*, 2018], MADDPG [Lowe *et al.*, 2017], and LICA [Zhou *et al.*, 2020] have explored policy-based approaches to multi-agent problems, utilizing a centralized critic to evaluate the value of each distributed policy. Value decomposition methods, including VDN [Sunehag *et al.*, 2017], QMIX [Rashid *et al.*, 2020], and QTRAN [Hostallero *et al.*, 2019], decompose environmental feedback into individual agent value functions, thereby enabling credit assignment. QPLEX [Wang *et al.*, 2020] proposed using dueling networks to relax the monotonicity constraint of mixing networks, expanding their representational capacity. Learning from others is an innate human survival skill, a phenomenon mirrored in agent policies, where information exchange via mixing networks enhances policy generation. However, most existing methods focus on centralized mixing network structures to satisfy value function constraints, often relying on strong assumptions about these functions. In contrast, our approach learns explicit intrinsic rewards for each agent at each time step, avoiding such assumptions and enabling immediate credit assignment.

Our work addresses the problem of designing intrinsic rewards in cooperative multi-agent systems, a topic that has been explored in some previous studies. EOI [Jiang and Lu, 2021] proposed learning a classifier for observations to compute the probability that an observation belongs to each agent, using this probability as an intrinsic reward to adjust the agent's final reward. CDS [Li *et al.*, 2021] introduced maximizing the mutual information between agent IDs and trajectories as an intrinsic reward to promote diverse policies. However, such methods do not consider the influence of other agents, overlooking the importance of learning from them, and they lack intuitive interpretability. LIIR [Du *et al.*, 2019] designed a module for generating intrinsic rewards to guide more diverse policies, ensuring that the two-level optimization is equivalent to the initial optimization problem. However, it also overlooks the positive competitive influence that other agents can have on policy development. Furthermore, directly calculating intrinsic rewards can result in an excessively large optimization space, increasing the risk of policies converging to local optima.

3 Preliminary

3.1 Cooperative Multi-Agent Reinforcement Learning

Multi-agent reinforcement learning [Sutton and Barto, 2018] extends traditional reinforcement learning to address decision-making problems with multiple agents in sequential environments. A fully cooperative multi-agent problem can be represented as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [Oliehoek and Amato, 2016]. In this model, a Dec-POMDP is described by the tuple $\langle S, A, U, Z, O, P, r, n, \gamma \rangle$. Here, S represents the global states of the environment, and A denotes the set of n agents. At each time step t, each agent $i \in A$ selects an action u_i from its action set U_i, resulting in a joint action $\boldsymbol{u} \in \boldsymbol{U} \equiv U^n$. The state transition function P determines the next state $s' \in S$ based on the current state s and the joint action $\boldsymbol{u}$. All agents share a common reward $r(s, \boldsymbol{u})$, which depends on the state and joint action. A discount factor $\gamma \in [0, 1)$ is used to weigh future rewards. In the partially observable setting, each agent receives a local observation $z_i \in Z$, derived from the observation function $O(s, \boldsymbol{u}) : S \times \boldsymbol{U} \rightarrow Z$, where Z is the observation space. Each agent learns an individual policy $\pi_i(u_i | \tau_i; \theta_i)$ with parameters θ_i based on its own action-observation history τ_i. The agents' combined policies determine the joint action taken in the environment.

3.2 Centralized Training with Decentralized Execution

Centralized Training and Decentralized Execution (CTDE) is a commonly used paradigm for addressing multi-agent problems. In this paradigm, actor-critic methods are often chosen [Williams, 1992; Andrychowicz *et al.*, 2016; Schulman, 2015; Sutton and Barto, 2018; Yu *et al.*, 2022]. In our approach, we choose the MAPPO [Yu *et al.*, 2022] algorithm as the base framework, which consists of n independently parameterized policies π_{θ_i}. The policy parameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_n\}$ are optimized using policy gradient methods to maximize the extrinsic reward:

$$J^{\text{ex}}(\boldsymbol{\theta}) = \mathbb{E}_{s,\boldsymbol{u}} \left[\min \left(p(\boldsymbol{\theta}) A_\pi (s, \boldsymbol{u}), \widetilde{p}(\boldsymbol{\theta}) A_\pi (s, \boldsymbol{u}) \right) \right], \quad (1)$$
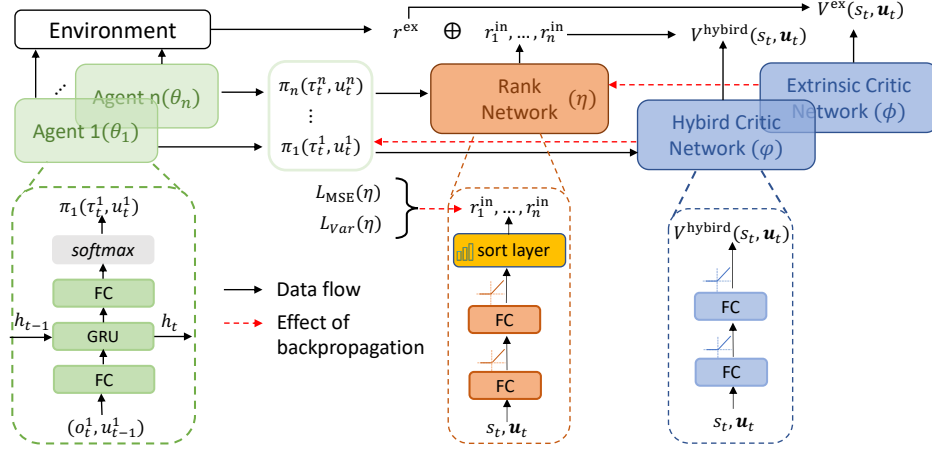
Figure 1: Framework of the proposed method. The framework consists of four parameterized modules: the agent module with parameters $\boldsymbol{\theta}$, the ranking module with parameters η, the hybrid critic module with parameters φ, and the extrinsic critic module with parameters ϕ.

where $p(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(u|\boldsymbol{s})}{\pi_{\text{old},\boldsymbol{\theta}}(u|\boldsymbol{s})}$ denotes the policy ratio, which depends on the policy parameters $\boldsymbol{\theta}$, and $\widetilde{p}(\boldsymbol{\theta}) = \text{clip}(p(\boldsymbol{\theta}), 1 - \epsilon, 1 + \epsilon)$ is the clipped policy ratio. $A_\pi(s, \boldsymbol{u})$ represents the advantage function based on the state and actions. There are several methods to estimate $A_\pi(s, \boldsymbol{u})$. For example, $A_\pi(s, \boldsymbol{u}) = r^{\text{ex}}(s, \boldsymbol{u}) + V^{\text{ex}}(s') - V^{\text{ex}}(s)$ is the standard advantage function [Schulman, 2015], where s' is the next state.

4 Method

In this section, we present the competitive intrinsic reward algorithm, CoDiCon. First, we provide a formal definition of the problem and represent it as a constrained bilevel optimization problem. Then, we detail the approach for solving this problem, which effectively addresses the interplay between agents' learning and their competitive interactions.

4.1 The Optimization Objective

We define the components of the agent's reward, which consist of intrinsic reward r^{in} and extrinsic reward r^{ex}. The intrinsic reward is parameterized by η. At each time step, each agent takes a state-action pair as input, the hybrid reward can be expressed as:

$$r_{i,t}^{\text{hybrid}} = r_t^{\text{ex}} + \lambda r_{i,t}^{\text{in}}(\eta). \tag{2}$$

In equation (2), λ represents the hyperparameter that balances intrinsic and extrinsic rewards. It is important to note that the additional intrinsic reward does not appear in the standard multi-agent problem. After defining a hybrid reward $r_{i,t}^{\text{hybrid}}$ for each agent at each time step, we define the discounted hybrid reward for each agent as follows:

$$R_{i,t}^{\text{hybrid}} = \sum_{l=0}^{\infty} \gamma^l \left(r_{t+l}^{\text{ex}} + \lambda r_{i,t+l}^{\text{in}}(\eta) \right), \tag{3}$$

and the hybrid value function for agent i is defined as:

$$V_i^{\text{hybrid}}(s_t) = \mathbb{E}_{s_{t+1}, u_{i,t}, \dots} \left[R_{i,t}^{\text{hybrid}} \right]. \tag{4}$$

Unlike the extrinsic value function V^{ex}, these hybrid value functions V_i^{hybrid} do not have any actual physical meaning. They are just used to update the parameters θ_i of each agent's policy. Next, we consider the overall optimization objective, defined as:

$$\max_{\eta, \boldsymbol{\theta}} \quad J^{\text{ex}}(\eta), \tag{5}$$

$$\text{s.t.} \quad \theta_i = \arg\max_{\boldsymbol{\theta}} J_i^{\text{hybrid}}(\boldsymbol{\theta}, \eta), \quad \forall i \in [1, 2, \dots, n],$$

$$r_1^{\text{in}} < r_2^{\text{in}} < \cdots < r_n^{\text{in}}, \quad \text{where} \quad r_i^{\text{in}} = \text{rank}(\eta)$$

where $J_i^{\text{hybrid}} = \mathbb{E}_{s_0, u_{i,0}, \dots} \left[R_{i,0}^{\text{hybrid}} \right]$ depending on θ_i and η, η indicates the parameter of the ranking module and $\boldsymbol{\theta}$ indicates the policy parameter set $\{\theta_1, \theta_2, \cdots, \theta_n\}$. The updates of the ranking module parameters and policy parameters are performed alternately. The ranking network module consists of fully connected layers and a sorting function layer, ensuring that the intrinsic rewards r^{in} it outputs have numerical differences and are arranged in ascending order. When the parameters of the ranking module are frozen, the policy parameter θ_i is optimized by maximizing the hybrid expected cumulative return J_i^{hybrid} for agent i. The advantage of this approach lies in using the ranked intrinsic rewards at each step for policy learning, where competition is captured by the intrinsic rewards, defined through pairwise performance ranking as $r_i^{\text{in}} = \text{rank}_\eta(i, -i)$. This mechanism motivates agents to compete and learn from each other, leading to more complex and diverse behaviors. Conflict arises when an agent's action that maximizes its intrinsic reward r_i^{in} opposes the team's extrinsic reward r^{ex}. Specifically, conflict exists if for some agent i, $\nabla_{u_i} r^{\text{ex}} \cdot \nabla_{u_i} r_i^{\text{in}} < 0$, implying the agent's optimal behavior hinders team performance. From an optimization perspective, problem (5) can be viewed as a constrained bilevel optimization problem, where the outer optimization constrains the inner policy improvement process. The ultimate goal of optimization is to maximize the environmental reward under the constraint of intrinsic rewards. This ensures alignment with the original objective of maximizing

the environmental reward, allowing better-performing policies to receive higher intrinsic rewards. In the next section, we will discuss the relationship between J^{ex} and the intrinsic reward parameter η during the optimization process.

4.2 Algorithm

For the constrained two-level optimization problem, at each update step, the policy parameters are updated based on the hybrid expected return, while the update of the ranking module parameters depends on the ranking process and the extrinsic expected return.

Specifically, the parameters of each agent are updated using the hybird critic network. Given the trajectory data generated by the agents, the policy parameters are updated through the policy gradient method as described in (1):

$$\nabla_{\theta_i} \min\left(p(\theta_i) A_i^{\text{hybird}}(s, \boldsymbol{u}), \widetilde{p}(\theta_i) A_i^{\text{hybird}}(s, \boldsymbol{u}) \right), \quad (6)$$

where $p(\theta_i)$ is the policy ratio of agent i. $A_i^{\text{hybird}}(s, \boldsymbol{u})$ denotes the hybird critic, which can be chosen in various ways [Sutton and Barto, 2018; Schulman, 2015; Yu *et al.*, 2022]. In this paper, we choose $A_i^{\text{hybird}}(s, \boldsymbol{u}) = r_i^{\text{hybird}}(s, \boldsymbol{u}) + V_{\varphi}^{\text{hybird}}(s') - V_{\varphi}^{\text{hybird}}(s)$ as the advantage function, where $V_{\varphi}^{\text{hybird}}$ represents the hybird state value, parameterized by φ, and s' denotes the next state of the agent in the trajectory. Given (6) and a policy learning rate a, the updated policy parameter θ_i' can be expressed as: $\theta_i' = \theta_i + \alpha \nabla_{\theta_i} \min\left(p(\theta_i) A_i^{\text{hybird}}(s, \boldsymbol{u}), \widetilde{p}(\theta_i) A_i^{\text{hybird}}(s, \boldsymbol{u}) \right)$.

To ensure that intrinsic rewards promote competition among agents, we update η by applying a relative numerical constraint to the intrinsic rewards produced by the ranking module. This ensures that the rewards follow a sequential distribution. Specifically, we use mean squared error (MSE) loss and variance loss of intrinsic rewards to maintain the output as a sequence with numerical differences:

$$\mathcal{L}_{\text{MSE}}(\eta) = \frac{1}{n} \sum_{i=1}^{n} (r_i^{\text{in}} - y_i)^2, \quad (7)$$

$$\mathcal{L}_{\text{Var}}(\eta) = \frac{1}{n} \sum_{i=1}^{n} (r_i^{\text{in}} - \bar{r}^{\text{in}})^2, \quad (8)$$

where y is the optimization target for the intrinsic reward, represented as an ordered sequence. This sequence is randomly initialized at the beginning of training (in our setup, 20% positive values and 80% negative values) and remains fixed during subsequent training. $\bar{r}^{\text{in}}$ is the mean of the intrinsic rewards. For the update of the parameter η, we first minimize the MSE loss and maximize the variance loss:

$$\mathcal{L}(\eta) = \beta_1 \mathcal{L}_{\text{MSE}}(\eta) - \beta_2 \mathcal{L}_{\text{Var}}(\eta). \quad (9)$$

Given (9) and a learning rate β, the updated parameters of the ranking module can be expressed as: $\eta' = \eta - \beta \nabla_{\eta} \mathcal{L}(\eta)$.

Next, we construct expressions for η' and J^{ex} to update the parameter η'. Using the updated policy parameter θ', we apply the chain update rule to get:

$$\nabla_{\eta'} J^{\text{ex}} = \nabla_{\theta_i'} J^{\text{ex}} \nabla_{\eta'} \theta_i'. \quad (10)$$

Algorithm 1 The algorithm of CoDiCon.

1: **Input:** Policy learning rate α and intrinsic reward learning rate β.
2: **Initialize:** Policy parameters $\boldsymbol{\theta}$ and intrinsic reward parameters η.
3: **for** $t = 1$ to $T_{\max}$ **do**
4:     Sample a trajectory $\mathcal{D} = \{s_0, \boldsymbol{u_0}, s_1, \boldsymbol{u_1}, \cdots\}$ by executing actions with the decentralized policies $\{\pi_{\theta_1}, \cdots, \pi_{\theta_n}\}$;
5:     Update $\boldsymbol{\theta}$ according to (6) with learning rate α;
6:     Update η according to (9) with learning rate β;
7:     Compute (11) and (12) using samples from $\mathcal{D}$ after the first update of η;
8:     Update η' according to (10) and step 7 with learning rate β;
9: **end for**
10: **return** policy parameters $\boldsymbol{\theta}$

The purpose of (10) is to formally express the impact of J^{ex} on the updated policy parameter η' through the updated parameter θ_i'. This technique is widely adopted in meta-gradient learning [Andrychowicz *et al.*, 2016; Santoro *et al.*, 2016; Xu *et al.*, 2018]. Using samples generated by the updated policy network, the meta-gradient $\nabla_{\eta'}$ can be computed. In (10), $\nabla_{\theta_i'} J^{\text{ex}}$ can be estimated by stochastic gradient as

$$\nabla_{\theta_i'} \min\left(p(\theta_i') A^{\text{ex}}(s, \boldsymbol{u}), \widetilde{p}(\theta_i') A^{\text{ex}}(s, \boldsymbol{u}) \right), \quad (11)$$

here, $A^{\text{ex}}(s, \boldsymbol{u})$ denotes the centralized extrinsic critic. Similar to the centralized hybrid critic, we define $A^{\text{ex}}(s, \boldsymbol{u}) = r^{\text{ex}}(s, \boldsymbol{u}) + V_{\phi}^{\text{ex}}(s') - V_{\phi}^{\text{ex}}(s)$, where $V_{\phi}^{\text{ex}}(s)$ is the extrinsic value function with parameters ϕ. For the update of η', the second term in (10) can be derived as:

$$\begin{aligned}\nabla_{\eta'} \theta_i' =& \nabla_{\eta'} \left[\theta_i + \alpha \nabla_{\theta_i} \min\left(p(\theta_i) A_i^{\text{hybird}}, \widetilde{p}(\theta_i) A_i^{\text{hybird}} \right) \right] \\ =& \alpha \lambda \min(\nabla_{\theta_i} p(\theta_i) \nabla_{\eta'} r_i^{\text{hybird}}, \nabla_{\theta_i} \widetilde{p}(\theta_i) \nabla_{\eta'} r_i^{\text{hybird}}). \end{aligned}$$
$$(12)$$

Figure 1 presents the overall framework of the CoDiCon algorithm. A detailed description of the algorithm is provided in Algorithm 1.

5 Experiments

In this section, we analyze and illustrate the performance and effectiveness of our algorithms in Pac-Men, Google Reasearch Football (GRF) and Starcraft Multi-agent Changellenge (SMAC) environments. We select the multi-agent policy-based method (MAPPO), the intrinsic reward policy-based method (LIIR), and the independent emergence methods (EOI, CDS) as baseline methods for comparison. First, we analyze the effectiveness of introducing a competitive mechanism to improve policy performance in the Pac-Man environment. Next, we conduct performance comparison experiments on GRF and SMAC. Then, we perform a case study on GRF to analyze the effectiveness of the intrinsic rewards in our algorithm. Additionally, we use t-SNE visualization in the SMAC environment to compare the final
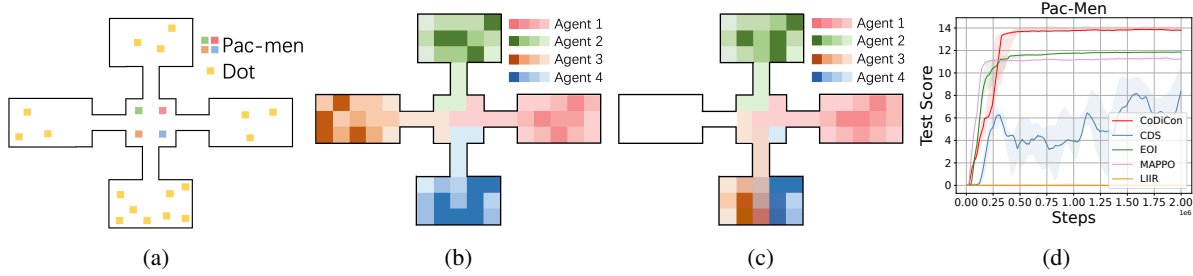
Figure 2: (a) The designed Pac-Men environment. (b) Visitation heatmap of EOI. (c) Visitation heatmap of our CoDiCon. The darker color means the higher value. (d) The training curve for the compared methods.
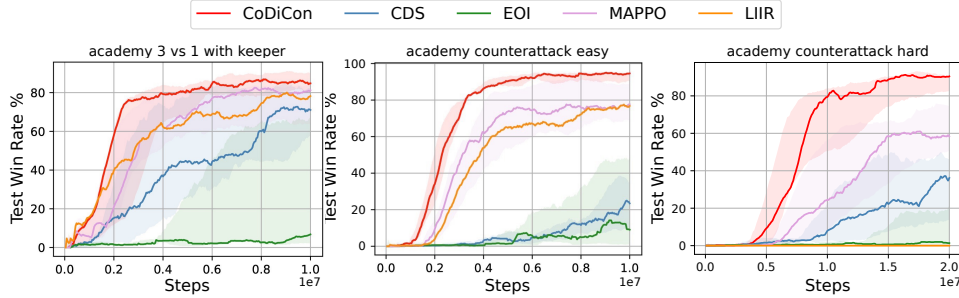

Figure 3: Training curves compared with the baselines on GRF.

distribution of policies in the state-reward space, explaining the intrinsic reasons behind the effectiveness of the algorithm. Finally, we experimentally validate the two modules of the algorithm that encourage intrinsic reward variability through ablation studies.

5.1 Competitive Intrinsic Reward

We design the Pac-Men environment, shown in Figure 2a, to demonstrate how our algorithm encourages agents to compete, thereby improving overall algorithm performance. In this environment, the central room connects four equally sized rooms and is equidistant from each of them. Four agents are initialized at fixed positions near the entrance of each room, and each agent has a 5x5 field of view. To foster competition among agents, the lower room contains the highest number of "dots" compared to the other three rooms. To make the environment more challenging, the game is limited to 17 timesteps, making it difficult for a single agent to collect all dots in the lower room, requiring at least two agents. The best case scenario is that one of the agents gives up the closest room and chooses to enter the room below, and then the two agents eat all the dots in the room below together. Agents incur a -0.25 penalty per timestep but gain 1 reward for each bean eaten. We compare our algorithm with SOTA algorithms. As shown in the visitation heatmap in Figure 2b and 2c, the EOI agents dispersed to four different rooms, whereas two agents in our algorithm both moved to the lower room, which contains the highest number of dots. In this environment, overly dispersed strategies do not result in optimal rewards. The experimental results shown in Figure 2 demonstrate that our algorithm successfully discovers the optimal strategy, while other algorithms fall into local optimal

solutions. The possible reason is that our algorithm enables other agents to learn the high-reward strategy of the agent that moves to the lower room to collect "dots" by continuously ranking the intrinsic rewards, that is, by assigning higher intrinsic rewards to the agents that go to the lower room.

5.2 Performance on GRF

In this section, we first evaluate the performance of the algorithm in the GRF environment. Specifically, we compare our algorithm with others on GRF tasks of increasing difficulty, including academy_3_vs_1_with_keeper, academy_counterattack_easy, and academy_counterattack_hard. In the GRF task, the agents need to collaborate in time and space to organize the attacking opportunities, and only the scoring are rewarded. In our experiments, we control the agents on the left (in yellow) except the goalkeeper's agent, and the agents on the right are controlled via the roule-based AI built into the game engine. The agents have 19 discrete action spaces, including moving from different angles, sliding, shooting, and passing. Observations include the position and direction of movement of the agents and other agents, as well as the position and direction of movement of the ball. The movement of the soccer ball in the z-axis direction is also included in the observations, and the agents receive feedback from the environment only at the end of the game.

Our algorithm outperforms all other algorithms in all test environments, and the results are shown in Figure 3. In the academy_3_vs_1_with_keeper, other baselines also perform well, such as MAPPO and LIIR, showing performance close to that of our algorithm. However, on the progressively more difficult environments of academy_counterattack_easy and academy_counterattack_hard, the advantage of our algo-
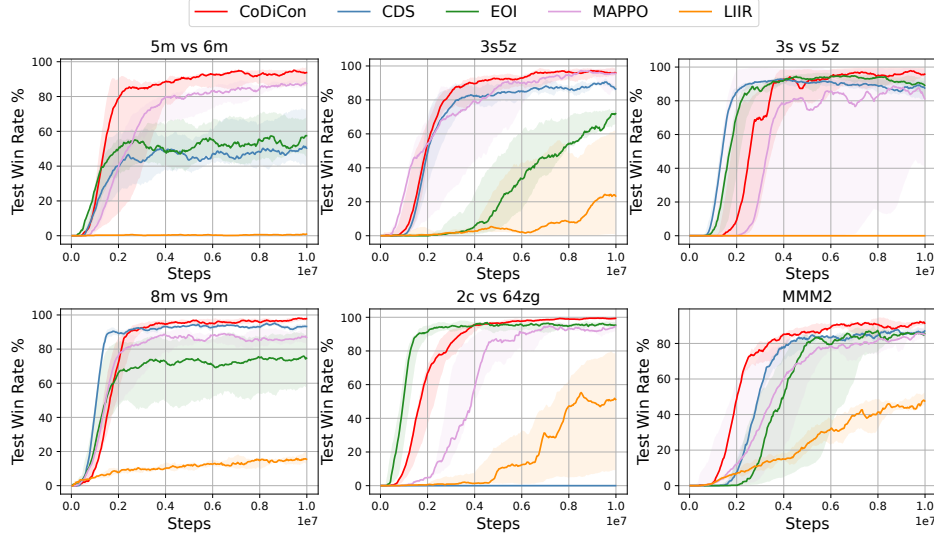
Figure 4: Training curves compared with the baselines on SMAC.

rithm gradually widens, and the interval between the training curves becomes more and more obvious. This demonstrates that our algorithm is able to effectively capture valuable states and actions in sparse reward-difficult environments, generating intrinsic rewards that encourage the agent to learn better.

5.3 Performance on SMAC

In this section, we test our algorithm in the StarCraft Micromanagement (SMAC) environment, a popular MARL benchmark. Each agent has attributes like health, weapon CD, shield, unit type, last action, and relative distances to observed units, with enemy units sharing similar attributes except CD. In partially observable settings, agents gather information within a circular range. The action space includes four movement directions, k attack actions (where k is the max number of enemies), a stop action, and a no-op action, with invalid actions masked. We evaluate on scenarios: 3s_vs_5z, 8m_vs_9m, MMM2, 2c_vs_64zg, 5m_vs_6m, 3S5Z. Only the Zealot is melee, the Medivac is a non-attacking support unit, and others are ranged. Agents receive a joint team reward based on total damage, with a significant bonus for winning.

We present the experimental results of the comparison in Figure 4. Our algorithm outperforms the baseline algorithms in most environments, with performance comparable to EOI only in the environment of 2c_vs_64zg. Both CDS and EOI achieve performance close to our algorithm on some maps, such as 3s_vs_5z and 8m_vs_9m. However, their performance on 5m_vs_6m and 2c_vs_64zg are highly unstable, indicating that the diversity mechanisms in CDS and EOI lack effective exploration and utilization of the state space, which may lead to local optima. MAPPO shows relatively stable and high performance across all environments, but it still lags behind our algorithm in terms of training efficiency and convergence speed. In contrast, LIIR performs poorly on all SMAC maps, suggesting that the unconstrained learnable intrinsic rewards in LIIR make learning highly unstable, particularly in scenarios with denser rewards. In contrast, our algorithm provides
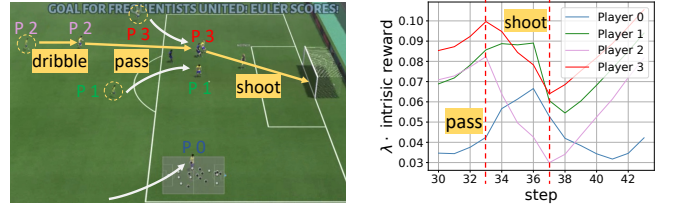

Figure 5: **Left.** Visualization of the trained policy. **Right.** The learned intrinsic reward curve for the agent players.

stable initial feedback during training by introducing a distinctly different prior intrinsic reward.

5.4 Visualizing the Learned Intrinsic Reward

The effectiveness of the strategy is demonstrated on the GRF counter_attack_hard mission, where we control four players positioned at different locations, cooperating to kick the ball from the half-field area into the opponent's goal. During the course of the play, the team encounters two opposing players controlled by the built-in AI, as well as a goalkeeper defending the opponent's goal. We visualize the trajectories of the players and the ball in Figure 5 (left), where the players' paths are represented by white arrows and the ball's path by yellow arrows. Agents are distinguished by numerical identifiers. Player 2 dribbles the ball and passes it to player 3, who receives it and shoots it into the goal. Player 1 collaborates with player 2 in the attack, applying pressure on the two opposing defenders, while player 0 moves toward the penalty area to support the offensive play. In Figure 5 (right), we plot the intrinsic reward curves corresponding to each player during the passing and shooting process.

- In the passing process, higher rewards are assigned to player 3 (the player taking the shot), player 2 (the player dribbling the ball), and player 1 (the player supporting the attack). In contrast, player 0, who is farther from the
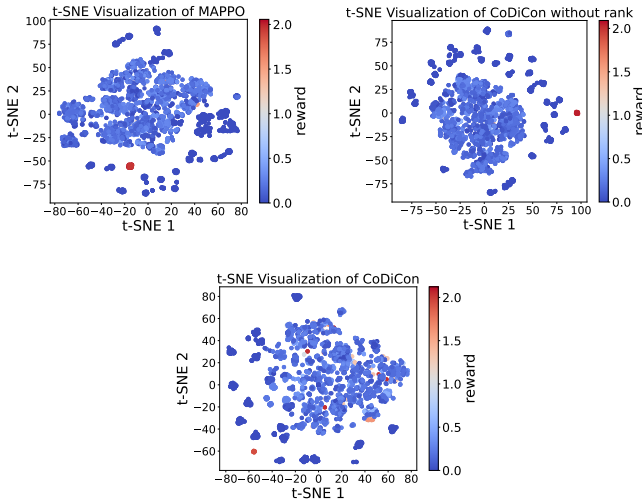
Figure 6: t-SNE results on the state-reward space.

Figure 7: The training curves of success rate for ablation methods.

offensive play, receives lower rewards.

- After player 2 passes the ball, his intrinsic reward drops sharply. During the flight of the ball, player 1 maintains a relatively high intrinsic reward because he is close to the two opposing defenders and supports the offense. Once player 3 receives the ball and takes the shot, the intrinsic rewards of the three players involved in the offensive play increase significantly.

This demonstrates that the intrinsic rewards in our algorithm effectively evaluate each agent's state and its contribution to the overall team performance.

5.5 Strategy Visualization in State-Reward Space

In addition to evaluating the performance of the trained policies, we are also interested in understanding how the learned intrinsic reward function influences policy learning. To analyze the intrinsic impact of our algorithm on policy training, we combine the state space and reward space to form a high-dimensional state-reward space. In this space, we can intuitively compare the distribution of the learned policy across states and the rewards obtained by executing actions in these states. We use t-SNE [Van der Maaten and Hinton, 2008] to perform a low-dimensional mapping of the state-reward space, where the specific value of each point is represented by the environmental reward. Different colors are used on the graph to represent the values, with warmer colors indicating higher values, as shown in the figure 6. We compared CoDiCon, MAPPO, and the intrinsic reward algorithm without ranking (CoDiCon without rank) based on the results of the trained policies over 50 episodes. As shown in the figure, MAPPO explores more meaningless states and attempts actions over a larger area (larger canvas size) compared to algorithms utilizing intrinsic rewards, but this does not lead to higher rewards. In contrast to the policy without ranked intrinsic rewards, our algorithm avoids meaningless exploration in adjacent feature spaces with high feature similarity (the points are more dispersed yet relatively concentrated)
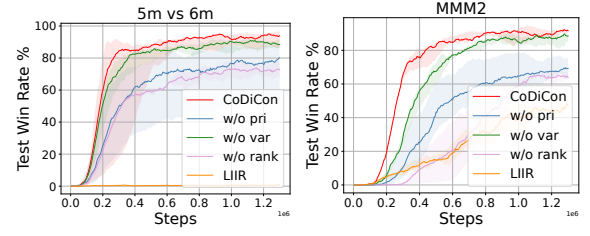
and identifies high-value actions at key nodes (points with more warm colors). The possible reason is that our algorithm facilitates mutual learning among agents for high-reward actions, and the differences in intrinsic rewards make the strategy more complex, diverse, and efficient.

5.6 Ablation Study

In our algorithm, two loss functions are designed to promote intrinsic reward variability and constrain the training of the ranking module parameters: the priori intrinsic reward MSE loss and the intrinsic reward maximum variance loss. To evaluate the effectiveness of these constraints, we conduct ablation experiments on the two loss functions as shown in Figure 7. Here, w/o pri refers to the algorithm without the priori loss constraint, w/o var refers to the algorithm without the variance loss constraint, and w/o rank refers to the algorithm without both loss constraints. We perform comparative experiments on the 5m_vs_6m and MMM2 scenarios in SMAC, and the results in both scenarios are consistent. These results indicate that both loss functions make positive contributions to policy training, outperforming the performance achieved without the intrinsic reward ranking loss, thereby demonstrating that both constraints are effective. Notably, using only the priori intrinsic reward constraint achieves higher performance and faster convergence compared to using only the variance constraint. A possible explanation is that reinforcement learning involves a large number of low-value states during the early stages of training, where distinguishing these low-value states and assigning them lower intrinsic rewards is critical. In contrast, variance constraints primarily become effective during the later stages of training.

6 Conclusion

We propose a novel competitive intrinsic reward multi-agent algorithm, CoDiCon, which enables each agent to learn an intrinsic reward with ranking properties. This approach effectively guides mutual competition and learning among agents, enhancing the efficiency of learning diverse strategies even when the environment provides only a single team reward. Our algorithm is formulated as a constrained bilevel optimization problem, theoretically ensuring that the final optimization objective aligns with maximizing the original environmental rewards. Experimental results on the SMAC, GRF, and Pac-Men environments demonstrate that CoDiCon outperforms existing state-of-the-art methods in terms of performance. Furthermore, case studies further validate the effectiveness of our intrinsic reward design.

References

[Andrychowicz *et al.*, 2016] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

[Colby *et al.*, 2015] Mitchell K Colby, Sepideh Kharaghani, Chris HolmesParker, and Kagan Tumer. Counterfactual exploration for improving multiagent learning. In *AAMAS*, pages 171–179, 2015.

[Du *et al.*, 2019] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[Foerster *et al.*, 2018] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Haarnoja *et al.*, 2019] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. In Antonio Bicchi, Hadas Kress-Gazit, and Seth Hutchinson, editors, *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, 2019.

[Hostallero *et al.*, 2019] Wan Ju Kang David Earl Hostallero, Kyunghwan Son, Daewoo Kim, and Yung Yi Qtran. Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR*, 2019.

[Hu *et al.*, 2019] Yeping Hu, Alireza Nakhaei, Masayoshi Tomizuka, and Kikuo Fujimura. Interaction-aware decision making with adaptive strategies under merging scenarios. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 151–158. IEEE, 2019.

[Jiang and Lu, 2021] Jiechuan Jiang and Zongqing Lu. The emergence of individuality. In *International Conference on Machine Learning*, pages 4992–5001. PMLR, 2021.

[King *et al.*, 2009] Eden B King, Michelle R Hebl, and Daniel J Beal. Conflict and cooperation in diverse workgroups. *Journal of Social Issues*, 65(2):261–285, 2009.

[Kirchmeyer and Cohen, 1992] Catherine Kirchmeyer and Aaron Cohen. Multicultural groups: Their performance and reactions with constructive conflict. *Group & Organization Management*, 17(2):153–170, 1992.

[Li *et al.*, 2021] Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:3991–4002, 2021.

[Lowe *et al.*, 2017] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[Oliehoek and Amato, 2016] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[Rashid *et al.*, 2020] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[Santoro *et al.*, 2016] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Schulman, 2015] John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

[Sunehag *et al.*, 2017] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Van der Maaten and Hinton, 2008] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[Wang *et al.*, 2020] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling

multi-agent q-learning. In *International Conference on Learning Representations*, 2020.

[Wiering, 2000] Marco A Wiering. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158, 2000.

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

[Xu *et al.*, 2018] Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[Yu *et al.*, 2022] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[Zhou *et al.*, 2020] Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2007.02529*, 2020.