

LangMARL: Natural Language Multi-Agent Reinforcement Learning via Semantic Credit Assignment

Huaiyuan Yao^{1*} Longchao Da^{1*} Xiaoou Liu¹

Charles Fleming² Tianlong Chen³ Hua Wei¹

¹Arizona State University

{huaiyuan, longchao, xiaoouli, hua.wei}@asu.edu

²Cisco Research chfleming@cisco.com

³University of North Carolina at Chapel Hill tianlong@cs.unc.edu

*Equal Contribution

Abstract

Large language model (LLM) agents struggle to autonomously evolve coordination strategies in dynamic environments, largely because coarse global outcomes obscure the causal signals needed for local policy refinement. We identify this bottleneck as a multi-agent credit assignment problem, which has long been studied in classical multi-agent reinforcement learning (MARL) but remains under-addressed in LLM-based systems. Building on this observation, we propose LangMARL, a framework that brings credit assignment and policy gradient evolution from cooperative MARL into the language space. LangMARL introduces agent-level language credit assignment, pioneers gradient evolution in language space for policy improvement, and summarizes task-relevant causal relations from replayed trajectories to provide dense feedback and improve convergence under sparse rewards. Extensive experiments across diverse cooperative multi-agent tasks demonstrate improved sample efficiency, interpretability, and strong generalization. Code and project resources are available at https://darl-genai.github.io/LangMARL_homepage/

1 Introduction

LLMs have advanced multi-agent systems (MAS) by enabling effective collaboration on diverse complex tasks (Li et al., 2024; Zhao et al., 2025a). However, deploying these systems in dynamic environments remains challenging due to a lack of principled adaptive mechanisms. Most current MAS architectures rely on manually engineered prompts and static configurations, which prevent agents from evolving their strategies as task distributions change (Yao et al., 2025; Guo et al., 2024). A deeper obstacle is that global outcomes often provide insufficient guidance for local policy refinement, as shown in Fig. 1. Without attributing team performance to individual agents, existing systems

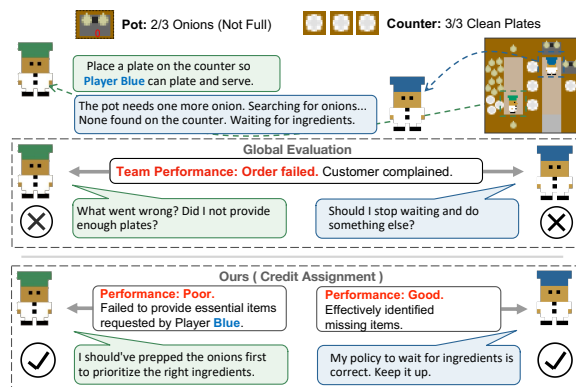
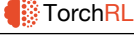


Figure 1: **Challenges in multi-agent credit assignment.** Global evaluation fails to pinpoint individual contributions, leading to ambiguous reflections. LangMARL addresses this by decomposing team performance into structured, agent-specific credits.

resort to coarse-grained optimization such as global reflections or monolithic prompt rewrites (Bo et al., 2024; Ma et al., 2026), leading to inefficient learning or coordination collapse. This bottleneck is fundamentally a *credit assignment* problem: *how to derive effective learning signals for each agent?* While well-studied in MARL (Foerster et al., 2018), it remains largely overlooked in LLM-based multi-agent systems (Li et al., 2025).

These gaps can be addressed by incorporating mature wisdom from the MARL field, which offers principled solutions to exactly these challenges. MARL formalizes multi-agent learning as an experience-driven optimization process (Tan, 2003), and the ‘*centralized training decentralized execution (CTDE)*’ paradigm (Lowe et al., 2017) enables agents to leverage global information during training while maintaining decentralized autonomy at execution. Crucially, MARL has developed mature credit assignment techniques such as counterfactual baselines (Foerster et al., 2018) and value decomposition (Sunehag et al., 2017) that precisely attribute global outcomes to individual agents.

1 Analogy in abstractions

	Formula	 TorchRL	 LangMARL
State	o_t^i	<code>TensorDict (obs)</code>	<code>langmarl.State (next obs)</code>
Policy Actor	$a_t^i \sim \pi_i(\cdot o_t^i)$	<code>ProbabilisticActor (net)</code>	<code>langmarl.LLMActor (policy text)</code>
Critic	$c_t^i = f(\tau)$	<code>ValueOperator ()</code>	<code>langmarl.LLMCritic ()</code>
Optimizer	$\pi_i \leftarrow \pi_i + \Delta\pi_i$	<code>Adam (module, parameters)</code>	<code>langmarl.LanguagePolicyOptimizer (policy text)</code>

2 Differentiation

TorchRL and LangMARL share the same syntax for MARL principle.

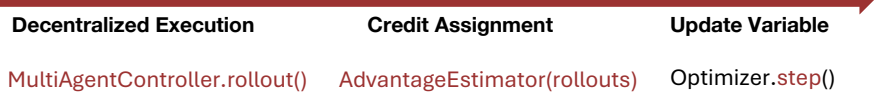


Figure 2: **An Easy-to-Use Toolkit for LangMARL.** LangMARL mirrors the syntax and abstractions of classical MARL libraries (e.g., TorchRL), redefining core components in natural language space, making LLM-based multi-agent optimization as straightforward to implement as standard deep RL pipelines.

However, directly applying MARL techniques to LLM-based agents is non-trivial, as LLMs operate in discrete language spaces as black-box models, creating a fundamental mismatch with conventional numeric learning signals. To tackle this challenge, we introduce **LangMARL**, a framework that draws on the core principles of MARL and credit assignment to design an automatic optimization system for multi-agent LLMs. Concretely, LangMARL consists of the following components: (1) *Language Policy Actors*, where each agent maintains a natural language policy and selects actions conditioned on textual observations; (2) *Centralized Language Critic*, which observes complete episodic trajectories and performs causal credit assignment in natural language, attributing team outcomes to individual agents; (3) *Language Policy Gradient Estimator*, which converts agent-specific language credits into language-form policy update directions, serving as an analogue of policy gradients; and (4) *Language Policy Optimizer*, which aggregates language gradients and applies semantic policy updates through LLM-based optimization operators. By grounding these MARL-inspired components, LangMARL places credit assignment at the core of multi-agent LLM optimization, where the Centralized Language Critic attributes global outcomes to individual agents and drives all downstream policy updates, enabling principled autonomous evolution of LLM-based multi-agent systems. Our contributions are summarized as follows.

- We identify credit assignment as the central bottleneck for autonomous optimization in multi-agent LLM systems, and introduce a paradigm that draws on core MARL principles

to enable LLM-based agents to autonomously learn and evolve coordination strategies.

- We propose LangMARL, a general framework that instantiates MARL-inspired automatic optimization for multi-agent LLMs, featuring language-parameterized policies, centralized credit assignment, and a credit-driven language policy optimizer. We modulate the framework into an easy-to-implement toolkit like TorchRL, as depicted in Fig. 2.
- Extensive experiments across reasoning, QA, coding, and game tasks demonstrate the effectiveness and scalability of LangMARL, with strong performance maintained across varying numbers of agents.

2 Related Work

2.1 Multi-agent LLMs

LLM-based multi-agent systems have demonstrated strong collaborative capabilities across diverse domains (Wu et al., 2023; Yao et al., 2026a), and recent work has begun to evaluate and survey LLMs’ abilities in multi-agent coordination (Tran et al., 2025; Yao et al., 2026b; Zhao et al., 2025a). However, most existing systems rely on static prompting approaches such as chain-of-thought reasoning (Wei et al., 2022) and manually designed agent pipelines, which depend heavily on human engineering and lack adaptive mechanisms. To reduce this reliance, automatic prompt engineering methods (Yang et al., 2024) have been integrated into agent systems, yet they optimize each agent

Table 1: Comparison across prompting and self-evolving paradigms. LangMARL provides the most comprehensive framework by unifying self-evolution, centralized critique, and explicit credit assignment.

Paradigm	Method	Self-Evolving	Central Critic	Text Gradient	Credit Assignment
Static Prompting	CoT (Wei et al., 2022)	✗	✗	✗	✗
	Agents (Zhou et al., 2023a)	✗	✗	✗	✗
Single-Agent Self-Evolving	Auto Prompt Engineer (AutoPE) (Zhou et al., 2023b)	✓	✓	✗	✗
	DSPy (Khattab et al., 2024)	✓	✗	✗	✗
	Reflexion (Shinn et al., 2023)	✓	✓	✗	✗
	TextGrad (Yuksekgonul et al., 2025)	✓	✓	✓	✗
Multi-Agent Self-Evolving	Agent Neural Network [†] (Ma et al., 2026)	✓	✓	✓	✗
	Symbolic (Ou et al., 2025)	✓	✓	✓	✗
	LangMARL (ours)	✓	✓	✓	✓

[†] Code is not publicly available at the time of writing.

independently without considering cooperative dynamics. And more multi-agent self-evolving frameworks (Ou et al., 2025; Zhao et al., 2025b; Hua et al., 2025; Hong et al., 2024) enable autonomous improvement by optimizing prompts, tools, and pipelines via language gradients.

2.2 Prompt Optimization

Prompt optimization aims to automatically discover or refine the textual instructions given to LLMs, reducing reliance on manual prompt engineering. Early work, such as Automatic Prompt Engineer (Zhou et al., 2023b), shows that LLMs can generate instructions that rival those from human-designed prompts. DSPy (Khattab et al., 2024) further abstracts LLM pipelines as declarative, modular programs with compiler-based prompt and demonstration optimization. A complementary direction explores learning through verbal feedback: Reflexion (Shinn et al., 2023) reinforces language agents via linguistic self-reflection stored in episodic memory, TextGrad (Yuksekgonul et al., 2025) backpropagates natural language feedback as textual gradients through computation graphs, and agent symbolic learning (Ou et al., 2025) extends language-based gradient descent to optimize entire agent pipelines. While these methods advance single-agent self-improvement, they do not address the multi-agent setting where credit must be disentangled among agents. LangMARL extends these self-evolving paradigms to multi-agent systems by introducing credit assignment, as shown in Table 1 and Appendix A.1.

3 Preliminaries

3.1 Cooperative MARL

We consider a cooperative MARL setting, where N agents interact with a shared environment to maximize a common team objective. The environment is modeled as a decentralized partially observable

Markov decision process (Dec-POMDP), defined by the tuple $\mathcal{M} = \langle S, \{A_i\}_{i=1}^N, \{O_i\}_{i=1}^N, P, R, \gamma \rangle$. Here, S denotes the global state space, A_i and O_i denote the action and observation spaces of agent i , respectively, P is the state transition function, R is a shared team reward function, and $\gamma \in [0, 1)$ is the discount factor. At each timestep t , the environment transitions according to $s_{t+1} \sim P(s_{t+1} | s_t, a_t^1, \dots, a_t^N)$, where $a_t^i \in A_i$ is the action taken by agent i . All agents receive a common reward $r_t = R(s_t, a_t^1, \dots, a_t^N)$. Each agent maintains a decentralized policy $\pi_i(a_i | o_i)$ that maps observations to actions. The joint policy is denoted by $\pi = (\pi_1, \dots, \pi_N)$. The learning objective is to maximize the expected discounted return: $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^T \gamma^t r_t \right]$.

3.2 Centralized Training with Decentralized Execution

A standard and effective paradigm for cooperative MARL is centralized training with decentralized execution (CTDE), which decouples training-time information access from execution-time constraints. At execution time, each agent acts solely based on its local observation: $a_t^i \sim \pi_i(\cdot | o_t^i)$, where $i = 1, \dots, N$. During training, a centralized critic is permitted to condition on global information, including the full state s_t and the joint action $a_t = (a_t^1, \dots, a_t^N)$. A typical realization is an action-value function: $Q^\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=t}^T \gamma^{k-t} r_k \mid s_t, a_t \right]$, it evaluates the expected team return under the joint policy.

3.3 Credit Assignment in Cooperative MARL

A central challenge in cooperative MARL is *credit assignment*: how to derive agent-specific learning signals from a shared team reward. At each timestep t , all agents observe the same scalar reward $r_t = R(s_t, a_t)$, where the joint action $a_t = (a_t^1, \dots, a_t^N)$ collectively determines the team out-

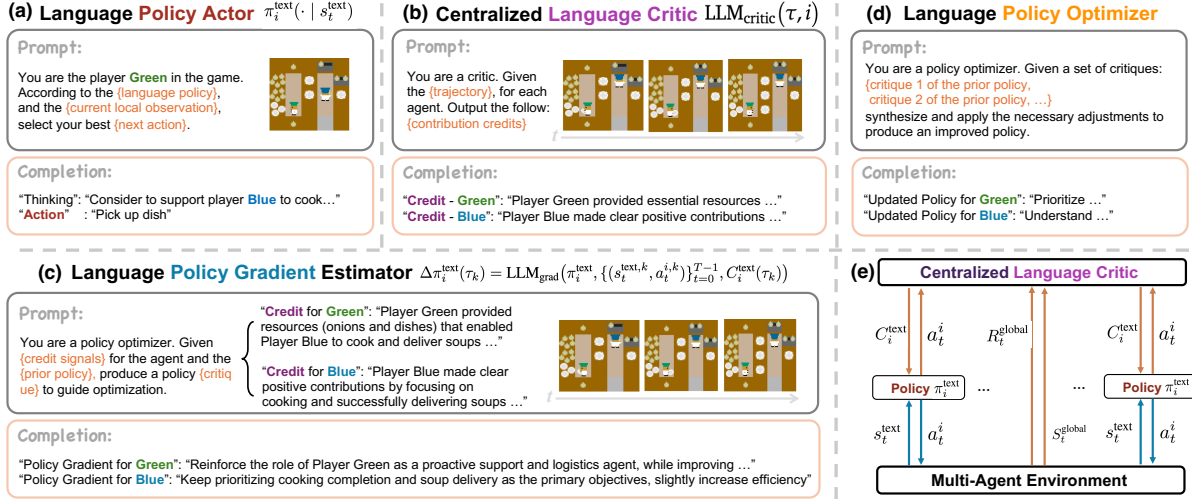


Figure 3: **The LangMARL System Pipeline.** The framework follows a CTDE paradigm: (i) *Language Policy Actors* execute decentralized actions, (ii) a *Centralized Language Critic* assigns trajectory-level causal credits, and (iii) the *Language Policy Optimizer* updates policies in natural language.

come. However, effective policy optimization requires feedback that reflects each agent’s individual contribution to this joint result. Rather than assuming an explicit per-agent reward decomposition, credit assignment is more generally formulated as the construction of agent-specific learning signals:

$$r_t \longrightarrow \{c_t^1, c_t^2, \dots, c_t^N\} \quad (1)$$

where c_t^i denotes the signal used to update agent i ’s policy. Under an actor–critic formulation, the policy gradient for agent i can be written as:

$$\nabla_{\theta_i} J(\pi) = \mathbb{E}_{\pi} \left[\sum_{t=0}^T \nabla_{\theta_i} \log \pi_i(a_t^i | o_t^i) c_t^i \right] \quad (2)$$

Making the design of c_t^i central to how team-level performance is attributed to individual agents.

4 Method: Natural Language MARL

In this section, we introduce LangMARL, a framework designed to optimize multi-agent LLM systems through credit assignment mechanisms. We first describe the overall paradigm, followed by the specific designs of the language policy actor, the centralized critic, and the language-based credit assignment and optimization, as illustrated in Fig. 3.

4.1 Language CTDE Paradigm

We adopt a cooperative multi-agent reinforcement learning setting as introduced in the preliminaries, and instantiate its core components directly in natural language. LangMARL follows a centralized-training–decentralized-execution paradigm, in

which credit assignment and policy optimization are implemented in language space.

During training, a centralized language critic has access to the complete episodic trajectory and produces agent-specific credit assignments expressed in natural language. These credits serve as structured, agent-specific training signals that summarize agents’ causal contributions across the entire trajectory. During execution, each agent acts independently using only its language policy, without access to global trajectories or feedback.

4.2 Language Policy Actor

Each agent i is instantiated as a large language model equipped with a *language policy*. Formally, we define a text-parameterized policy:

$$\pi_i^{\text{text}} : \mathcal{S}_i^{\text{text}} \rightarrow \mathcal{A}_i^{\text{text}} \quad (3)$$

where $\mathcal{S}_i^{\text{text}}$ denotes a natural-language encoding of agent i ’s observation, and $\mathcal{A}_i^{\text{text}}$ denotes the space of action descriptions available to agent i .

At timestep t , agent i samples its action via $a_t^i \sim \pi_i^{\text{text}}(\cdot | s_t^{\text{text}})$, which is implemented in practice by querying the LLM, conditioned on the current policy text and state description:

$$a_t^i = \text{LLM}_{\text{actor}}(\pi_i^{\text{text}}, s_t^{\text{text}}) \quad (4)$$

From a reinforcement learning perspective, π_i^{text} plays the same role as a stochastic policy π_{θ_i} in classical, parameterized MARL formulations. Its parameters, however, are implicitly encoded in natural language instructions, rules, or exemplars rather than numeric vectors.

Algorithm 1 LangMARL: Centralized Training

Require: Trajectory set $\{\tau_k\}_{k=1}^K$; Language policies $\{\pi_i^{\text{text}}\}_{i=1}^N$; Centralized language critic $\text{LLM}_{\text{critic}}$; Language policy gradient estimator LLM_{grad} ; Language policy optimizer LLM_{opt}

- 1: *#Agent-wise Credit Assignment*
- 2: **for** each trajectory τ_k **do**
- 3: **for** each agent $i \in \{1, \dots, N\}$ **do**
- 4: Generate language credit with Eq. (5)
- 5: **end for**
- 6: **end for**
- 7: *#Language Policy Gradient Estimation*
- 8: **for** each agent $i \in \{1, \dots, N\}$ **do**
- 9: **for** $k = 1$ to K **do**
- 10: Compute language gradient with Eq. (7)
- 11: **end for**
- 12: **end for**
- 13: *#Language Policy Optimization*
- 14: **for** each agent $i \in \{1, \dots, N\}$ **do**
- 15: Update language policy with Equation (8)
- 16: **end for**
- 17: **Output:** Updated language policies $\{\pi_i^{\text{text}}\}_{i=1}^N$

4.3 Centralized Language Critic for Credit Assignment

Consistent with Monte Carlo policy evaluation, LangMARL evaluates policies using complete episodic rollouts without temporal-difference bootstrapping. Given a trajectory $\tau = (s_0, \mathbf{a}_0, s_1, \mathbf{a}_1, \dots, s_T)$, we employ a centralized *language critic*, implemented as an LLM, that consumes the full episodic trajectory and performs causal attribution of individual agent contributions.

Rather than estimating a scalar value function, the language critic performs structured credit assignment by generating agent-specific feedback:

$$C_i^{\text{text}}(\tau) = \text{LLM}_{\text{critic}}(\tau, i) \quad (5)$$

where $C_i^{\text{text}}(\tau)$ describes how agent i 's sequence of actions $\{a_t^i\}_{t=0}^{T-1}$ positively or negatively influenced the final outcome of the episode.

4.4 Language Policy Gradient Estimator

In classical policy gradient methods, policy improvement for each agent is driven by optimizing:

$$\nabla_{\theta_i} J(\pi_i) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T-1} \nabla_{\theta_i} \log \pi_{\theta_i}(a_t^i | s_t) \cdot G_i(\tau) \right] \quad (6)$$

where $G_i(\tau)$ is a return or advantage estimate.

In LangMARL, actor-side policy updates are driven by a dedicated *policy gradient estimator*, implemented as an LLM. For each trajectory τ_k and agent i , we generate a *language policy gradient*:

$$\Delta \pi_i^{\text{text}}(\tau_k) = \text{LLM}_{\text{grad}}(\pi_i^{\text{text}}, C_i^{\text{text}}(\tau_k)) \quad (7)$$

The estimator conditions on the current policy, the agent's state-action sequence, and the language credit assigned to the trajectory. The output $\Delta \pi_i^{\text{text}}(\tau_k)$ is a language-form update direction that specifies how the policy should be adjusted in response to the experience τ_k . This module serves as a language-space analogue of policy gradient estimation, without computing numeric gradients.

4.5 Language Policy Optimizer

To perform batch policy optimization, LangMARL applies a single policy update based on language-form gradient information collected from multiple trajectories. For each agent i , we obtain a set of trajectory-level language gradients $\{\Delta \pi_i^{\text{text}}(\tau_k)\}_{k=1}^K$, as described in Section 4.4. Both gradient aggregation and policy update are implemented using large language models. Specifically, we employ an LLM-based *aggregator*, denoted by LLM_{agg} , to semantically integrate multiple trajectory-level gradients into a single update direction, and an LLM-based *language policy optimizer*, denoted by LLM_{opt} , to apply the resulting update to the policy.

Formally, the policy update is expressed as a composition of two LLM operators:

$$\pi_i^{\text{text}} \leftarrow \text{LLM}_{\text{opt}}\left(\pi_i^{\text{text}}, \text{LLM}_{\text{agg}}(\{\Delta \pi_i^{\text{text}}(\tau_k)\}_{k=1}^K)\right) \quad (8)$$

The aggregation operator LLM_{agg} serves as a language-space analogue of batch gradient averaging. Rather than performing arithmetic averaging, it semantically integrates multiple update directions by identifying consistent modification patterns, resolving conflicts among suggestions, and suppressing noisy or idiosyncratic gradients.

In general, LangMARL organizes policy optimization in natural language into a batch actor-critic style procedure. Trajectory-level update directions are first aggregated and then applied via LLM-based operators, reflecting the functional separation between gradient aggregation and optimizer steps in classical batch policy gradient methods,

Table 2: Comprehensive Comparison across QA, Reasoning, Coding and Multi-Agent Environments. All reported values represent the mean performance after 5 training iterations using the same backbone LLMs.

Category	Static-Prompting		Self-Evolving					Ours
Method	CoT	Agents	AutoPE	DSPy	Reflexion	TextGrad	Symbolic	LangMARL
<i>Natural Language Benchmarks (Accuracy / Pass Rate)</i>								
HotPotQA	38.8	37.5	39.8	43.9	59.1	57.3	44.8	60.2
MATH	23.2	23.8	22.5	17.3	49.4	53.8	38.8	56.0
HumanEval	59.2	59.5	63.5	66.7	70.1	68.9	64.5	73.2
<i>Multi-Agent Game: Overcooked-AI (Mean Reward)</i>								
Forced Coord.	73.3	68.4	71.9	62.1	138.5	85.3	104.2	148.9
Coord. Ring	140.0	122.3	133.4	141.4	157.8	163.8	148.5	184.4
Counter Circuit	40.0	43.6	38.2	52.4	68.7	54.5	55.1	77.8
Asymm. Adv.	226.7	202.7	143.5	217.3	204.1	230.6	232.4	244.4
Cramped Room	126.7	118.5	122.8	145.7	151.1	131.3	147.5	171.4
<i>Multi-Agent Game: Pistonball (Mean Reward)</i>								
$N = 10$	-0.9	11.6	5.8	20.0	23.7	33.5	21.1	37.2
$N = 12$	-0.1	-2.7	-0.8	7.8	-6.2	24.3	3.1	29.6
$N = 16$	-8.5	-1.0	-3.1	-0.9	8.9	10.1	-7.6	17.2
$N = 20$	-11.5	-3.8	-6.2	-7.3	-1.2	14.3	3.3	22.9

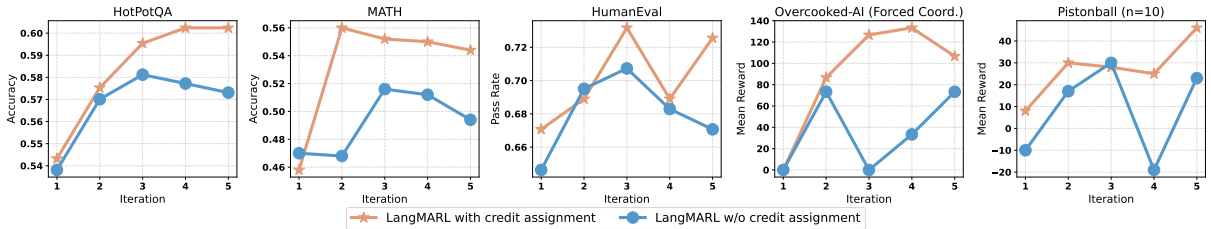


Figure 4: **Impact of credit assignment on the convergence quality of LangMARL.** The learning curves across five benchmark tasks demonstrate that credit assignment is pivotal for efficient policy optimization and superior final performance. Without this mechanism (green lines), the models exhibit sub-optimal learning rates and significant instability, particularly in complex reasoning and multi-agent coordination scenarios.

while operating in a language space, as summarized in Algorithm 1, and the specific process for trajectory collection is outlined in Appendix A.2.

5 Experiments

We conduct comprehensive experiments across diverse domains including code generation, multi-hop reasoning, mathematics, and strategic games to evaluate the efficacy of LangMARL. Our evaluation is guided by the following Research Questions:

- **RQ1: Overall Effectiveness.** Does LangMARL enhance team performance in cooperative tasks compared to other baselines?
- **RQ2: Credit Assignment Quality.** How effectively does the Centralized Language Critic disentangle individual agent contributions compared to global reflection?
- **RQ3: Robustness and Scalability.** How does LangMARL’s performance scale with respect to: (i) the reasoning capacity of the underlying LLM, and (ii) the volume of trajectory data available for policy evolution?

5.1 Environment Settings

We evaluate LangMARL across two distinct categories of environments (see Appendix A.3).

• **Strategic Game Environments:** Strategic games feature fixed agent roles and physical constraints, providing a controlled setting to evaluate credit assignment ability. *Overcooked-AI* (Carroll et al., 2019) is a benchmark for coordination where two agents collaborate to cook and deliver soup. We employ a hierarchical action space where high-level semantic subgoals are translated into primitive movements. *Pistonball* (Terry et al., 2021) involves a team of pistons working together to move a ball. Each agent has severe partial observability, seeing only a local vertical slice of the environment.

• **Cooperative Language Tasks:** To assess LangMARL’s performance in open-task environments, we reframe three prominent NLP datasets *HumanEval* (coding), *HotPotQA* (multi-hop reasoning), and *MATH* (mathematics) into a multi-agent reinforcement learning framework. Unlike the fixed-role games, these tasks evaluate the framework’s capability to foster the emergence of novel

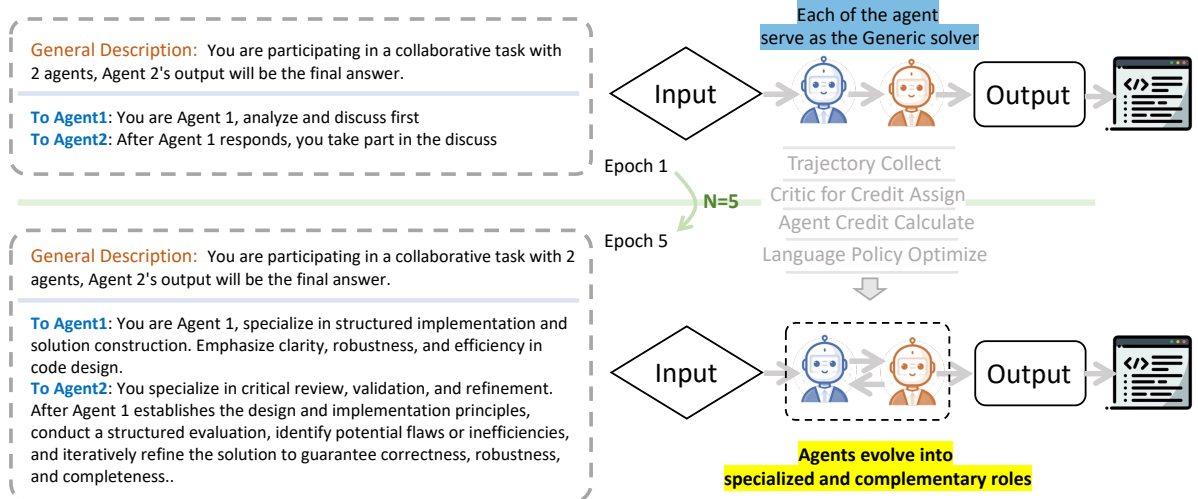


Figure 5: **Emergent role specialization in LangMARL.** Top: In the initial symmetric setup, both agents operate as generic problem solvers without predefined role differentiation. Bottom: After iterative trajectory-level credit assignment and language-based policy optimization, agents self-organize into complementary roles, such as in the coding scenario, Agent 1 specializing in structured implementation and Agent 2 in critical evaluation and refinement. This division of labor is not explicitly specified in the prompt but emerges through centralized language credits.

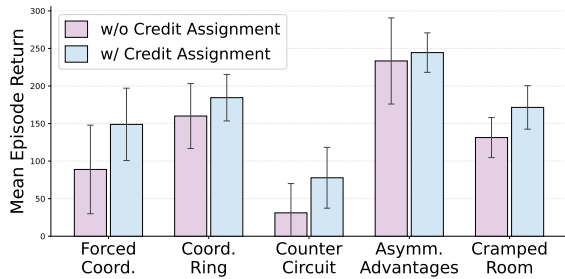


Figure 6: Ablation study on credit assignment.

Table 3: Ablation results on credit assignment quality.

Layout	w/o Credit	w/ Credit	Δ
Forced Coord.	82.7 $\pm$ 61.1	148.9 $\pm$ 48.1	+66.2**
Coord. Ring	160.1 $\pm$ 40.3	184.4 $\pm$ 30.9	+24.3**
Counter Circuit	30.9 $\pm$ 36.4	77.8 $\pm$ 40.4	+46.9***
Asymm. Adv.	227.3 $\pm$ 59.8	244.4 $\pm$ 26.2	+17.1
Cramped Room	128.6 $\pm$ 29.1	171.4 $\pm$ 28.9	+42.8***

*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$.

coordination strategies.

5.2 Baselines and Comparisons

We compare LangMARL with prompting and self-evolving methods listed in Table 1. Specifically, these baselines can be categorized in two groups: (1) *Static prompting*. CoT (Wei et al., 2022) and Agents (Zhou et al., 2023a) use fixed prompts without adaptive updates. (2) *Prompt optimization*. AutoPE (Zhou et al., 2023b) and DSPy (Khattab et al., 2024) iteratively refine prompts, but optimize each agent independently. Reflexion (Shinn et al., 2023) and TextGrad (Yuksekgonul et al., 2025) update prompts using linguistic feedback. Symbolic Learning (Ou et al., 2025) enables multi-agent optimization and relies on shared team-level signals.

5.3 Results of Overall Performance (RQ1)

In this experiment, following existing work (Ou et al., 2025; Khattab et al., 2024; Shinn et al., 2023), GPT-3.5-turbo is employed for natural language benchmarks, while GPT-4o-mini is utilized for both

game environments. As summarized in Table 2, LangMARL achieves superior performance across all benchmarks, consistently outperforming both static prompting and self-evolving baselines.

Natural Language Benchmarks. On reasoning-intensive tasks, LangMARL significantly surpasses other optimization methods on *MATH*, *HotPotQA* and *HumanEval* datasets. These results demonstrate that addressing the credit assignment problem during interaction optimization allows for robust performance gains across diverse benchmarks. Notably, the performance on these benchmarks exhibits a strong positive correlation with rollout budgets, suggesting that LangMARL effectively translates increased computational resources into reasoning improvements.

Strategic Game Environments. In *Overcooked-AI*, LangMARL attains the highest mean reward across all layouts. In *Pistonball*, LangMARL maintains stable positive returns as team size scales to $N = 20$, whereas baselines like TextGrad degrade.

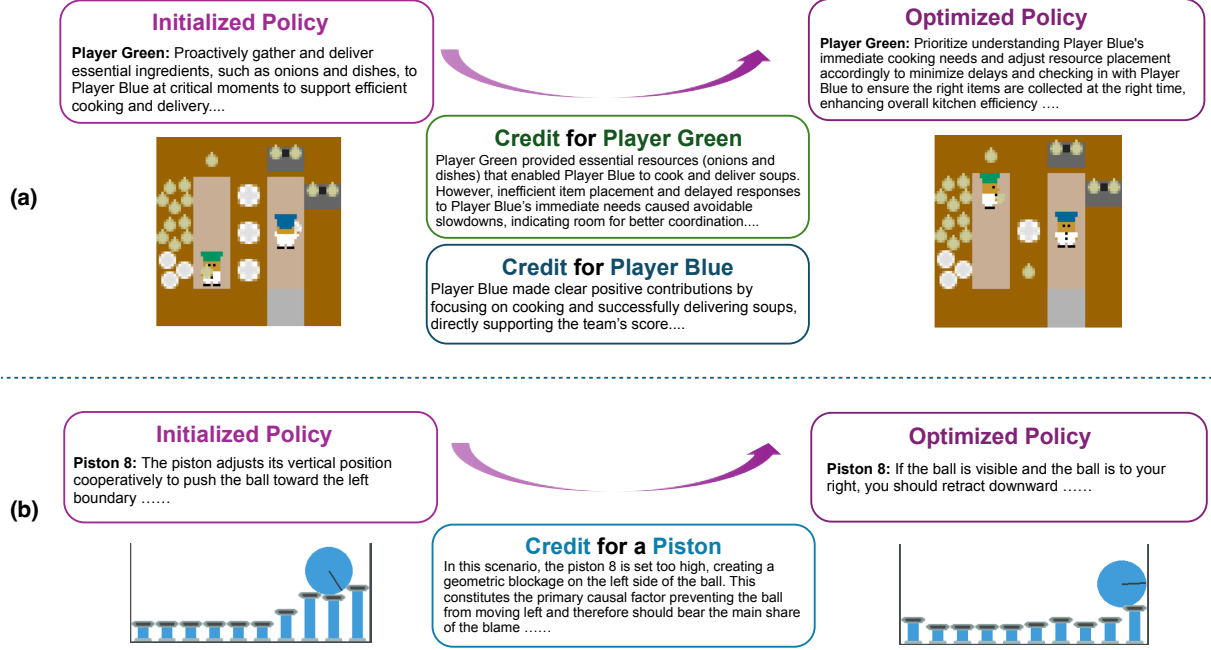


Figure 7: **Examples of language-based credit assignment and policy optimization in LangMARL.** (a) Overcooked: the centralized language critic analyzes a full trajectory and produces agent-specific credits, which are then used to revise the language policy of Player Green, resulting in improved coordination with Player Blue. (b) Pistonball: the language critic identifies a causally responsible piston from the trajectory and assigns targeted credit, leading to a refined local policy that corrects blocking behavior.

This confirms that our centralized critic effectively enables scalable cooperation under partial observability and sparse rewards. Furthermore, LangMARL demonstrates superior sample efficiency in large-scale coordination tasks.

5.4 Credit Assignment Quality (RQ2)

We validate the efficacy of the Centralized Language Critic through learning stability and qualitative emergent behaviors.

Ablation on Credit Assignment. Fig. 6 evaluates the impact of explicit agent-wise credit assignment. Removing credit assignment and training all agents with a shared global reward leads to a clear performance drop in both Overcooked-AI and Pistonball. These results confirm that explicit credit assignment is a key component for stable learning and effective coordination in LangMARL.

Learning Dynamics and Stability. As shown in Fig. 4, centralized credit assignment (blue lines) is pivotal for stable optimization. Across all benchmarks, its inclusion leads to faster convergence and higher final performance, whereas models without it (green lines) suffer from learning instability and suboptimal plateaus, particularly in complex tasks

like *MATH* and *Pistonball*.

Emergent Role Specialization. We showcase that LangMARL can automatically induce sophisticated functional specialization without predefined roles: (1) **Collaborative Coding:** Agents evolve from symmetric solvers into complementary roles over five epochs. Agent 1 focuses on *structured implementation* while Agent 2 specializes in *critical review and refinement*, as shown in Fig. 5. (2) **Strategic Games:** Fig. 7 illustrates the critic resolving bottlenecks. In *Overcooked-AI*, Player Green optimizes resource placement based on identified needs of Player Blue. In *Pistonball*, the critic identifies and corrects "geometric blockages" caused by specific pistons.

Human Verification of Credit Assignment. To further verify that the learned per-agent credits reflect true causal responsibility, we conduct a human-agreement study on Overcooked-AI. We sample 70 trajectories in total, with 14 trajectories from each layout, and ask human annotators to identify the agent primarily responsible for each outcome. We then compare the critic's top attribution with the human judgments.

As shown in Table 4, the critic's attribution

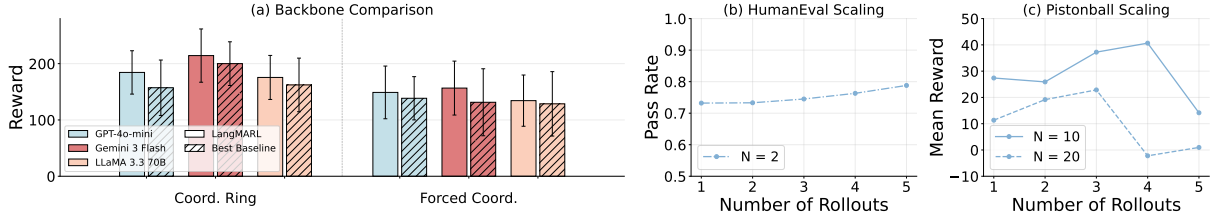


Figure 8: **Sensitivity analysis of LangMARL components.** (a) Performance comparison of various LLM backbones against the best baseline Reflexion. (b-c) Impact of Monte Carlo rollout counts on HumanEval and Pistonball.

agrees with human labels on 89.9% of trajectories overall. The agreement reaches 100.0% on Forced Coordination, where responsibility is particularly clear because neither agent can complete the task alone. These results provide direct evidence, independent of downstream task performance, that the Centralized Language Critic captures meaningful per-agent causal responsibility.

Table 4: Human verification of credit assignment quality on Overcooked-AI environment.

Metric	Forced Coord.	Coord. Ring	Counter Circuit	Asymm. Adv.	Cramped Room
Accuracy	100.0%	85.7%	92.9%	78.6%	92.9%

5.5 Sensitivity Analysis (RQ3)

We evaluate robustness across model backbones and rollout budgets.

Backbone Robustness. As illustrated in Fig. 8(a), LangMARL demonstrates strong compatibility across various LLM backbones, including GPT-4o-mini, Gemini-3-Flash, and LLaMA-3.3-70B. In both layouts of *Coord. Ring* and *Forced Coord.*, LangMARL consistently achieves higher mean rewards compared to the strongest baselines, regardless of the underlying model. Notably, Gemini-3-Flash paired with our framework delivers the peak performance, while even the open-weight LLaMA-3.3-70B architecture maintains competitive results, underscoring LangMARL’s robustness and its generalizability across different scales and families of LMs.

Impact of Rollout Budget. Furthermore, the impact of rollout budgets varies significantly with task complexity and agent scale. In few-agent tasks such as *HumanEval* (Fig. 8b), increasing rollouts leads to a consistent and significant performance gain. However, in the large-scale multi-agent *Pistonball* environment (Fig. 8c), the benefit of an increased budget is highly sensitive to the team size N . While performance for a large team ($N = 10$

or 20) initially improves, it subsequently reaches a plateau. This suggests an optimal rollout frequency exists, as excessive rollouts in large-scale systems can exacerbate policy bias despite improving coverage in simpler settings.

6 Conclusion

This paper establishes a framework designed to optimize multi-LLM agents through credit assignment mechanisms inspired by MARL. Our experiments across strategic games and reasoning benchmarks demonstrate that LangMARL consistently outperforms other self-evolving baselines. Ablation studies and learning dynamics confirm that agent-specific credit assignment is the primary driver of optimization stability, enabling the spontaneous emergence of functional specialization without manual engineering. Furthermore, sensitivity analysis shows that these performance gains are robust across diverse LLM backbones. Future research will explore hierarchical sub-goal decomposition and dynamic agent synthesis to facilitate more fluid, large-scale autonomous collaboration.

Acknowledgement

The work was partially supported by Cisco Faculty Award, Amazon Research Award, and NSF awards #2442477, #2550203, #2616632, #2623317, #2536297, and #2613637. The views and conclusions in this paper should not be interpreted as representing any funding agencies.

Limitations

Despite the promising results, several challenges remain to be addressed in future research:

Scaling to Long-horizon Agentic Tasks. A primary limitation is LangMARL’s performance in long-horizon workflows, such as autonomous software development, where sparse rewards and error propagation over hundreds of steps pose significant challenges. To address this, future work could

investigate the hierarchical decomposition of extended trajectories into manageable semantic sub-goals. This approach would allow the centralized critic to provide more granular feedback, maintaining the consistency of semantic credit assignment even when the causal link between an initial intervention and the final outcome is significantly attenuated over time.

Dynamic Sub-agent Synthesis and Orchestration. Currently, our framework operates within a pre-defined agent topology, which may limit its adaptability in real-world collaboration requiring fluid organizational structures. We aim to extend LangMARL toward "on-the-fly" synthesis and instantiation of new agents. A potential solution involves developing mechanisms for dynamic task partitioning and the evolution of hierarchical communication protocols. This would enable the system to autonomously spawn specialized sub-agents and adaptively scale its computational resources in response to emerging sub-tasks, a direction we plan to explore in subsequent studies.

Ethical Considerations

LangMARL is a research framework for improving coordination in multi-agent LLM systems, evaluated only in controlled benchmark environments. If applied to real-world or high-stakes settings, it should be used with human oversight, explicit safety constraints, and domain-specific evaluation. Because LangMARL relies on LLMs to generate critiques and assign agent-level credit, its feedback may contain biases or incorrect causal explanations, especially for ambiguous trajectories. We include human verification, but its scale and coverage are limited.

Our experiments use existing benchmarks and simulated environments and do not involve sensitive personal data. The human agreement study only asks annotators to judge agent responsibility from sampled trajectories. We also acknowledge that LLM-based critics and rollouts increase computational cost, and we encourage future work on more efficient and reliable credit assignment methods.

References

Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. 2024. Reflective multi-agent collaboration based on large

language models. *Advances in Neural Information Processing Systems*, 37:138595–138631.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. 2019. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32.

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

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. [Large language model based multi-agents: A survey of progress and challenges](#). *Preprint*, arXiv:2402.01680.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiaowu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pages 23247–23275.

Yuncheng Hua, Sion Weatherhead, Mehdi Jafari, Hao Xue, and Flora D. Salim. 2025. [Socia- \$\nabla\$: Textual gradient meets multi-agent orchestration for automated simulator generation](#). *Preprint*, arXiv:2505.12006.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. Dspy: Compiling declarative language model calls into self-improving pipelines.

Wenhao Li, Dan Qiao, Baoxiang Wang, Xiangfeng Wang, Wei Yin, Hao Shen, Bo Jin, and Hongyuan Zha. 2025. Multi-agent credit assignment with pre-trained language models. In *International Conference on Artificial Intelligence and Statistics*, pages 1945–1953. PMLR.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinity*, 1(1):9.

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

Xiaowen Ma, Yunpu Ma, Chenyang Lin, Sikuan Yan, Jinhe Bi, Zixuan Cao, Yijun Tian, Volker Tresp, and Hinrich Schuetze. 2026. [Self-evolving multi-agent systems via textual backpropagation](#). *Preprint*, arXiv:2506.09046.

- Yixin Ou, Wangchunshu Zhou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, and 1 others. 2025. Symbolic learning enables self-evolving agents. *AI Open*.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.
- Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, and 1 others. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.
- Ming Tan. 2003. [Multi agent reinforcement learning independent vs cooperative agents](#).
- J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, and 1 others. 2021. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043.
- Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O’Sullivan, and Hoang D. Nguyen. 2025. [Multi-agent collaboration mechanisms: A survey of llms](#).
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. [Autogen: Enabling next-gen llm applications via multi-agent conversation](#).
- Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pages 12028–12068.
- Huaiyuan Yao, Longchao Da, Vishnu Nandam, Justin Turnau, Zhiwei Liu, Linsey Pang, and Hua Wei. 2025. Comal: Collaborative multi-agent large language models for mixed-autonomy traffic. In *Proceedings of the 2025 SIAM International Conference on Data Mining (SDM)*, pages 409–418. SIAM.
- Huaiyuan Yao, Wanpeng Xu, Justin Turnau, Nadia Kellam, and Hua Wei. 2026a. Instructional agents: Reducing teaching faculty workload through multi-agent instructional design. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4087–4109.
- Huaiyuan Yao, Wanpeng Xu, Justin Turnau, Nadia Kellam, and Hua Wei. 2026b. Instructional agents: Reducing teaching faculty workload through multi-agent instructional design. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4087–4109.
- Mert Yuksekogunul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616.
- Lina Zhao, Jiaxing Bai, Zihao Bian, Qingyue Chen, Yafang Li, Guangbo Li, Min He, Huaiyuan Yao, and Zongjiu Zhang. 2025a. [Autonomous multi-modal llm agents for treatment planning in focused ultrasound ablation surgery](#). *Preprint*, arXiv:2505.21418.
- Wanjia Zhao, Mert Yuksekogunul, Shirley Wu, and James Zou. 2025b. Sirius: Self-improving multi-agent systems via bootstrapped reasoning. *arXiv preprint arXiv:2502.04780*.
- Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Xiangru Tang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. 2023a. [Agents: An open-source framework for autonomous language agents](#). *Preprint*, arXiv:2309.07870.
- Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023b. [Large language models are human-level prompt engineers](#). *Preprint*, arXiv:2211.01910.

A Appendix

A.1 Paradigm Comparison

We compare LangMARL against existing optimization paradigms (Fig. 9) to highlight the evolution toward precise multi-agent coordination. Early methods like **Auto Prompt Engineer (1)** treat policies as black boxes, relying on global outcomes to propose updates; however, they lack the internal feedback loops necessary for complex joint strategy spaces. **Reflexion (2)** introduces a shared memory for agents to process past failures, yet these reflections are often monolithic, leading to a "bystander effect" where individual agents cannot distinguish their specific contributions to a collective error. **TextGrad and Symbolic (3)** advance this by utilizing LLMs for global evaluations, but they typically generate a single, joint feedback signal. In tasks like Overcooked or Pistonball, this prevents the system from pinpointing specific bottlenecks, often causing "credit drift" where a high-performing agent's policy is incorrectly modified due to a teammate's mistake. In contrast, **LangMARL (4)** introduces an explicit individual credit assignment mechanism. By utilizing a Centralized Language Critic to decompose global trajectories into agent-specific "Language Gradients," our approach captures the causal dependencies between agents (e.g., identifying when one agent's action blocks another). This granular feedback transforms global rewards into localized, actionable guidance, preventing coordination collapse and enabling the spontaneous emergence of the functional specializations observed in our experiments.

A.2 Implementation of Experience Collection

In this section, we present the formal procedure for the decentralized experience collection phase of LangMARL, as summarized in Algorithm 2. This stage is crucial for operationalizing the Decentralized Execution part of the CTDE paradigm within a language space. **Process Overview:** During each training iteration, the system performs K independent rollouts. Each agent i maintains its own decentralized language-based policy π_i^{text} , which consists of a set of textual instructions and few-shot examples. At each time step t , agents process the current textual state s_t^{text} —which includes environmental observations and potential communication history—to sample a discrete action $a_t^{i,k}$ in parallel.

Algorithm 2 LangMARL: Decentralized Experience Collection

Require: Number of agents N ; Language policies $\{\pi_i^{\text{text}}\}_{i=1}^N$; Number of rollouts K ; Episode horizon T

- 1: **for** $k = 1$ to K **do**
- 2: Initialize environment
- 3: Obtain initial textual state s_0^{text}
- 4: **for** $t = 0$ to $T - 1$ **do**
- 5: **for each agent** $i \in \{1, \dots, N\}$ **in parallel do**
- 6: Sample action from language policy:
 $a_t^{i,k} \sim \pi_i^{\text{text}}(\cdot \mid s_t^{\text{text}})$
- 7: **end for**
- 8: Execute joint action $\mathbf{a}_t^k = (a_t^{1,k}, \dots, a_t^{N,k})$
- 9: Observe next textual state s_{t+1}^{text}
- 10: **end for**
- 11: Store complete trajectory: $\tau_k = (s_0^{\text{text}}, \mathbf{a}_0^k, \dots, s_T^{\text{text}})$
- 12: **end for**
- 13: **Output:** Trajectory set $\{\tau_k\}_{k=1}^K$

A.3 Environment Settings

We briefly summarize the key environment configurations used in our experiments, including the number of agents, observation spaces, action spaces, and reward semantics.

- **Overcooked-AI** (Carroll et al., 2019) is a cooperative environment in which two agents collaborate on a cooking task, as shown in Fig. 10. Each episode proceeds in synchronized time steps, where both agents act simultaneously. At each step, each agent receives a partial observation consisting of a symbolic, egocentric grid representation of the kitchen layout, including nearby terrain, objects, and the other agent's position and orientation.

For action modeling, we adopt the hierarchical action space. At the high level, an agent selects a semantic subgoal (e.g., fetch ingredient, deliver soup), while at the low level, the policy outputs primitive actions from a discrete set including movement, interaction, and orientation changes. This two-layer design enables structured temporal abstraction while retaining compatibility with the original Overcooked-AI dynamics. The reward function in Overcooked-AI is sparse and team-based: a positive reward is issued only when a soup is successfully delivered.

- **Pistonball** (Terry et al., 2021) is a cooperative

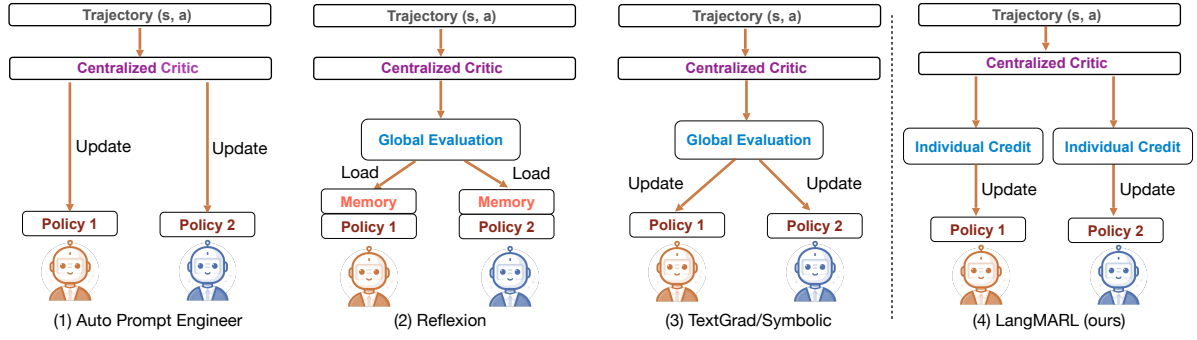


Figure 9: Comparison of optimization paradigms in multi-agent systems. While existing methods rely on direct updates (1), shared memory (2), or collective global evaluations (3), LangMARL (4) introduces an explicit individual credit assignment mechanism. This allows the centralized critic to decompose global feedback into agent-specific guidance, facilitating more precise policy optimization in collaborative tasks.

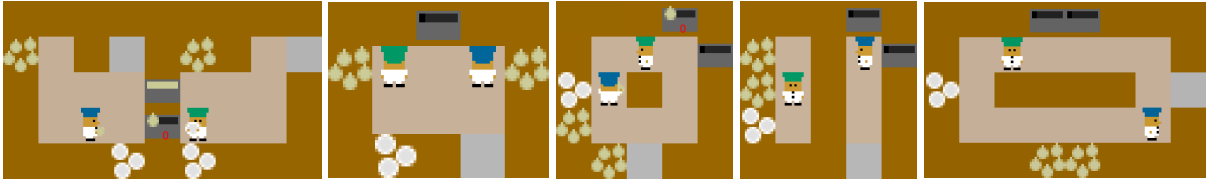


Figure 10: **Overcooked-AI Layouts.** From left to right: Asymmetric Advantages tests the adoption of high-level strategies based on individual strengths; Cramped Room presents low-level coordination challenges within a confined, collision-prone space; Coordination Ring requires synchronized navigation between opposite corners; Forced Coordination removes collision friction to focus on joint strategy, as neither agent can serve dishes alone; and Counter Circuit necessitates a non-obvious strategy of passing ingredients over counters rather than carrying them.

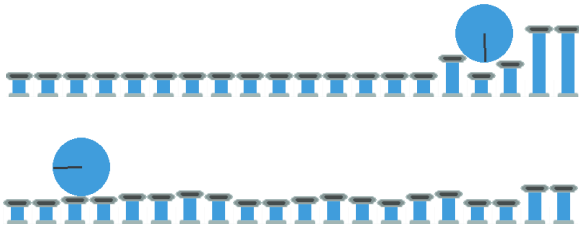


Figure 11: **Pistonball Layouts.** The upper panel illustrates a failure case where poor coordination among piston agents leads to an uneven surface, trapping the ball and resulting in a significantly low negative score due to the accumulated time penalty. In contrast, the lower panel demonstrates successful coordination, where agents synchronously adjust their vertical positions to smoothly propel the ball toward the target.

multi-agent environment with variable team sizes. In our experiments, the number of piston agents ranges from 10 to 20, covering different levels of coordination difficulty. The environment evolves in discrete time steps governed by a physics simulator, as shown in Fig. 11.

Each piston agent receives a local, high-dimensional pixel-based observation capturing a

narrow vertical slice of the environment centered around the piston, including the ball’s relative position and nearby physical context. No agent has access to global state information, resulting in severe partial observability. At each step, each agent selects from a small discrete set of vertical movement actions, which control the piston’s vertical force. All agents share a single global cooperative reward composed of two terms: a distance-based component proportional to the ball’s leftward progress in the current time step, and a constant time penalty (default: -0.1) applied at every step to incentivize faster task completion. This reward structure encourages collective physical coordination while still requiring agents to infer their individual contributions from local observations.

A.4 Learning Dynamics across Overcooked Scenarios

To supplement the primary results, Figure 12 illustrates the learning curves across four representative Overcooked-AI layouts. The dynamics vary significantly due to the distinct coordination challenges inherent in each map:

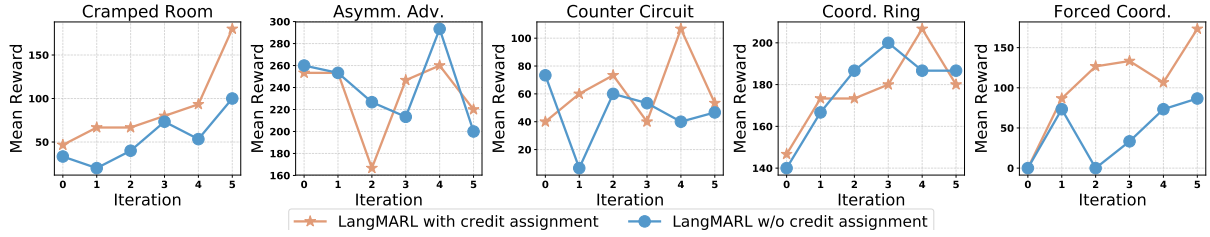


Figure 12: Learning curves on Overcooked-AI layouts. Comparison of LangMARL with (orange) and without (blue) credit assignment. Results across five maps show that language-level credit assignment significantly improves mean reward and convergence stability, especially in layouts requiring high-degree coordination (e.g., Coord. Ring and Forced Coord.).

- **Cramped Room:** While simpler, it often leads to spatial conflict. LangMARL quickly resolves these "collision" bottlenecks through precise language feedback, outperforming Reflexion’s noisier global updates.
- **Asymmetric Advantages:** This scenario necessitates labor division. LangMARL demonstrates a clear "role discovery" capability, where the critic assigns credit based on individual task efficiency, leading to specialized policies (e.g., dedicated onion fetcher vs. cook).
- **Coordination Ring:** The high temporal delay in the ring layout causes "feedback dilution" for most baselines. LangMARL’s dense, trajectory-based credit assignment enables agents to maintain stable clockwise/counter-clockwise coordination.
- **Forced Coordination:** This is the most difficult layout due to the required item-passing. While other self-evolving methods show flat learning curves, LangMARL successfully bridges the coordination gap, showing a robust and steady increase in reward.

A.5 Prompt Templates

In this section, we provide the specific prompt templates used for the four core language operators in LangMARL. These templates define the functional roles of language modules in the system and illustrate how policy execution, trajectory evaluation, credit assignment, and policy optimization are realized through natural language interactions.

All agents are instantiated with the same base LLM but guided by different functional instructions to achieve the Centralized Training, Decentralized Execution (CTDE) paradigm in language space, where agents act based on local observations while a centralized critic analyzes global trajectories to generate learning signals for policy evolution.

Prompt Template 1: Language Policy Actor

System Role: You are a specialized agent [Agent ID] in a cooperative environment.

Inputs:

- **Language Policy:** [Current text-based strategy, e.g., "Focus on supplying ingredients to the chef."]

- **Local Observation:** [Current textual description of the state, e.g., "You see a tomato at (3,4)."]

Instruction: Based on your current policy and the observation, choose the best action to maximize team reward.

Output Format:

- **Thinking:** [Reasoning process via Chain-of-Thought]
- **Action:** [The primitive or semantic action to execute]

Prompt Template 2: Centralized Language Critic

System Role: You are an expert analyst observing a multi-agent team trajectory.

Inputs:

- **Joint Trajectory τ :** [Full sequence of states and joint actions of all agents.]
- **Team Reward:** [Sparse/delayed reward signal, e.g., "Task Success"]

Instruction: Perform causal reasoning to disentangle individual contributions. Identify which specific actions by which agent led to the final outcome.

Output Format:

- **Credit Assignment [Agent i]:** [Detailed linguistic critique of Agent i ’s causal contribution.]

A.6 Qualitative Demonstrations and Case Studies

To provide a deeper understanding of how LangMARL facilitates coordination, we present qualitative case studies from Overcooked and Pistonball in Figure 7. These examples illustrate the transition from raw trajectory data to actionable language-based policy updates.

Prompt Template 3: Language Policy Gradient Estimator

System Role: You are a policy optimization specialist.

Inputs:

• **Language Credits:** [Critiques from the Centralized Critic for the specific agent.] • **Prior Policy:** [The agent's current text-based reasoning protocol.]

Instruction: Translate the credit signals into a "language gradient." Specify exactly how the policy should be modified to reinforce positive behaviors or correct failures.

Output Format:

• *Language Gradient:* [Directional instructions for policy improvement.]

Prompt Template 4: Language Policy Optimizer

System Role: You are a high-level strategy optimizer.

Inputs:

• **Aggregated Gradients:** [A set of language gradients derived from multiple trajectories $\{\tau_k\}$.] • **Prior Policy:** [The existing policy to be updated.]

Instruction: Synthesize the gradients across different scenarios. Identify consistent patterns, resolve conflicting feedback, and generate a refined, more robust version of the policy.

Output Format:

• *Updated Policy:* [The new comprehensive textual strategy for the agent.]

Overcooked. As shown in Fig. 1 and Fig. 7(a), the agents initially fail because Player Green prioritizes placing plates rather than supplying the onion needed to complete the soup, leaving Player Blue unable to finish cooking. A global reward signal only indicates "order failed," causing both agents to question their own behavior without identifying the true bottleneck (Fig. 1). In contrast, our Centralized Language Critic identifies the causal link: Player Green's misaligned resource prioritization is responsible for the delay. The resulting per-agent language credit specifically instructs Player Green to "prioritize understanding Player Blue's immediate cooking needs and adjust resource placement accordingly." As shown in the optimized policy (Fig. 7(a)), this targeted credit drives Player Green to proactively coordinate item delivery timing, resolving the bottleneck and improving overall kitchen efficiency.

Pistonball. Correcting Misaligned Actions. In the Pistonball task (Figure 7(b)), coordination often collapses when specific pistons remain elevated, blocking the ball's progress. The critic analyzes the trajectory to identify the causally responsible agent (the specific piston) and generates a "Language

Gradient" highlighting the negative impact of its timing. The policy is subsequently optimized to move down earlier, ensuring the ball's momentum is maintained.

These cases demonstrate that semantic credit assignment effectively improves policy optimization. By replacing trial-and-error with causal reasoning in natural language, LangMARL ensures that each agent understands its unique role and contribution within the team, which is the key to scaling to complex multi-agent environments.

A.7 AI usage

During the preparation of this work, the authors employed LLMs to assist with code implementation and to refine the manuscript's grammatical clarity and stylistic flow. All final content, technical contributions, and experimental analyses were rigorously reviewed and verified by the authors.