WHEN IS FURTHER COLLABORATION WORTHWHILE? COST-AWARE MULTI-AGENT REASONING VIA COUNTERFACTUAL BRANCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-Agent Debate (MAD) can enhance the reasoning capabilities of Large Language Models (LLMs), but additional collaboration rounds introduce substantial token overhead. Existing approaches typically make decisions at the instance level to determine whether collaboration should be initiated. However, they do not continuously reassess the need for further interaction as the reasoning state evolves. Consequently, they could incur redundant interactions on tasks that have already been resolved while providing insufficient refinement for those that still require further reasoning. To address this limitation, we introduce RLMAS, a cost-sensitive reinforcement learning framework for adaptive multi-agent reasoning. RLMAS augments the collaboration policy state with capability-aware confidence signals and trains a state-dependent policy using a reward that balances improvements in answer quality against incremental token costs. We introduce an offline counterfactual branching scheme to improve per-state action coverage beyond single-path trajectories by expanding alternative actions from shared reasoning states. The resulting feedback for different actions helps the policy learn when to stop and which refinement action to select. During deployment, the learned collaboration policy selects actions based on the current reasoning state without constructing counterfactual branches. Across diverse reasoning benchmarks and model scales, RLMAS reduces token usage by up to 92.6% while improving accuracy by up to 8.8% relative to baselines.

1 INTRODUCTION

Large Language Model (LLM) agents have demonstrated strong reasoning capabilities across a wide range of tasks, including code generation and complex problem solving Liu et al. (2024). However, conventional single-agent reasoning is often confined to a limited set of reasoning trajectories, restricting solution diversity and leaving latent errors insufficiently examined. These limitations have motivated growing interest in multi-agent systems across both academia and industry, as reflected by platforms such as Microsoft's AutoGen Wu et al. (2024) and Google's Agent Development Kit for coordinating specialized agents in complex workflows. Among various multi-agent reasoning paradigms, Multi-Agent Debate (MAD) enables multiple LLM agents to exchange complementary perspectives and iteratively critique and refine one another's reasoning. Such interaction can expose latent errors, correct flawed intermediate reasoning, and improve reasoning performance over single-agent inference Du et al. (2024).

Despite its potential to improve reasoning accuracy, MAD incurs substantial computational overhead because multi-round interaction requires repeated LLM generation and communication, often consuming several times more tokens than single-agent inference Zeng et al. (2025). More importantly, the benefit of additional collaboration is inherently instance-dependent and often exhibits diminishing returns as interaction proceeds Smit et al. (2024). Some instances may already be sufficiently resolved after only limited collaboration, whereas others may require additional rounds of refinement. Figure 1 provides an illustrative example of this phenomenon, showing that extending the debate beyond the point at which the problem has been effectively resolved does not necessarily provide further benefit. These observations motivate more selective and adaptive mechanisms that allocate collaboration according to the evolving needs of each instance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

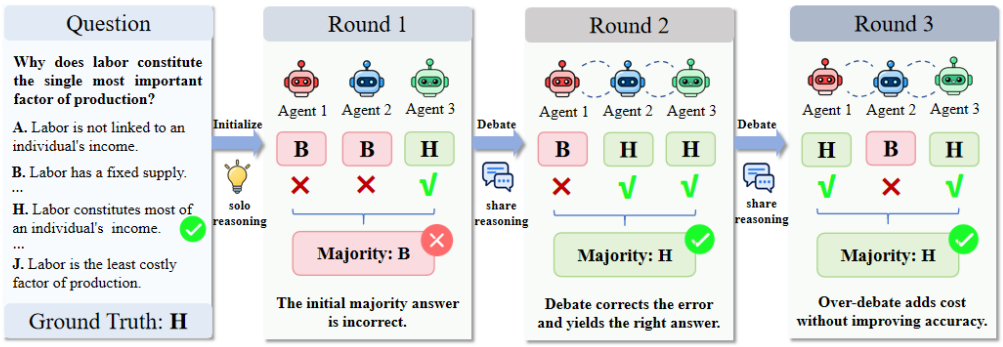

Figure 1: Diminishing returns of additional collaboration. The initially incorrect majority answer is corrected in Round 2. A further round changes individual responses but leaves the correct majority answer unchanged, incurring additional token cost without improving the final prediction.

Recent efforts to improve the efficiency of multi-agent reasoning seek to avoid unnecessary collaboration and reduce interaction overhead. Selective collaboration uses task complexity or initial-answer reliability signals to determine whether multi-agent reasoning is needed Kim et al. (2024); Fan et al. (2026). Once collaboration begins, communication sparsification removes redundant messages, links, or participants to limit repeated generation and information exchange Zeng et al. (2025); Wang et al. (2025b), while structure and workflow optimization reorganizes agent coordination through optimized communication topologies or structured review Zhuge et al. (2024); Wang et al. (2025a). However, deciding which instances require collaboration and making their interactions more efficient do not by themselves determine how collaboration should evolve as reasoning progresses. An instance that initially warrants extensive collaboration may become sufficiently resolved after limited interaction, while another may still require refinement. The initial need for collaboration therefore does not alone justify its continuation, and even an economical interaction may be unnecessary when little further improvement is expected.

Efficient multi-agent reasoning therefore requires more than selective collaboration: it requires deciding at each stage, whether and how additional collaboration is worth its cost. This perspective is consistent with recent findings that the benefit of additional test-time computation is strongly instance-dependent and that adaptive compute allocation can substantially improve inference efficiency Snell et al. (2025). Learning the value of further collaboration involves two fundamental challenges. First, the intermediate reasoning state of an LLM is difficult to quantify, making it unclear whether the current response has sufficient capability to terminate or still requires further refinement. Second, supervision is typically limited to sparse trajectory-level outcomes such as final-answer correctness, which reveal success but provide limited information about the contribution of individual intermediate decisions. Addressing these challenges requires both an informative representation of the model's reasoning state and finer-grained supervision for assigning value to alternative collaboration decisions.

To address these challenges, we propose RLMAS (Reinforcement Learning for Multi-Agent Systems), a cost-sensitive reinforcement learning framework for adaptive multi-agent collaboration. RLMAS learns a state-dependent policy over collaboration actions with different token costs, adaptively allocating reasoning effort to match each instance's complexity and evolving refinement needs. To support these decisions, we augment the policy state with capability-aware confidence features derived by discretizing token-level log probabilities. For offline training, we construct counterfactual multi-branch trajectories that expose the outcomes of alternative actions from shared reasoning states, addressing the limited per-state action coverage of single-path trajectories. The resulting branch returns supervise action-value learning under a reward that credits improvements in answer quality and penalizes incremental token consumption, allowing termination and continued refinement to be evaluated within a unified cost-sensitive objective. During online collaboration control, RLMAS executes only the selected action and reassesses the need for further collaboration after each step, without constructing counterfactual branches.

Our main contributions are summarized as follows: (1) We formulate efficient multi-agent collaboration as a cost-sensitive sequential decision problem and propose **RLMAS**, a reinforcement learning framework that dynamically determines whether and how collaboration should proceed by jointly considering reasoning utility and interaction cost. (2) We introduce a capability-aware state representation based on discretized token-level confidence signals, enabling the policy to better characterize the evolving reasoning state and make more informed collaboration decisions. (3) We construct counterfactual multi-branch trajectories that expose the returns of alternative collaboration actions from the same reasoning state, providing comparative supervision for more effective action-value learning and credit assignment under sparse outcome rewards. (4) Extensive experiments across diverse reasoning benchmarks and LLM scales demonstrate that RLMAS consistently improves the accuracy–efficiency trade-off, reducing token consumption by up to **92.6%** while maintaining or improving reasoning accuracy compared with competitive multi-agent reasoning baselines.

2 Related Work

Single-Agent Reasoning. Inference-time reasoning techniques have substantially improved the reasoning ability of LLMs. Chain-of-Thought (CoT) Wei et al. (2022) elicits intermediate reasoning steps, while Self-Consistency Wang et al. (2023) improves robustness by aggregating multiple reasoning trajectories. Tree-of-Thoughts (ToT) Yao et al. (2023) further extends linear reasoning into tree-structured search over candidate solutions. Despite their effectiveness, these methods remain confined to the perspective of a single model and may produce correlated reasoning errors. This limitation motivates multi-agent collaboration, where multiple agents provide complementary reasoning and external feedback.

Multi-Agent Collaborative Reasoning. MAD enables multiple LLM agents to iteratively exchange, critique, and refine their solutions. Representative approaches show that structured debate can improve factuality and reasoning by encouraging diverse perspectives and mutual correction Du et al. (2024); Liang et al. (2024). ReConcile Chen et al. (2024) promotes consensus among heterogeneous LLMs through multi-round discussion, while DMAD Liu et al. (2025) explicitly introduces diverse reasoning strategies across agents. However, such improvements often come with substantial computational overhead, as multiple agents repeatedly generate and exchange long reasoning trajectories. Consequently, token consumption and inference latency increase rapidly with the number of agents and interaction rounds Liu et al. (2026).

Efficient Multi-Agent Collaboration. Recent work has increasingly focused on improving the efficiency of multi-agent reasoning. GroupDebate Liu et al. (2026) reduces communication overhead through subgroup discussions, while OPTIMA Chen et al. (2025) optimizes the effectiveness–efficiency trade-off of agent interactions. iMAD Fan et al. (2026) selectively triggers debate only when it is likely to improve the initial response, thereby avoiding unnecessary multi-agent inference. MARS Wang et al. (2025a) instead adopts a hierarchical author–reviewer–meta-reviewer architecture, reducing expensive inter-agent communication while preserving external feedback. These approaches improve multi-agent reasoning efficiency by reducing redundant debate, restructuring communication, or optimizing interaction patterns.

3 Method

In this section, we introduce a cost-aware reinforcement learning framework for adaptive multi-agent collaboration. RLMAS is trained offline on counterfactual multi-branch trajectories, which reveal the outcomes of alternative collaboration actions originating from the same reasoning state. By encoding model capabilities in state representations and incorporating token costs into action-value learning, RLMAS learns to continue collaboration only when the expected improvement in reasoning quality outweighs the additional token cost.

3.1 System Overview

RLMAS assesses the expected benefits of collaboration at each reasoning state and adaptively orchestrates mechanisms with varying computational costs. Rather than applying a fixed interac-

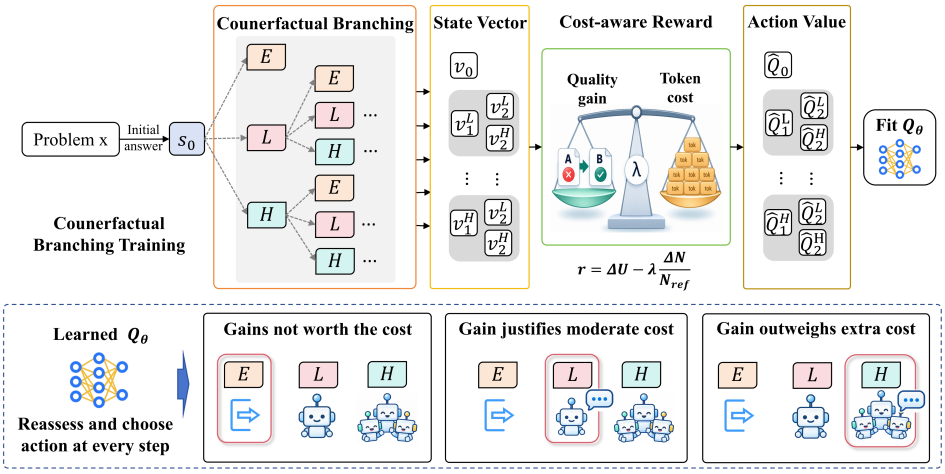

Figure 2: Overview of RLMAS. During offline controller training, RLMAS learns an action-value function from counterfactual branching trees, where branch outcomes are encoded as state vectors and evaluated using a cost-aware reward that balances answer-quality improvement against additional token cost. During online collaboration, the learned controller iteratively selects actions with different costs and terminates when no further collaboration is selected.

tion protocol, it treats the need for additional reasoning as a state-dependent decision that evolves throughout the reasoning process. This allows the framework to retain the current answer when further computation is unlikely to justify its cost, or to engage lightweight refinement or more intensive multi-agent collaboration when their anticipated benefits outweigh the associated overhead. By adjusting both the form and extent of collaboration, RLMAS aligns computational effort with potential reasoning gains, aiming to improve the accuracy–efficiency trade-off through more targeted interaction rather than uniformly increasing computation.

As illustrated in Figure 2, RLMAS comprises three core components: a counterfactual data generator, a state encoder, and a cost-sensitive value learning module. During offline training, the generator constructs counterfactual collaboration trajectories to capture the outcomes and computational costs of alternative actions. The state encoder summarizes the current reasoning status into a compact representation, while the recorded branch outcomes are used to derive rewards that balance answer quality and computation cost. These signals are then used to train a lightweight Q-network for collaboration control, with model and reward configurations selected on held-out validation data.

Counterfactual Branching Tree. For each training question x_i, RLMAS obtains an initial response and constructs a counterfactual branching tree:

$$\widehat{\mathcal{T}}_i = \left(s_{i,0}, \mathcal{A}, H, \widehat{\mathcal{V}}_i, \widehat{\mathcal{E}}_i\right), \tag{1}$$

where $s_{i,0}$ is the initial reasoning state, H is the maximum decision horizon, and $\widehat{\mathcal{V}}_i = \bigcup_{t=0}^{H} \widehat{\mathcal{V}}_{i,t}$ and $\widehat{\mathcal{E}}_i$ are the node and transition sets, respectively. Nodes at depth H are terminal. The action space is $\mathcal{A} = \{\mathrm{E}, \mathrm{L}, \mathrm{H}\}$: E terminates interaction and returns the current answer with zero incremental reward and cost; L invokes lightweight collaboration through self-refine; and H invokes high-cost collaboration through team review.

Suppressing the question index i, the generator expands every legal action $a \in \mathcal{A}$ at each nonterminal state, thereby constructing counterfactual successor branches that expose the outcomes of alternative collaboration decisions:

$$s_{t+1} = F_t(s_t, a_t), \tag{2}$$

where F_t is the successor mapping on the constructed tree. Each transition records the resulting answer, confidence, correctness, and incremental token cost. The capability-aware encoder represents each nonterminal state as $\mathbf{v}_t = \phi(s_t)$, summarizing observable confidence, reasoning progress, task characteristics, and token usage. Nonterminal successors are encoded as $\mathbf{v}_{t+1}^{a_t} = \phi(s_{t+1})$.

Backward dynamic programming computes action-value targets $\widehat{Q}_t(s,a)$, rewarding corrections while penalizing harmful revisions and incremental token costs. These targets train a lightweight Q-network, $Q_{\theta,t}(s,a) := q_\theta(\phi(s),a)$, while the underlying LLMs remain fixed.

Counterfactual Fitted Q-Learning. Single-path interaction data reveal only the outcome of the selected action, limiting comparisons among alternative collaboration decisions from the same reasoning state. RLMAS addresses this limitation by constructing a finite counterfactual branching tree $\widehat{\mathcal{T}}_i$ for each training question x_i. Each history- and stage-specific nonterminal state is expanded over its legal actions, with each branch recording the resulting response, answer correctness, capability-related signals, and incremental token cost.

Action-value targets are computed by backward recursion:

$$\widehat{Q}_t(s,a) = \begin{cases} 0, & a = \mathrm{E}, \\ r_t(s,a) + \widehat{V}_{t+1}(F_t(s,a)), & a \neq \mathrm{E}, \end{cases} \tag{3}$$

where $r_t(s,a)$ balances changes in answer quality against incremental token costs, and $\widehat{V}_t(s) = \max_{a\in\mathcal{A}_t(s)} \widehat{Q}_t(s,a)$ for nonterminal states, with zero value at terminal states. Under complete branch coverage and consistent action masks, these targets equal the optimal action values on the fixed counterfactual tree (Theorem 2). The targets remain fixed during critic training, reducing approximation error associated with learned bootstrap values.

Capability-Aware State Representation. To assess whether further collaboration is worthwhile, RLMAS encodes the current reasoning state using only signals available during online collaboration control. For a response o of length $L(o)$ with token log-probabilities $\ell_1,\ldots,\ell_{L(o)}$, we define its sequence-level confidence as

$$c(o) = \exp\left(\frac{1}{L(o)}\sum_{j=1}^{L(o)} \ell_j\right). \tag{4}$$

We retain both the continuous confidence score and its quantile-based bucket encoding to capture fine-grained variation and coarse reliability levels.

At decision stage t, the encoder maps the reasoning state s_t to a compact state vector:

$$\mathbf{v}_t = \phi(s_t) \in \mathbb{R}^{d_{\mathrm{feat}}}, \tag{5}$$

where d_{feat} is the feature dimension. The encoded features comprise the initial, current, and review-response confidences $c_0, c_t, c_t^{\mathrm{rev}}$; an indicator d_t of whether the latest collaboration step changed the selected answer; normalized cumulative token usage ρ_t; decision progress t/H; collaboration history $\mathbf{h}_t$; and domain information $e(x)$. These features characterize answer reliability and reasoning progress without including reference answers or correctness labels, supporting state-dependent collaboration decisions.

Cost-Aware Reward. RLMAS balances improvements in answer quality against the token cost of additional collaboration. Let y_t and $y^\star$ denote the current and reference answers, respectively, and define the correctness indicator $z_t = \mathbb{I}[y_t = y^\star]$. For a non-exit action $a_t \in \mathcal{A}_t(s_t)\setminus\{\mathrm{E}\}$, the incremental reward is

$$r_t = \underbrace{\mathbb{I}[z_t=0, z_{t+1}=1] - \eta\,\mathbb{I}[z_t=1, z_{t+1}=0]}_{\text{quality change}} - \underbrace{\lambda\frac{\Delta N_t}{N_{\mathrm{ref}}}}_{\text{incremental token cost}}, \tag{6}$$

where the quality term assigns a positive reward when collaboration corrects an initially incorrect answer and a penalty of magnitude $\eta > 0$ when it overturns a correct answer. Transitions that leave answer correctness unchanged receive no quality reward. The second term penalizes the additional computation introduced by the selected action, where ΔN_t denotes its incremental token usage, N_{ref} is a normalization constant used to keep the cost term on a comparable numerical scale, and $\lambda \geq 0$ controls the trade-off between answer improvement and computational efficiency.

This reward therefore encourages the policy to invoke additional collaboration only when its expected improvement in answer quality is sufficient to justify the associated cost, while discouraging both unnecessary interaction and actions that degrade an already correct answer.

Offline Training and Online Deployment. Counterfactual branching is performed only during offline training, where RLMAS caches alternative action outcomes and uses them to construct full action-value targets. At deployment, the controller observes the current state, selects the legal action with the highest predicted Q-value, and executes only that action. It repeats this process until termination or the decision horizon is reached, without constructing a counterfactual tree or accessing reference answers.

4 EXPERIMENTS

In this section, we evaluate RLMAS across multiple benchmarks. We first introduce the experimental setup, followed by the main results, ablation studies, and an analysis of the learned collaboration policy. Our experiments address three key questions: (1) Can RLMAS achieve a better accuracy–cost trade-off than existing single-agent and multi-agent reasoning methods? (2) Can RLMAS correct erroneous responses while preserving correct ones? (3) How do capability-aware state modeling, counterfactual supervision contribute to RLMAS performance?

4.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate RLMAS on three challenging reasoning benchmarks with increasing difficulty: **MATH** Hendrycks et al. (2021), **MMLU-Pro** Wang et al. (2024), and **SuperGPQA** Team et al.. MATH focuses on multi-step mathematical problem solving across a range of subjects and difficulty levels. MMLU-Pro evaluates broad knowledge and reasoning ability across diverse academic domains. SuperGPQA provides the most challenging setting, assessing expert-level, knowledge-intensive reasoning over a wide range of disciplines.

Models. We evaluate RLMAS with LLMs of different scales and capabilities to examine whether the learned collaboration mechanism generalizes across varying model strengths. Specifically, we use Qwen3-4B Yang et al. (2025), GPT-OSS-20B Agarwal et al. (2025), and DeepSeek-v4-Flash Xu et al. (2026) as the underlying LLMs. Unless otherwise specified, DeepSeek-v4-Flash is used as the default agent model in our experiments.

Baselines. We compare RLMAS with representative single-agent and multi-agent reasoning methods. **Single** directly returns the initial response without additional inference, serving as the minimum-compute baseline, while **CoT** Wei et al. (2022) elicits explicit intermediate reasoning within a single model. For multi-agent reasoning, **MAD** Du et al. (2024) follows a fixed k-round debate protocol, providing a standard non-adaptive collaboration baseline. We further compare with recent efficiency-oriented methods: **iMAD** Fan et al. (2026) selectively determines whether multi-agent debate should be triggered, **MARS** Wang et al. (2025a) reduces communication overhead through a hierarchical author–reviewer–meta-reviewer architecture, and **AgentDropout** Wang et al. (2025b) dynamically prunes redundant agents and communication links by optimizing round-specific communication topologies.

4.2 MAIN RESULTS

As shown in Table 1, we compare RLMAS with a range of baselines across three benchmarks and multiple LLMs. Full MAD generally achieves stronger accuracy at substantially higher inference cost, whereas RLMAS retains most of the performance gains with significantly fewer tokens, yielding a more favorable accuracy–efficiency trade-off in most settings.

Observation 1. MAD often improves accuracy, but incurs substantial token cost. As shown in Table 1, MAD improves over Single in eight of nine dataset–model settings, but achieves the highest accuracy in only four, including ties, while consuming 14.1–29.3$\times$ more tokens. With DeepSeek-v4-Flash, for example, MAD improves accuracy from 75.4% to 84.4% on MMLU-Pro, 58.4% to 61.0% on SuperGPQA, and 94.6% to 97.6% on MATH, while increasing token usage from 0.55K, 0.46K, and 0.66K to 12.29K, 11.00K, and 17.52K, respectively. Moreover, on SuperGPQA with GPT-OSS-20B, accuracy decreases from 33.6% to 33.0% despite token usage rising from 0.54K to 7.63K. Thus, additional multi-agent interaction incurs substantial overhead without consistently improving accuracy.

Table 1: Main results on MATH, MMLU-Pro, and SuperGPQA. **Acc.** (%) and **Tok.** (10^3) denote accuracy and average generated tokens per instance, respectively. Auxiliary values for AgentDropout, iMAD, MARS, and RLMAS indicate accuracy changes over Single (p.p.) and token changes over MAD (%). ↑/↓ indicates higher/lower is better. **Bold** and underline mark the best and second-best accuracy, respectively.

	Qwen3-4B		GPT-OSS-20B		DeepSeek-v4-Flash	
Method	Acc. (%) ↑	Tok. (10^3) ↓	Acc. (%) ↑	Tok. (10^3) ↓	Acc. (%) ↑	Tok. (10^3) ↓
MATH						
Single	80.2	0.90	85.6	0.78	94.6	0.66
CoT	81.2	0.88	78.6	0.80	94.6	0.69
MAD	**85.0**	26.34	**94.0**	14.67	**97.6**	17.52
AgentDropout	77.4 (−2.8)	10.39 (−60.6%)	82.0 (−3.6)	6.99 (−52.3%)	94.4 (−0.2)	8.75 (−50.1%)
iMAD	74.6 (−5.6)	11.68 (−55.7%)	87.8 (+2.2)	4.37 (−70.2%)	95.2 (+0.6)	3.58 (−79.6%)
MARS	<u>82.4</u> (+2.2)	15.07 (−42.8%)	<u>89.8</u> (+4.2)	7.44 (−49.3%)	96.0 (+1.4)	13.84 (−21.0%)
RLMAS	**85.0** (+4.8)	1.96 (−92.6%)	89.0 (+3.4)	2.49 (−83.0%)	<u>96.2</u> (+1.6)	1.90 (−89.1%)
MMLU-Pro						
Single	56.6	0.73	64.4	0.63	75.4	0.55
CoT	<u>59.0</u>	1.05	67.6	0.67	81.4	0.69
MAD	58.4	14.61	68.0	10.02	**84.4**	12.29
AgentDropout	53.0 (−3.6)	6.23 (−57.4%)	70.0 (+5.6)	4.84 (−51.7%)	**84.4** (+9.0)	6.60 (−46.3%)
iMAD	51.4 (−5.2)	13.57 (−7.1%)	71.2 (+6.8)	3.63 (−63.8%)	82.4 (+7.0)	2.46 (−80.0%)
MARS	56.0 (−0.6)	8.38 (−42.7%)	<u>71.8</u> (+7.4)	7.34 (−26.7%)	83.2 (+7.8)	8.89 (−27.7%)
RLMAS	**62.2** (+5.6)	9.05 (−38.1%)	**72.2** (+7.8)	1.38 (−86.2%)	<u>84.2</u> (+8.8)	2.26 (−81.6%)
SuperGPQA						
Single	26.6	0.62	33.6	0.54	58.4	0.46
CoT	26.8	0.87	33.4	0.62	60.0	0.67
MAD	<u>31.2</u>	12.34	33.0	7.63	61.0	11.00
AgentDropout	26.6 (+0.0)	4.98 (−59.6%)	33.2 (−0.4)	4.32 (−43.4%)	<u>62.2</u> (+3.8)	6.47 (−41.2%)
iMAD	29.2 (+2.6)	14.67 (+18.9%)	35.6 (+2.0)	4.90 (−35.8%)	61.6 (+3.2)	5.18 (−52.9%)
MARS	28.2 (+1.6)	7.31 (−40.8%)	<u>36.2</u> (+2.6)	6.68 (−12.5%)	60.2 (+1.8)	8.61 (−21.7%)
RLMAS	**31.4** (+4.8)	3.85 (−68.8%)	**36.6** (+3.0)	1.92 (−74.8%)	**66.2** (+7.8)	1.47 (−86.6%)

Observation 2. RLMAS achieves a favorable accuracy–efficiency trade-off. RLMAS reduces token consumption relative to MAD by 38.1–92.6% across all nine settings while matching or exceeding its accuracy in six. On MATH with Qwen3-4B, it matches MAD at 85.0% accuracy while reducing token usage from 26.34K to 1.96K (92.6%). With DeepSeek-v4-Flash, RLMAS reduces tokens by 81.6%, 86.6%, and 89.1% on MMLU-Pro, SuperGPQA, and MATH, respectively, with accuracy changes of -0.2, $+5.2$, and -1.4 percentage points. With GPT-OSS-20B, the corresponding token reductions are 86.2%, 74.8%, and 83.0%, with accuracy changes of $+4.2$, $+3.6$, and -5.0 points. RLMAS also improves over Single in all nine settings, including gains of 8.8 and 7.8 points on MMLU-Pro with DeepSeek-v4-Flash and GPT-OSS-20B. These results show that RLMAS substantially reduces collaboration cost, although it does not preserve MAD's accuracy in every setting.

Observation 3. The accuracy–cost trade-off varies across agent models and tasks. RLMAS achieves higher accuracy with fewer tokens than iMAD in all nine settings, and than MARS and AgentDropout in seven settings each. With DeepSeek-v4-Flash, it outperforms both iMAD and MARS in accuracy and token cost across all three benchmarks; on SuperGPQA, for example, RLMAS achieves 66.2% accuracy with 1.47K tokens, versus 61.6%/5.18K for iMAD and 60.2%/8.61K for MARS. With GPT-OSS-20B, it similarly outperforms both baselines on MMLU-Pro and SuperGPQA, while on MATH it uses 66.6% fewer tokens than MARS with a small accuracy decrease from 89.8% to 89.0%. For Qwen3-4B, savings relative to MAD vary from 38.1% on MMLU-Pro to 68.8% on SuperGPQA and 92.6% on MATH; on MMLU-Pro, RLMAS is more accurate than MARS (62.2% vs. 56.0%) but uses more tokens (9.05K vs. 8.38K). These results demonstrate the advantage of RLMAS over other methods in achieving a better accuracy–cost trade-off.

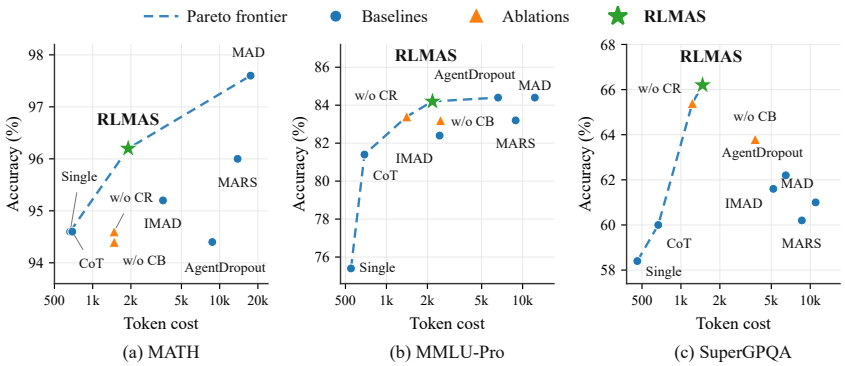

Figure 3: Accuracy–token trade-offs on (a) MATH, (b) MMLU-Pro, and (c) SuperGPQA. Dashed lines connect the Pareto-optimal points among the displayed methods, favoring higher accuracy and lower token cost. Stars, circles, and triangles denote RLMAS, baselines, and ablations, respectively. Token cost is shown on a logarithmic scale.

4.3 Ablation Study

We evaluate the contributions of confidence representations and counterfactual branching on MATH, MMLU-Pro, and SuperGPQA using DeepSeek-Flash as the agent model. *w/o Confidence Representation (CR)* removes the proposed confidence signals from the policy state. *w/o Counterfactual Branching (CB)* instead trains the collaboration policy with standard DQN on single-path trajectories, without expanding alternative collaboration actions from shared reasoning states. All other components are kept unchanged for controlled comparison. Table 2 reports the results.

Table 2: Ablation study of RLMAS on MATH, MMLU-Pro, and SuperGPQA. **Acc.** (%) and **Tok.** (10^3) denote accuracy and average generated tokens per instance, respectively. **Bold** and underline mark the best and second-best distinct values in each column, respectively.

	MATH		MMLU-Pro		SuperGPQA	
Variant	Acc. (%) ↑	Tok. ↓	Acc. (%) ↑	Tok. ↓	Acc. (%) ↑	Tok. ↓
RLMAS (Full)	**96.20**	1.90	**84.20**	2.26	**66.20**	1.47
w/o CR	<u>94.60</u>	1.48	<u>83.40</u>	1.41	<u>65.40</u>	1.23
w/o CB	94.40	1.48	83.20	2.50	63.80	3.75

Compared with w/o CR, RLMAS improves accuracy by 1.60, 0.80, and 0.80 percentage points on MATH, MMLU-Pro, and SuperGPQA, respectively. Although removing confidence representations reduces token usage across all three datasets, these savings are accompanied by lower accuracy. The results support the usefulness of confidence signals for collaboration control, while showing that the resulting accuracy gains come with additional computational cost.

Compared with w/o CB, RLMAS improves accuracy by 1.80, 1.00, and 2.40 percentage points on MATH, MMLU-Pro, and SuperGPQA, respectively. It also reduces token usage from 2.50K to 2.26K on MMLU-Pro and from 3.75K to 1.47K on SuperGPQA. Although w/o CB uses fewer tokens on MATH, this saving comes at the expense of lower accuracy. These results suggest that feedback on alternative actions from shared reasoning states helps the policy allocate collaboration more effectively than training on single-path trajectories, supporting better accuracy–efficiency trade-offs.

4.4 More Analyses

Pareto Frontier Across All Methods As shown in Figure 3, RLMAS lies on the empirical Pareto frontier across all three datasets among the methods shown. On MATH, RLMAS achieves 96.2% accuracy using approximately 1,904 tokens, reducing token consumption by 89.1% relative to MAD at the cost of 1.4 percentage points in accuracy. On MMLU-Pro, it achieves 84.2% accuracy with

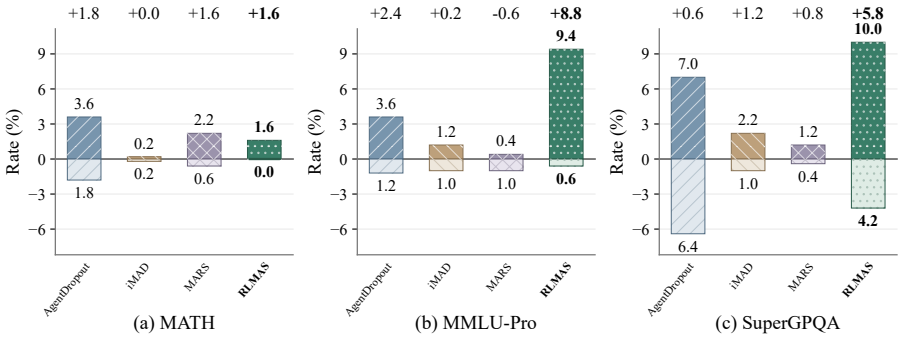

Figure 4: Correction and destruction rates on (a) MATH, (b) MMLU-Pro, and (c) SuperGPQA. Correction rates are plotted above zero, while destruction rates are plotted below zero with lighter fills. Bar labels report the magnitudes of the corresponding rates in percent. Numbers above each panel indicate correction minus destruction in percentage points. Colors and patterns distinguish methods, with RLMAS highlighted in green and its results shown in bold.

approximately 2.26K tokens, requiring 81.6% fewer tokens than MAD while trailing its accuracy by only 0.2 percentage points. On SuperGPQA, RLMAS improves accuracy over MAD by 5.2 percentage points, reaching 66.2%, while reducing token consumption by 86.6%. RLMAS also achieves higher accuracy at lower token cost than both IMAD and MARS on every dataset.

Error correction and answer preservation. Figure 4 compares the correction and destruction rates across the three datasets. Since the methods use different initialization procedures, these rates are computed by comparing each method's final predictions with its own initial answers. Correction and destruction correspond to incorrect-to-correct and correct-to-incorrect transitions, respectively. On MMLU-Pro, RLMAS achieves the highest correction rate (9.4%) and the lowest destruction rate (0.6%) among the compared methods, yielding a net correction of 8.8 percentage points. On SuperGPQA, it attains a 10.0% correction rate and a 4.2% destruction rate, resulting in a net correction of 5.8 percentage points. These net corrections exceed the best corresponding baseline values by 6.4 and 4.6 percentage points, respectively. On MATH, RLMAS achieves a 1.6% correction rate with no observed destruction. Overall, RLMAS provides a favorable balance between correcting erroneous responses and preserving correct ones, with particularly strong net correction relative to its own initialization on MMLU-Pro and SuperGPQA.

5 Conclusion

In this work, we focus on improving the efficiency of multi-agent reasoning through adaptive compute allocation. We find that additional collaboration does not always yield proportional reasoning gains and can introduce substantial inference overhead. To address this problem, we propose **RLMAS**, a cost-aware reinforcement learning framework that formulates multi-agent collaboration as a sequential decision process. RLMAS uses capability-aware confidence signals to model the current reasoning state and constructs counterfactual multi-branch trajectories to evaluate alternative collaboration actions. Combined with cost-aware value learning, RLMAS dynamically determines whether further collaboration is necessary and which action should be taken. Extensive experiments across multiple reasoning benchmarks and agent models demonstrate that RLMAS retains most of the performance gains of multi-agent reasoning while significantly reducing token consumption. Our results suggest that effective multi-agent reasoning requires not more interaction, but more selective and state-dependent collaboration.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Justin Chen, Swarnadeep Saha, and Mohit Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7066–7085. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.381.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11534–11557. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.601.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR, 2024.

Wei Fan, JinYi Yoon, and Bo Ji. iMAD: Intelligent multi-agent debate for efficient and accurate LLM inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 29403–29411, 2026. doi: 10.1609/aaai.v40i35.40181.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik S Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae W Park. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452, 2024.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.992.

Tongxuan Liu, Xingyu Wang, Weizhe Huang, Wenjiang Xu, Yuting Zeng, Lei Jiang, Hailong Yang, and Jing Li. Groupdebate: Enhancing the efficiency of multi-agent debate using group discussion. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '26, pp. 1147–1155, Richland, SC, 2026. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400723179. doi: 10.65109/DSDX8860.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Yexiang Liu, Jie Cao, Zekun Li, Ran He, and Tieniu Tan. Breaking mental set to improve reasoning through diverse multi-agent debate. In *International Conference on Learning Representations*, volume 2025, pp. 24666–24725, 2025.

Andries Smit, Nathan Grinsztajn, Paul Duckworth, Thomas D. Barrett, and Arnu Pretorius. Should we be going mad? a look at multi-agent debate strategies for llms. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025.

P Team, X Du, Y Yao, K Ma, B Wang, T Zheng, K Zhu, M Liu, Y Liang, X Jin, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines, 2025. *URL https://arxiv.org/abs/2502.14739*.

Xiao Wang, Jia Wang, Yijie Wang, Pengtao Dang, Sha Cao, and Chi Zhang. Mars: toward more efficient multi-agent collaboration for llm reasoning. *arXiv preprint arXiv:2509.20502*, 2025a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. AgentDropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035. Association for Computational Linguistics, 2025b. doi: 10.18653/v1/2025.acl-long.1170.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

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. Autogen: Enabling next-gen LLM applications via multi-agent conversation. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL `https://openreview.net/forum?id=uAjxFFing2`.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023.

Yuting Zeng, Weizhe Huang, Lei Jiang, Tongxuan Liu, XiTai Jin, Chen Tianying Tiana, Jing Li, and Xiaohua Xu. S^2-MAD: Breaking the token barrier to enhance multi-agent debate efficiency. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9393–9408. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.475.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

A Appendix Overview

Appendix B analyzes counterfactual fitted Q-learning, establishing the exactness of dynamic-programming targets and a bound relating critic approximation error to the oracle gap on fixed cached trees. It also presents empirical diagnostics of target-construction error, critic approximation error, and the oracle-gap bound. Appendix C describes the benchmarks and data splits, model configurations and generation settings, and the controller architecture and optimization procedure. The AI Use Statement is provided at the end of the appendices.

B Theoretical Analysis of Counterfactual Fitted Q-Learning

This appendix analyzes counterfactual branching for the finite-horizon collaboration controller in RLMAS. We establish exact dynamic-programming targets under complete branching and a bound relating critic approximation error to policy suboptimality on fixed counterfactual trees.

B.1 Counterfactual Branching Trees and Assumptions

We consider an undiscounted finite-horizon decision problem ($\gamma = 1$) with at most H decision stages, indexed by $t = 0, \dots, H-1$. The initial response is generated before the first decision at $t = 0$, and nodes at depth H are terminal. The RLMAS setting analyzed here uses $H = 2$. The underlying reasoning state space $\mathcal{S}$ need not be finite; the analysis concerns a finite collection of finite counterfactual branching trees.

For each training question x_i, let $\widehat{\mathcal{T}}_i$ denote the counterfactual branching tree constructed from the initial response and the outcomes of legal collaboration actions. The analysis is conditional on these realized branch outcomes. A state $s \in \mathcal{S}$ is a history-labeled tree node together with its decision stage, and its encoded state vector is $\mathbf{v} = \phi(s)$. States induced by distinct interaction histories remain distinct even when their state vectors coincide.

The global action space is

$$\mathcal{A} = \{\mathrm{E}, \mathrm{L}, \mathrm{H}\}, \tag{7}$$

where E denotes immediate exit, which terminates interaction and returns the current answer; L invokes lightweight collaboration through *Self-Refine*; and H invokes high-cost collaboration through *Team Review*, in which multiple agents review and refine the current solution.

At a nonterminal state s and decision stage t, the legal action set is a nonempty subset of the global action space:

$$\varnothing \neq \mathcal{A}_t(s) \subseteq \mathcal{A}. \tag{8}$$

We write $\widehat{\mathcal{S}}_t$ for the disjoint union of all stage-t nonterminal nodes across the counterfactual trees.

On each fixed tree, every legal state–action pair has a specified reward $r_t(s,a)$ and successor $F_t(s,a)$. For the exit action E, whenever legal, we define $r_t(s,\mathrm{E}) = 0$ and $\Delta N_t(s,\mathrm{E}) = 0$, and let $F_t(s,\mathrm{E})$ be a terminal node that retains the current answer. For a non-exit action, the successor retains the answer produced by that action. Every successor at the final decision stage $t = H-1$ is terminal.

For a fixed question, let $y(s)$ denote the answer associated with state s and $y^\star$ the reference answer. Define

$$z(s) = \mathbb{I}[y(s) = y^\star] \in \{0,1\}. \tag{9}$$

For each legal non-exit action $a \in \mathcal{A}_t(s) \setminus \{\mathrm{E}\}$, let $s' = F_t(s,a)$. The incremental reward is

$$\begin{aligned} r_t(s,a) = \mathbb{I}[z(s)=0,\, z(s')=1] \\ - \eta\, \mathbb{I}[z(s)=1,\, z(s')=0] \\ - \lambda \frac{\Delta N_t(s,a)}{N_{\mathrm{ref}}}, \end{aligned} \tag{10}$$

where $\Delta N_t(s,a) \geq 0$ denotes the total number of tokens consumed by all model calls triggered by action a, $N_{\mathrm{ref}} > 0$ is the reference token scale, $\lambda \geq 0$ controls the quality–cost trade-off, and $\eta \geq 1$ penalizes revisions that turn a correct answer into an incorrect one.

The quality term assigns $+1$ to answer corrections, $-\eta$ to revisions that destroy a correct answer, and 0 when correctness is unchanged. Every non-exit action additionally incurs its incremental token penalty, while the initial response cost is treated as sunk. Terminal nodes retain their resulting answers, so $z(s')$ remains well defined when a collaboration action reaches a terminal node, including at the final decision stage. Reference answers and correctness labels are used for offline reward construction and evaluation but are excluded from controller inputs. All rewards on the constructed trees are assumed to be finite.

Assumption 1 (Complete branching and action-set consistency)**.** At every nonterminal node, a reward and successor are available for every legal action, and each nonterminal successor is recursively expanded through the remaining decision stages. Each branch is evaluated using its corresponding interaction history; an exit transition may be represented analytically. Tree construction, critic training, offline evaluation, and online collaboration control use the same legal action masks, horizon, and termination rules. Offline reward computation, target construction, and policy evaluation consistently use the reward definition in equation 10, with zero reward for exit. Online collaboration control does not require reference answers or correctness-based reward computation.

This assumption concerns coverage of all legal action paths rather than repeated independent sampling of each action. A single realized outcome per legal non-exit state–action pair, together with the analytically defined exit transitions, suffices to define a fixed counterfactual tree. The resulting guarantees apply to these trees and do not, by themselves, establish exact action values for the underlying stochastic collaboration process.

B.2 Exact Dynamic-Programming Targets

Fix a counterfactual branching tree $\widehat{\mathcal{T}}_i$ and suppress its index in the value functions below. Decision stages are indexed by $t = 0, \ldots, H-1$, and every terminal node, including all nodes at depth H, has zero continuation value. Rewards follow equation 10, with action costs measured by incremental token usage.

Define the state-value targets as

$$\widehat{V}_t(s) = \begin{cases} \max\limits_{a \in \mathcal{A}_t(s)} \widehat{Q}_t(s,a), & s \text{ is nonterminal}, \\ 0, & s \text{ is terminal}. \end{cases} \tag{11}$$

For each nonterminal state s at stage t and legal action $a \in \mathcal{A}_t(s)$, the action-value targets are computed by backward recursion:

$$\widehat{Q}_t(s,a) = \begin{cases} 0, & a = \mathrm{E}, \\ r_t(s,a) + \widehat{V}_{t+1}(F_t(s,a)), & a \neq \mathrm{E}. \end{cases} \tag{12}$$

The first case applies whenever $\mathrm{E} \in \mathcal{A}_t(s)$. For a legal non-exit action, let $s' = F_t(s,a)$. If s' is nonterminal, the recursion becomes

$$\widehat{Q}_t(s,a) = r_t(s,a) + \max_{a' \in \mathcal{A}_{t+1}(s')} \widehat{Q}_{t+1}(s',a'). \tag{13}$$

If s' is terminal, it reduces to

$$\widehat{Q}_t(s,a) = r_t(s,a). \tag{14}$$

For the $H = 2$ setting used in RLMAS, non-exit targets at the final decision stage $t = 1$ equal their immediate rewards. At $t = 0$, they additionally include the best continuation value whenever the successor is nonterminal. Exit targets are zero at both stages whenever exit is legal.

Let $\widehat{Q}_t^\star(s,a)$ and $\widehat{V}_t^\star(s)$ denote the optimal action- and state-value functions on the same fixed counterfactual tree, defined by cumulative incremental rewards over the remaining decision stages. At nonterminal states,

$$\widehat{V}_t^\star(s) = \max_{a \in \mathcal{A}_t(s)} \widehat{Q}_t^\star(s,a), \tag{15}$$

with $\widehat{V}_t^\star(s) = 0$ at terminal states.

Theorem 2 (Exactness of counterfactual dynamic-programming targets)**.** *Under Assumption 1, for every fixed counterfactual branching tree $\widehat{\mathcal{T}}_i$, every stage-t nonterminal state s, and every legal action $a \in \mathcal{A}_t(s)$,*

$$\widehat{Q}_t(s,a) = \widehat{Q}^\star_t(s,a), \qquad t = 0, \dots, H-1. \tag{16}$$

Consequently, the targets computed by equation 12 contain no approximation error from learned bootstrap values.

Proof. We prove the result by backward induction over the decision stages.

At the final decision stage $t = H-1$, every successor is terminal and therefore has zero continuation value. For each legal non-exit action, its optimal action value and dynamic-programming target both equal its immediate reward. For exit, whenever legal, both are zero because $r_{H-1}(s, \mathrm{E}) = 0$. Thus, for every $a \in \mathcal{A}_{H-1}(s)$,

$$\widehat{Q}^\star_{H-1}(s,a) = r_{H-1}(s,a) = \widehat{Q}_{H-1}(s,a). \tag{17}$$

Now suppose that

$$\widehat{Q}_{t+1}(s',a') = \widehat{Q}^\star_{t+1}(s',a') \tag{18}$$

for every stage-$(t+1)$ nonterminal state s' and every $a' \in \mathcal{A}_{t+1}(s')$. Consider a nonterminal state s at stage t.

If $\mathrm{E} \in \mathcal{A}_t(s)$, exit has zero incremental reward and zero continuation value, so

$$\widehat{Q}^\star_t(s, \mathrm{E}) = 0 = \widehat{Q}_t(s, \mathrm{E}). \tag{19}$$

For any legal non-exit action a, let $s' = F_t(s,a)$. Assumption 1 ensures that the tree specifies both $r_t(s,a)$ and s'.

If s' is terminal, its continuation value is zero, yielding

$$\widehat{Q}^\star_t(s,a) = r_t(s,a) = \widehat{Q}_t(s,a). \tag{20}$$

If s' is nonterminal, complete branching ensures that every legal continuation action in $\mathcal{A}_{t+1}(s')$ is represented in the tree. By the induction hypothesis,

$$\widehat{V}^\star_{t+1}(s') = \max_{a' \in \mathcal{A}_{t+1}(s')} \widehat{Q}^\star_{t+1}(s',a') \tag{21}$$

$$= \max_{a' \in \mathcal{A}_{t+1}(s')} \widehat{Q}_{t+1}(s',a') \tag{22}$$

$$= \widehat{V}_{t+1}(s'). \tag{23}$$

The Bellman recursion on the fixed tree then gives

$$\widehat{Q}^\star_t(s,a) = r_t(s,a) + \widehat{V}^\star_{t+1}(s') \tag{24}$$

$$= r_t(s,a) + \widehat{V}_{t+1}(s') \tag{25}$$

$$= \widehat{Q}_t(s,a). \tag{26}$$

Thus, equation 16 holds at stage t. Backward induction establishes the result at every decision stage of the fixed counterfactual tree. □

Comparison with Standard DQN. Under the same undiscounted convention, let $s' = F_t(s,a)$. At training update k, a standard DQN regression target uses a learned target network:

$$b^{\mathrm{DQN}}_{t,k}(s,a) = \begin{cases} r_t(s,a), & s' \text{ is terminal}, \\ r_t(s,a) + \max\limits_{a' \in \mathcal{A}_{t+1}(s')} Q_{\theta^-_k, t+1}(s',a'), & s' \text{ is nonterminal}. \end{cases} \tag{27}$$

Here, k indexes training updates, θ^-_k denotes the target-network parameters at update k, and $Q_{\theta^-_k, t+1}$ estimates continuation action values at decision stage $t+1$. Because this continuation estimate is

learned, the target of a transition with a nonterminal successor may change when the target network is updated.

In contrast, RLMAS uses the fixed regression target $b_{t,k}(s,a) = \widehat{Q}_t(s,a)$ at every training update. By Theorem 2, its continuation values are computed exactly on the constructed counterfactual tree rather than approximated by a learned network. This removes learned-bootstrap approximation from target construction, but does not imply an error-free learned critic or exact expected action values in the underlying stochastic collaboration process.

Empirical evaluation of Q-value estimation. We examine target-construction error and critic approximation error separately, using fixed counterfactual trees sampled with DeepSeek Flash on MMLU-Pro, SuperGPQA, and MATH. For each training seed, we evaluate all legal state–action pairs in the same held-out validation trees, containing 100 questions. All methods use the same source caches, critic architecture, and optimizer-update budget, with $H = 2$ and $\gamma = 1$. However, RLMAS supervises all legal actions, whereas the baselines supervise observed actions; action-label budgets and state-visitation distributions are therefore not matched.

Let $\mathcal{D}_{\mathrm{val}}$ denote the collection of legal validation tuples (h, s, a), and let $\widehat{Q}_h(s,a)$ be the exact backward-DP value on the fixed cached trees. For each seed and checkpoint t, we measure

$$\mathrm{TargetRMSE}_t = \left[\frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_{(h,s,a)\in\mathcal{D}_{\mathrm{val}}} \Big(y_{h,t}(s,a) - \widehat{Q}_h(s,a) \Big)^2 \right]^{1/2}, \tag{28}$$

$$\varepsilon_t = \max_{(h,s,a)\in\mathcal{D}_{\mathrm{val}}} \Big| Q_{\theta_t,h}(s,a) - \widehat{Q}_h(s,a) \Big|. \tag{29}$$

Here, $y_{h,t}$ denotes the constructed regression target, whereas $Q_{\theta_t,h}$ denotes the learned critic prediction.

Figure 5 shows that RLMAS has zero target-construction error by definition, since its regression targets are the exact cached-tree DP values. In contrast, the learned continuation term in equation 27 yields nonzero, training-dependent target error. After 1,000 updates, DQN's mean target RMSE is 0.0664, 0.0750, and 0.0247 on MMLU-Pro, SuperGPQA, and MATH, respectively. The random-path baseline also retains nonzero target error, illustrating that fixed sampled-return targets need not coincide with optimal cached-tree values. Importantly, exact targets do not imply an exact learned critic. Figure 6 shows that RLMAS achieves lower final mean uniform critic error than both baselines. Its final values are 1.201, 1.386, and 1.008, compared with 1.293, 1.652, and 1.031 for DQN, respectively. The improvement is modest on MATH, and RLMAS's validation error does not decrease monotonically on SuperGPQA. Together, these observations support the distinction between eliminating learned-bootstrap target-construction error and improving, rather than eliminating, critic approximation error. All exactness claims are conditional on the fixed sampled trees, not on the unknown expected action values of the underlying stochastic language-model environment.

In contrast, equation 12 replaces the learned bootstrap term with the exact continuation value obtained by backward dynamic programming on the fixed counterfactual branching tree $\widehat{\mathcal{T}}_i(\omega_i)$. Once the tree is fixed, the resulting targets are fixed and, by Theorem 2, equal the optimal action values on that tree.

Thus, under Assumption 1, complete counterfactual branching removes bootstrap approximation error from target construction on the fixed tree, separating target construction from subsequent critic fitting. This conclusion does not imply that the learned critic itself is error-free: regression error from function approximation and discrepancy between the fixed counterfactual trees and the stochastic deployment environment may still remain. Merely collecting counterfactual transitions while retaining learned bootstrap targets would not provide the exact-target guarantee of Theorem 2.

B.3 From Critic Error to the Counterfactual-Tree Oracle Gap

Let $Q_{\theta,t}$ denote the stage-aware critic, implemented by a shared network through $Q_{\theta,t}(s,a) := q_\theta(\phi(s), a)$. The state vector $\mathbf{v} = \phi(s)$ includes decision-stage information, so separate networks for different stages are not required. The corresponding greedy policy is defined over legal actions:

$$\pi_{\theta,t}(s) \in \arg\max_{a\in\mathcal{A}_t(s)} Q_{\theta,t}(s,a), \qquad t = 0, \ldots, H-1. \tag{30}$$

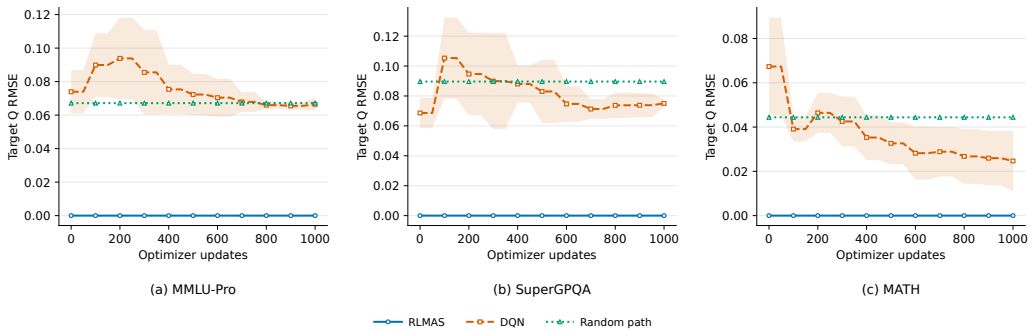

Figure 5: **Target-construction error relative to the cached-tree DP oracle.** Target RMSE, defined in equation 28, is evaluated on all legal validation state–action pairs. Panels correspond to (a) MMLU-Pro, (b) SuperGPQA, and (c) MATH. Lines and shaded bands show the mean and ± 1 sample standard deviation across five training seeds. RLMAS uses exact DP targets; DQN uses learned bootstrap targets; Random path uses fixed sampled-return targets. For a common-pair diagnostic, the random-path target is extended to both root continuation actions using the same fixed sampled second-stage action; training still uses only observed paths. Terminal-transition targets equal immediate rewards and contribute zero target error for every method. RLMAS's zero curve concerns target construction, not learned-critic prediction error.

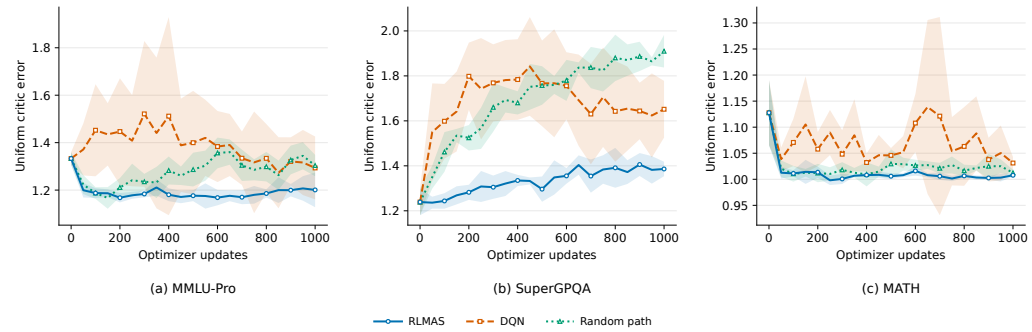

Figure 6: **Uniform critic approximation error on held-out cached trees.** For each training seed and checkpoint, we compute the maximum absolute discrepancy between the learned critic and the exact cached-tree DP values over all legal validation state–action pairs, as defined in equation 29. Lines and shaded bands report the mean and ± 1 sample standard deviation of these per-seed maxima across five seeds. Panels correspond to (a) MMLU-Pro, (b) SuperGPQA, and (c) MATH. Lower values indicate a smaller worst-case approximation error on the evaluated trees. Unlike Figure 5, this figure measures critic predictions rather than regression targets. Panel-specific vertical scales are used.

For a fixed counterfactual tree $\widehat{\mathcal{T}}_i$, let $\widehat{V}_t^{\pi}(s)$ denote the value of policy π and $\widehat{V}_t^{\star}(s)$ the optimal value, suppressing the tree index in both. All values are defined by the cumulative incremental rewards in equation 10, with collaboration costs measured by incremental token usage. Exit (E) has zero incremental reward, and all terminal nodes have zero continuation value.

Theorem 3 (Counterfactual-tree oracle-gap bound)**.** *Under Assumption 1, suppose that the critic satisfies the uniform approximation bound over the fixed collection of counterfactual trees:*

$$\max_{\substack{0\le t<H,\ s\in\widehat{\mathcal{S}}_t \\ a\in\mathcal{A}_t(s)}} \left|Q_{\theta,t}(s,a) - \widehat{Q}_t(s,a)\right| \le \varepsilon, \qquad \varepsilon \ge 0. \tag{31}$$

Then, for every root state $s_{i,0}$ in this collection,

$$0 \le \widehat{V}_0^{\star}(s_{i,0}) - \widehat{V}_0^{\pi_\theta}(s_{i,0}) \le 2H\varepsilon. \tag{32}$$

For the $H = 2$ setting used in RLMAS, the counterfactual-tree oracle gap is therefore at most 4ε.

Proof. By Theorem 2, the dynamic-programming targets $\widehat{Q}_t$ equal the optimal action values $\widehat{Q}_t^\star$ on each fixed counterfactual tree.

Fix a nonterminal state s at decision stage t, and choose

$$a^\star \in \arg\max_{a \in \mathcal{A}_t(s)} \widehat{Q}_t(s,a), \qquad a_\theta = \pi_{\theta,t}(s). \tag{33}$$

The uniform approximation bound in equation 31 and the greedy choice of a_θ imply

$$\widehat{Q}_t(s,a^\star) \le Q_{\theta,t}(s,a^\star) + \varepsilon \tag{34}$$
$$\le Q_{\theta,t}(s,a_\theta) + \varepsilon \tag{35}$$
$$\le \widehat{Q}_t(s,a_\theta) + 2\varepsilon. \tag{36}$$

Since $\widehat{V}_t^\star(s) = \widehat{Q}_t(s,a^\star)$, the one-step greedy gap satisfies

$$\widehat{V}_t^\star(s) - \widehat{Q}_t(s,a_\theta) \le 2\varepsilon. \tag{37}$$

Define the policy value gap at stage t as

$$\Delta_t(s) = \widehat{V}_t^\star(s) - \widehat{V}_t^{\pi_\theta}(s), \tag{38}$$

with $\Delta_t(s) = 0$ at terminal nodes.

If $a_\theta = \mathrm{E}$, the trajectory terminates immediately with zero reward and continuation value. Hence,

$$\Delta_t(s) = \widehat{V}_t^\star(s) - \widehat{Q}_t(s,a_\theta) \le 2\varepsilon. \tag{39}$$

Otherwise, let $s' = F_t(s,a_\theta)$. If s' is terminal, both $\widehat{Q}_t(s,a_\theta)$ and $\widehat{V}_t^{\pi_\theta}(s)$ equal $r_t(s,a_\theta)$, so the same one-step bound applies.

If s' is nonterminal, the Bellman recursions give

$$\Delta_t(s) = \widehat{V}_t^\star(s) - \widehat{Q}_t(s,a_\theta) + \Delta_{t+1}(s') \tag{40}$$
$$\le 2\varepsilon + \Delta_{t+1}(s'). \tag{41}$$

Unrolling this inequality along the trajectory induced by π_θ for at most H decision stages yields

$$\Delta_0(s_{i,0}) \le 2H\varepsilon. \tag{42}$$

The lower bound follows from the optimality of $\widehat{V}_0^\star$ on the fixed counterfactual tree. □

Theorem 3 links critic approximation error to policy suboptimality on the same counterfactual trees. Together, the analysis distinguishes three aspects of counterfactual fitted Q-learning: complete branching yields exact tree-specific targets and the remaining policy gap is controlled by the critic's uniform approximation error. Other sources of training variance may remain. These guarantees concern the constructed counterfactual trees and do not, by themselves, bound policy suboptimality in the underlying stochastic collaboration process.

To examine the empirical consistency of the bound, we evaluate RLMAS on fixed held-out counterfactual trees from MMLU-Pro, SuperGPQA, and MATH. Within each dataset, let $\mathcal{D}_{\mathrm{val}}$ contain all legal tuples (t,s,a) from the same n_{val} validation trees. For each training seed and checkpoint k, we compute

$$\varepsilon_k = \max_{(t,s,a)\in\mathcal{D}_{\mathrm{val}}} \left| Q_{\theta_k,t}(s,a) - \widehat{Q}_t(s,a) \right|, \tag{43}$$

$$\Delta_k^{\max} = \max_{1\le i\le n_{\mathrm{val}}} \left[\widehat{V}_0^\star(s_{i,0}) - \widehat{V}_0^{\pi_{\theta_k}}(s_{i,0}) \right]. \tag{44}$$

Here, k indexes training checkpoints rather than decision stages, and the seed index is omitted. Both quantities are computed separately for each seed on the same validation trees. The critic error is a maximum absolute error rather than RMSE, matching the assumption of Theorem 3.

Figure 7 compares $\Delta_k^{\max}$ with the theoretical bound $4\varepsilon_k$ for $H = 2$. Across five training seeds and 21 checkpoints per seed, all 105 observations in each dataset satisfy $\Delta_k^{\max} \le 4\varepsilon_k$, with no violations among the 315 observations. No checkpoints are excluded based on their errors or policy gaps. The observed gaps lie substantially below the theoretical boundary, indicating that the bound is conservative on these evaluated trees. The results are consistent with the bound, but do not establish tightness of its constant or a monotonic relationship between critic error and policy gap.

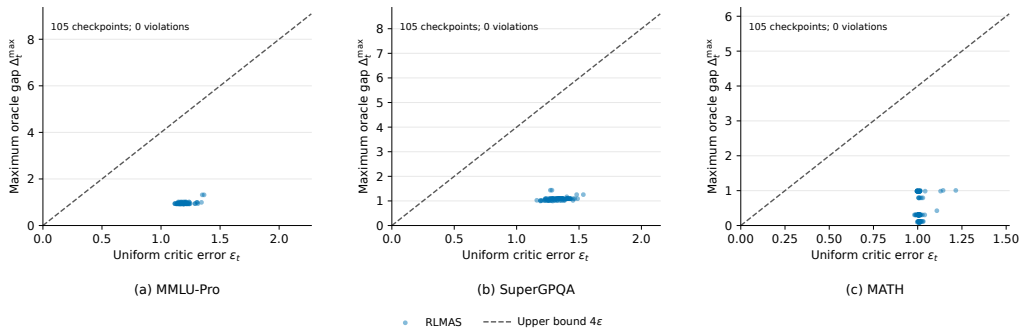

Figure 7: **Empirical consistency of the counterfactual-tree oracle-gap bound for RLMAS.** Each point represents one training checkpoint from one seed, evaluated on the same fixed validation trees within each dataset. The horizontal axis shows the uniform critic error ε_k over all legal state–action pairs; the vertical axis shows the maximum root oracle gap $\Delta_k^{\max}$ across validation questions. Panels correspond to (a) MMLU-Pro, (b) SuperGPQA, and (c) MATH. All 105 observations per dataset are included. The dashed line denotes the theoretical upper bound $4\varepsilon_k$ for $H = 2$, not a regression fit. No observations violate the bound. This diagnostic concerns the fixed counterfactual trees, not the underlying stochastic collaboration process.

C Experimental Details

C.1 Benchmarks and Data Splits.

We evaluate RLMAS on MMLU-Pro, SuperGPQA, and MATH. For each benchmark, we use 500 held-out questions for comparative evaluation and a separate pool of 500 training candidates, partitioned by category into a nominal 400-question training set and a 100-question validation set. Experiments are conducted with three LLMs: Qwen3-4B, DeepSeek-V4-Flash, and GPT-OSS-20B. For all model calls, we set the temperature to 0.7, top-p to 0.95, and the maximum output length to 8196 tokens.

C.2 Controller Architecture and Training Details

We train a unified Q-network with two 128-unit ReLU hidden layers and three action outputs. Full counterfactual trajectories provide targets for the available actions at both decision steps. The per-action reward is the change in answer correctness minus λ times the action's total token usage divided by 1024; initial-answer tokens are treated as a sunk cost for policy optimization. The reported configuration uses $\lambda = 0.01$, with no potential shaping and a unit penalty for changing a correct answer to an incorrect one. We use Adam with learning rate 10^{-3}, weight decay 10^{-4}, batch size 64, Huber loss, and gradient clipping at 5. Training runs for at most 200 epochs with validation-based early stopping after 20 epochs without improvement. Experiments train seeds 1–5; the evaluated checkpoint and seed are fixed by the corresponding run configuration before test evaluation.

AI Use Statement

We used generative AI tools, including OpenAI Codex and ChatGPT, to assist with language editing, research code development and debugging, and literature search. Separately, the LLMs evaluated in this study generated responses and counterfactual collaboration trajectories as part of the experimental pipeline described in Section 3 and Appendix C. The authors reviewed and revised the AI-assisted text, manually reviewed and tested the AI-assisted code, and checked the reported numerical results against the underlying experimental records. The authors take full responsibility for the research methodology, theoretical claims, experimental results, and final content of this work.