

DR. MAS: STABLE REINFORCEMENT LEARNING FOR MULTI-AGENT LLM SYSTEMS

Lang Feng¹ Longtao Zheng¹ Shuo He¹ Fuxiang Zhang¹ Bo An^{1,*}

¹Nanyang Technological University, Singapore

ABSTRACT

Multi-agent LLM systems enable advanced reasoning and tool use via role specialization, yet reliable reinforcement learning (RL) post-training for such systems remains difficult. In this work, we theoretically pinpoint a key reason for training instability when extending group-based RL to multi-agent LLM systems. We show that under GRPO-style optimization, a global normalization baseline may deviate from diverse agents' reward distributions, which ultimately leads to gradient-norm instability. Based on this finding, we propose Dr. MAS, a simple and stable RL training recipe for multi-agent LLM systems. Dr. MAS uses an agent-wise remedy: normalizing advantages per agent using each agent's own reward statistics, which calibrates gradient scales and dramatically stabilizes training, both theoretically and empirically. Beyond the algorithm, Dr. MAS provides an end-to-end RL training framework for multi-agent LLM systems, supporting scalable orchestration, flexible per-agent LLM serving and optimization configs, and shared resource scheduling of LLM actor backends. We evaluate Dr. MAS on multi-agent math reasoning and multi-turn search benchmarks using Qwen2.5 and Qwen3 series models. Dr. MAS achieves clear gains over vanilla GRPO (e.g., +5.6% avg@16 and +4.6% pass@16 on math, and +15.2% avg@16 and +13.1% pass@16 on search) while largely eliminating gradient spikes. Our code is available at <https://github.com/langfengQ/DrMAS>.

1 INTRODUCTION

Large Language Models (LLMs) (Achiam et al., 2023; Team et al., 2023; Liu et al., 2024; Yang et al., 2025) have demonstrated impressive reasoning abilities across diverse domains (e.g., question answering, code generation), but many practical applications involve complex scenarios where multiple agents need to interact and coordinate. For example, tasks like complex information retrieval (Chang et al., 2025; Zhang et al., 2025d), agentic software engineering (Hong et al., 2024; Qian et al., 2024), and open-ended device control (Wang et al., 2024; Tan et al., 2025) involve multiple participants working together over extended horizons. Organizing LLMs into a Multi-Agent System (MAS), where each agent specializes in a subtask or role, has become a trend to handle complex real-world tasks more effectively (Tran et al., 2025; Zhang et al., 2025a).

Reinforcement Learning (RL) (Sutton & Barto, 2018) now plays a foundational role in LLM post-training. Despite its growing importance, the extension of RL training to multi-agent LLM systems remains largely underexplored from both *algorithmic* and *system* perspectives. On the algorithmic side, while group-based RL methods like Group Relative Policy Optimization (GRPO) (Shao et al., 2024) excel in single-agent scenarios, adapting them to multi-agent settings introduces significant challenges due to the frequent *instability* observed across various scenarios (Chen et al., 2025; Zhao et al., 2025; Yuan et al., 2025). Agents are often invoked at different frequencies, leading to heterogeneous data distributions that greatly complicate end-to-end optimization (Hong et al., 2025). On the system side, recent large-scale RL post-training frameworks, like veRL (Sheng et al., 2024) and ROLL (Wang et al., 2025b), provide flexible, high-throughput training pipelines for LLMs, but are largely designed around optimizing a single LLM actor. They generally lack the native support for efficient multi-agent orchestration and multiple LLMs' co-training, restricting the flexible scheduling and resource sharing required for heterogeneous agent configurations.

*Corresponding author: boan@ntu.edu.sg

In this work, we theoretically identify that applying vanilla GRPO to train multi-agent LLM systems introduces systematic gradient variance and destabilizes training. We provide rigorous mathematical and empirical analysis, demonstrating that using a global advantage baseline across agents can inflate the second moment of their gradients, leading to gradient-norm explosion. Building on this analysis, we propose Dr. MAS, a simple and stable RL training recipe for multi-agent LLM systems. Dr. MAS adopts a straightforward yet effective remedy: each agent normalizes rewards using its own mean and variance. Concretely, we group action experience by agent so that each policy’s advantage estimates are normalized with respect to its own data distribution. This calibration balances per-agent gradients, thus resulting in a dramatic reduction in variance for the policy gradient estimator. Beyond the algorithm itself, Dr. MAS also provides an end-to-end RL training framework tailored for multi-agent LLM systems. It supports scalable multi-agent orchestration, flexible agent-model assignment with optional LLM sharing (e.g., co-training 7B and 3B models), per-agent configuration of optimization, and shared resource pooling for efficient scheduling of LLM actor backends. The result is a unified system that maintains well-conditioned gradients and high hardware efficiency while enabling stable co-training across multiple LLM agents.

We evaluate Dr. MAS on role-specialized multi-agent systems for math reasoning and multi-turn search, using Qwen2.5 (Bai et al., 2025) and Qwen3 (Yang et al., 2025) series models, under both LLM-sharing and non-sharing settings. Across tasks and settings, Dr. MAS consistently improves the performance over vanilla GRPO (e.g., +5.6% avg@16 and +4.6% pass@16 on math, and +15.2% avg@16 and +13.1% pass@16 on search). We also observe markedly improved stability, with gradient-norm spikes largely eliminated.

2 RELATED WORK

Reinforcement Learning for LLMs. RL post-training has evolved from alignment-focused RLHF (Ziegler et al., 2019; Ouyang et al., 2022) to Reinforcement Learning from Verifiable Rewards (RLVR) (Zeng et al., 2025), where group-based methods like GRPO (Shao et al., 2024) serve as efficient alternatives to PPO (Schulman et al., 2017). By aggregating multiple rollouts and performing relative comparisons, these techniques avoid explicit value-function learning and have been successfully extended to agentic tasks involving tool use and reasoning (Qian et al., 2025; Wang et al., 2025c; Feng et al., 2025b; Xue et al., 2025b; Dong et al., 2025).

Reinforcement Learning for Multi-Agent LLMs. Research has expanded into learning coordination within multi-agent systems, utilizing techniques such as self-play (Liu et al., 2025a; Yuan et al., 2025) and multi-agent distillation (Li et al., 2025a). Meanwhile, group-relative optimization has been extended to multi-agent settings (Liu et al., 2025b; Chen et al., 2025; Park et al., 2025; Wan et al., 2025; Mo et al., 2025; Hong et al., 2025), but these methods typically rely on heuristics and lack stability guarantees. Dr. MAS distinguishes itself by theoretically identifying *gradient-norm inflation* as the root cause of instability and proposing a simple yet rigorous agent-wise solution.

Reinforcement Learning Infrastructure. As RL post-training scales, infrastructure has shifted toward optimizing the end-to-end *rollout-train loop*. While general-purpose frameworks, like veRL (Sheng et al., 2024), OpenRLHF (Hu et al., 2024), ROLL (Wang et al., 2025b), and AReaL (Fu et al., 2025), improve throughput, and newer tools like MARTI (Zhang et al., 2025c) and PettingLLMs (Zhao et al., 2025) offer multi-agent interfaces, they often lack native support for heterogeneous model assignments or shared resource pooling. Dr. MAS addresses these system-level challenges to enable scalable and hardware-efficient multi-agent post-training.

3 PRELIMINARIES

3.1 MULTI-AGENT LLMs

We consider a cooperative multi-agent LLM system consisting of K distinct LLM agents $\pi_{\theta_1}, \pi_{\theta_2}, \dots, \pi_{\theta_K}$, each parameterized by its own LLM weights θ_k . The agents jointly engage in solving complex tasks sampled from a distribution $x \in p(X)$. Each full interaction process (trajectory) produces a single outcome reward $R \in \mathbb{R}$ to indicate success or failure. During task completion, the agents’ joint interaction unfolds as a trajectory $\tau = \{(s_1, \mathbf{a}_1, k_1), (s_2, \mathbf{a}_2, k_2), \dots, (s_T, \mathbf{a}_T, k_T)\}$, where s_t denotes the conversational or contextual

state (e.g., dialogue history, task prompt, or shared memory) at execution step t , \mathbf{a}_t is the text output produced, and $k_t \in 1, \dots, K$ denotes which LLM agent was active at step t . The active agent can change dynamically across the trajectory, for instance, in a hierarchical multi-agent framework, a high-level planner LLM may decide which sub-agent executes at each step. Hence, the index k_t explicitly denotes the identity of the agent executing at each step. The execution LLM agent k_t generates its output based on its policy $a_t \sim \pi_{\theta_{k_t}}(\cdot | s_t)$. Depending on the system design, the agents may share parameters (i.e., $\theta_1 = \dots = \theta_K$), differing in role-specific prompts, enabling efficient adaptation under a unified LLM, or they may maintain distinct parameters ($\theta_i \neq \theta_j$) to specialize in heterogeneous sub-tasks, allowing diverse reasoning capabilities across agents.

3.2 GROUP RELATIVE POLICY OPTIMIZATION

Group-based RL methods like Group Relative Policy Optimization (GRPO) (Shao et al., 2024) optimize policies by comparing multiple rollouts generated from the same task instruction and normalizing their rewards within each group, thereby avoiding explicit value-function estimation. Formally, given a task instruction x , the multi-agent LLM system samples a set of N trajectories

$$\{\tau^i = (s_1^i, a_1^i, k_1^i, \dots, s_{T_i}^i, a_{T_i}^i, k_{T_i}^i)\}_{i=1}^N, \quad (1)$$

where $k_t^i \in \{1, \dots, K\}$ denotes the active agent at step t of trajectory i . Each trajectory τ^i receives a scalar terminal reward $R^i = R(\tau^i) \in \mathbb{R}$ that measures the overall quality of the generated outcome. The normalized advantage for each trajectory is computed using the group’s mean and standard deviation:

$$A_{\text{global}}^i = \frac{R^i - \mu}{\sigma}, \quad \mu = \frac{1}{N} \sum_{i=1}^N R^i, \quad \sigma^2 = \frac{1}{N} \sum_{i=1}^N (R^i - \mu)^2. \quad (2)$$

This advantage is then propagated to all agent outputs that contributed to the trajectory. Formally, we define the set of outputs of agent k as $\mathcal{Y}_k = \{a_t^i | k_t^i = k\}$, i.e., the collection of all time steps (i, t) across the group at which agent k produces an action. Notably, agents are often invoked at different frequencies, which results in varying sample sizes $|\mathcal{Y}_k|$. The RL objective for agent k is then given by

$$\mathcal{J}_k(\theta_k) = \mathbb{E}_{x \sim p(x)} \left[\frac{1}{|\mathcal{Y}_k|} \sum_{\mathbf{a}_t^i \in \mathcal{Y}_k} \min \left(\rho_{\theta_k}(\mathbf{a}_t^i) A_{\text{global}}^i, \text{clip}(\rho_{\theta_k}(\mathbf{a}_t^i), 1 \pm \epsilon) A_{\text{global}}^i \right) \right], \quad (3)$$

where $\rho_{\theta_k}(\mathbf{a}_t^i) = \frac{\pi_{\theta_k}(\mathbf{a}_t^i | s_t^i)}{\pi_{\theta_k^{\text{old}}}(\mathbf{a}_t^i | s_t^i)}$ is the importance sampling ratio. Here, we omit the KL-divergence regularization for notational brevity.

4 METHODOLOGY

In a multi-agent LLM system, different agents often specialize in distinct functions (e.g., information retrieval vs. answer synthesis, high-level planning vs. low-level execution), and consequently can exhibit substantially different reward distributions. We find that using vanilla GRPO with the global baseline (μ, σ) for all agents can be suboptimal: some agents may consistently operate in reward distributions above the global mean, while others remain below it. This persistent bias in how advantages are normalized can introduce a deterministic shift in the effective advantages seen by each agent, which in turn can inflate gradient-estimator variance and destabilize training.

In this section, we introduce Dr. MAS by (1) theoretically formalizing the instability and analyzing the second moment of the per-agent gradient under GRPO optimization (Section 4.1); (2) proposing an agent-wise remedy that calibrates each agent’s advantage using its own reward statistics, thereby improve the training stability (Section 4.2); and (3) describing a system framework that implements efficient end-to-end RL training recipe for multi-agent LLM systems (Section 4.3).

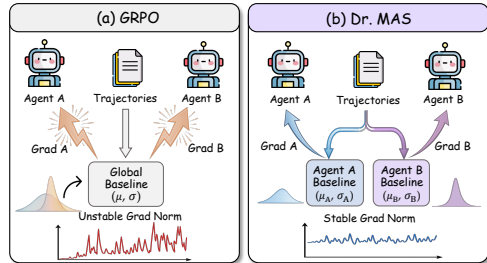


Figure 1: Algorithm comparison. (a) GRPO with global baseline (μ, σ) can cause unstable gradient norm. (b) Dr. MAS with per-agent normalization (μ_k, σ_k) stabilizes the training of MAS.

4.1 RISK OF GRADIENT NORM EXPLOSION

To focus on how advantage normalization causes the instability in MAS training, we perform a theoretical analysis of the gradient norm. We base our analysis on the unclipped GRPO gradient (clipping and other regularization only further bound the update and are orthogonal to the gradient issue we study). For each agent k , and for each step (i, t) such that $k_t^i = k$, we define the *score function* as $z_{i,t}^{(k)} \triangleq \nabla_{\theta_k} \rho_{\theta_k}(\mathbf{a}_t^i)$ and corresponding (unclipped) GRPO gradient contribution as

$$\tilde{g}_k^{\text{global}} \triangleq \frac{R^i - \mu}{\sigma} z_{i,t}^{(k)}. \quad (4)$$

Here, (μ, σ) are the mean and standard deviation used by vanilla GRPO. We assume that each agent's score function has a bounded second moment:

Assumption 4.1. *For each agent k , there exists a constant $C_k < \infty$ such that $\mathbb{E}[\|z_{i,t}^{(k)}\|^2] \leq C_k$.*

Then we can express the second moment of the per-agent gradient as follows.

Lemma 4.2. *Under Assumptions 4.1, for any agent k ,*

$$\mathbb{E}_{\mathbf{a}_t^i \sim \mathcal{Y}_k} [\|\tilde{g}_k^{\text{global}}\|^2] = \mathbb{E}_{\mathbf{a}_t^i \sim \mathcal{Y}_k} [\|z_{i,t}^{(k)}\|^2] \frac{\sigma_k^2 + (\mu_k - \mu)^2}{\sigma^2} + \Delta_k,$$

where $\mu_k \triangleq \frac{1}{|\mathcal{Y}_k|} \sum_{\mathbf{a}_t^i \in \mathcal{Y}_k} R^i$, $\sigma_k^2 \triangleq \frac{1}{|\mathcal{Y}_k|} \sum_{\mathbf{a}_t^i \in \mathcal{Y}_k} (R^i - \mu_k)^2$ are the mean and variance when sampling time steps uniformly from \mathcal{Y}_k (i.e., when agent k is active). Δ_k is a score-reward covariance correction term.

See Appendix B.1 for the proof. Lemma 4.2 separates the per-agent gradient norm into a dominant scaling factor and a residual covariance correction. The multiplicative factor $(\sigma_k^2 + (\mu_k - \mu)^2)/\sigma^2$ grows when agent k operates in a reward distribution whose mean is far from the global mean or agent k 's conditional reward variance is much larger than the global variance. The term Δ_k captures the residual score-reward correlation. In large-scale LLM training, rewards are typically low-dimensional signals of final task quality (e.g., pass/fail for reasoning, correctness for coding), while $z_{i,t}^{(k)}$ depends mainly on the local token-level stochasticity of the policy. Empirically, their covariance is often much smaller than the dominant scaling factor $\mathbb{E}_k[\|z_{i,t}^{(k)}\|^2](\sigma_k^2 + (\mu_k - \mu)^2)/\sigma^2$. This decomposition reveals the intrinsic instability of global normalization of GRPO in heterogeneous multi-agent training: a large deviation in the dominant scaling factor can inflate the gradient and lead to unstable updates. We formalize this phenomenon below.

Proposition 4.3 (Gradient-Norm Inflation). *As either the normalized mean deviation $|\mu_k - \mu|/\sigma$ or the normalized variance ratio σ_k^2/σ^2 becomes large, the second moment of $\tilde{g}_k^{\text{global}}$ grows at least linearly. Consequently, along any training process for which there exists a sequence of iterations indexed by m such that*

$$\frac{\sigma_{k,m}^2 + (\mu_{k,m} - \mu_m)^2}{\sigma_m^2} \rightarrow \infty, \quad \mathbb{E}[\|\tilde{g}_m^{\text{global}}\|^2] \rightarrow \infty,$$

where $\tilde{g}_m^{\text{global}} = (\tilde{g}_{1,m}^{\text{global}}, \dots, \tilde{g}_{K,m}^{\text{global}})$ stacking all LLM agents' gradients.

The proof is provided in Appendix B.2. Proposition 4.3 demonstrates that gradient-norm inflation can be triggered by *any* agent whose reward distribution is poorly aligned with the global baseline. In practice, the gradient norms in such cases typically do not reach mathematical infinity. However, they often grow large enough and trigger severe gradient spikes, thus destabilizing the training process of the entire multi-agent LLM system.

4.2 AGENT-WISE REMEDY

Fortunately, Proposition 4.3 suggests a straightforward and effective remedy: calibrating each agent's advantages using reward statistics computed exclusively on the steps where that agent is active. Specifically, we replace the global baseline (μ, σ) with (μ_k, σ_k) , which ensures that

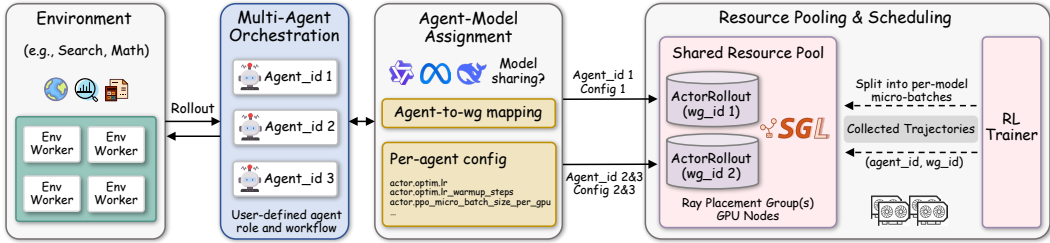


Figure 2: Overview of multi-agent LLM RL framework. A multi-agent orchestrator manages distributed rollouts, agents are mapped to LLM worker groups with optional LLM sharing, and a shared resource pool schedules actor backends for efficient inference and per-model optimization.

$(\sigma_k^2 + (\mu_k - \mu)^2)/\sigma^2 = 1$. In practice, this corresponds to normalizing each agent’s reward using its own empirical mean and variance:

$$A_{\text{agent}}^{i,k} = \frac{R^i - \mu_k}{\sigma_k}, \tag{5}$$

where $\mu_k \triangleq \frac{1}{|\mathcal{Y}_k|} \sum_{\mathbf{a}_t^i \in \mathcal{Y}_k} R^i$ and $\sigma_k^2 \triangleq \frac{1}{|\mathcal{Y}_k|} \sum_{\mathbf{a}_t^i \in \mathcal{Y}_k} (R^i - \mu_k)^2$. Therefore, an analysis analogous to Lemma 4.2 yields

$$\mathbb{E}_{\mathbf{a}_t^i \sim \mathcal{Y}_k} [\|\tilde{g}_k^{\text{agent}}\|^2] = \mathbb{E}_{\mathbf{a}_t^i \sim \mathcal{Y}_k} [\|z_{i,t}^{(k)}\|^2] + \Delta_k, \tag{6}$$

where $\tilde{g}_k^{\text{agent}} = \frac{R^i - \mu_k}{\sigma_k} z_{i,t}^{(k)}$. Thus, under agent-wise normalization, the second moment of each agent’s gradient is bounded purely by its own score statistics. Crucially, this effect is inherently *multi-agent*: as the number of specialized agents increases and their roles become more heterogeneous, a single global baseline is increasingly likely to be badly aligned with some agents, leading to gradient norm explosion. As shown in Figure 1, a simple agent-wise remedy, by adapting to each agent’s own statistics, achieves keeping all gradients in a comparable, well-conditioned range, while still enabling cooperative optimization of the overall multi-agent LLM system.

4.3 FRAMEWORK FOR MULTI-AGENT LLM RL

We next present a unified system framework that realizes end-to-end RL post-training for multi-agent LLMs. As illustrated in Figure 2, the system is designed to ensure well-conditioned gradient updates across agents, while maintaining scalable orchestration, flexible agent-model assignment, per-agent optimization configs, and efficient hardware utilization for multi-agent rollouts.

Multi-Agent Orchestration. Our system is coordinated by a multi-agent trajectory collector, which manages the distributed interaction between the multi-agent LLM system and the environment. It delegates the rollout to a user-defined multi-agent orchestra, which governs the agent roles and execution flow. The orchestra dynamically selects and invokes agent policies based on the current state or prior agent outputs, enabling flexible and conditional control over multi-agent decision-making.

Agent-Model Assignment. A core assignment logic maps logical agents $(1, \dots, K)$ to physical LLM worker groups (`wg_id`). In non-shared settings, each agent k is assigned a distinct worker group (e.g., 7B and 3B models). Conversely, in shared settings, all agents configured with the same model are mapped to a single, shared worker group, allowing joint training and inference while reusing model weights.

Per-Agent Configuration. Dr. MAS supports agent-specific training hyperparameters for granular control. This allows configurations like `actor.optim.lr` to be specified on a per-agent basis. Our system injects the k -th hyperparameter set into the configuration for agent k , which is then attached to its corresponding LLM work group. A runtime check ensures that all agents sharing the same worker group utilize identical configurations.

Shared Resource Pooling and Scheduling. This component decouples logical agent-model assignments from physical resource placement. A resource pool manager provisions hardware resources (e.g., GPUs) into named pools. All LLM actor backends (one for each `wg_id`) are mapped

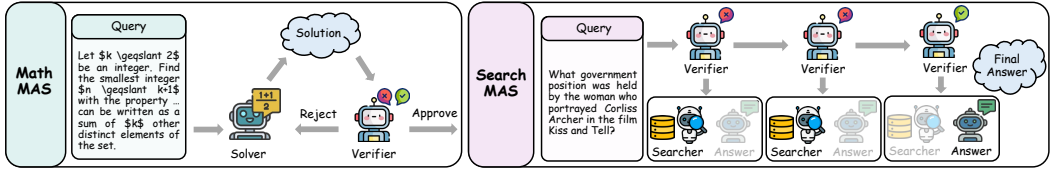


Figure 3: Illustration of the orchestrations. **Left:** Math orchestration uses a two-agent loop, where a solver proposes candidate solutions and a verifier evaluates and either approves or requests refinement. **Right:** Multi-turn search orchestration uses a hierarchical three-agent pipeline, where a top-level verifier selectively invokes either a search agent to retrieve external information or an answer agent to produce the final result.

Table 1: Math results on Qwen3-4B/8B. We report the avg@16 and pass@16 of single-agent training with GRPO and multi-agent training under LLM sharing/non-sharing, using vanilla GRPO and Dr. MAS. Subscripts for Dr. MAS denote Δ over the vanilla GRPO under the same multi-agent setting.

Benchmark	Single-Agent GRPO		Multi-Agent & LLM Sharing				Multi-Agent & LLM Non-Sharing			
	avg@16	pass@16	avg@16	pass@16	avg@16	pass@16	avg@16	pass@16	avg@16	pass@16
<i>Qwen3-4B</i>										
AIME'24	38.8	63.3	39.3	64.0	39.3 _{0.0}	63.3 _{0.7}	42.7	73.3	46.9 _{1.2}	80.0 _{16.7}
AIME'25	33.1	56.7	31.4	53.3	38.1 _{6.7}	63.3 _{10.0}	35.6	63.3	38.1 _{2.5}	66.7 _{3.4}
AMC'23	83.5	95.0	85.6	95.0	87.3 _{1.7}	95.0 _{0.0}	83.5	95.0	89.5 _{6.0}	97.5 _{2.5}
MATH500	89.0	94.2	89.5	96.2	90.5 _{1.0}	96.0 _{0.2}	89.6	95.0	92.4 _{2.8}	97.0 _{2.0}
Minerva	37.9	49.6	37.5	50.0	40.9 _{3.4}	53.3 _{3.3}	37.5	50.7	39.0 _{1.5}	51.5 _{0.8}
Olympiad	53.3	66.5	57.6	65.6	58.2 _{0.6}	68.6 _{3.0}	56.3	68.9	60.9 _{1.6}	73.6 _{1.7}
Average	55.9	70.9	56.8	70.7	59.0 _{2.2}	73.2 _{2.6}	57.5	74.4	61.1 _{3.6}	77.7 _{3.3}
<i>Qwen3-8B</i>										
AIME'24	36.0	67.3	42.7	66.7	54.8 _{12.1}	80.0 _{13.3}	42.9	70.0	44.6 _{1.7}	73.3 _{3.3}
AIME'25	32.7	50.0	31.4	53.3	39.4 _{8.0}	70.0 _{16.7}	31.8	53.3	41.5 _{9.7}	56.7 _{3.4}
AMC'23	87.0	95.0	87.3	95.0	88.9 _{1.6}	97.5 _{2.5}	86.1	95.0	87.5 _{1.4}	95.0 _{0.0}
MATH500	89.9	94.8	89.6	96.2	91.3 _{1.7}	96.0 _{0.2}	90.5	96.6	90.7 _{0.2}	96.2 _{0.4}
Minerva	36.0	46.7	37.5	50.0	39.9 _{2.4}	49.6 _{0.4}	39.2	50.7	40.9 _{1.7}	54.0 _{3.3}
Olympiad	57.9	67.5	58.2	71.4	59.3 _{1.1}	72.4 _{1.0}	58.2	67.6	59.0 _{0.8}	70.2 _{2.6}
Average	56.6	70.2	57.8	72.1	62.3 _{4.5}	77.6 _{5.5}	58.1	72.2	60.7 _{2.6}	74.2 _{2.0}

to the ActorRollout role. To support high-throughput and low-latency decoding in multi-agent rollouts, these actor backends use `sglang` (Zheng et al., 2024) as the inference engine. This allows them to be co-provisioned within the same shared resource pool using Ray placement groups, enabling scalable scheduling of multiple concurrent LLMs. Agent calls are routed by an `agent_to_wg_mapping` (`agent_id` \rightarrow `wg_id`). This mapping dynamically dispatches the agent’s generation request to the correct backend worker group (`actor_rollout_wg[wg_id]`). In the optimization phase, the trainer partitions the aggregated batch \mathcal{B} into per-model micro-batches \mathcal{B}_{wg} according to their worker group ID. Policy updates are then performed for each worker group, ensuring that gradients from an agent’s trajectories only update its designated LLM backend.

5 EXPERIMENT

In this section, we evaluate Dr. MAS on two multi-agent orchestrations: a two-agent loop pipeline for math reasoning and a three-agent hierarchical pipeline for multi-turn search (Figure 3), under both LLM sharing and non-sharing settings. Specifically, we aim to demonstrate: (1) the consistent performance gains of Dr. MAS over vanilla GRPO; (2) more stable training dynamics and smoother gradient norms of Dr. MAS; (3) the individual contribution of each normalization component via a detailed ablation study (Appendix F.1); and (4) the practical efficiency and compatibility of Dr. MAS when applied to heterogeneous agent-model assignments (Appendix F.2).

5.1 MATH

Math Orchestration. We first evaluate Dr. MAS on challenging mathematical reasoning tasks using a two-agent architecture (a *solver agent* and a *verifier agent*), as shown in Figure 3. In each

Table 2: Multi-turn search QA results on Qwen2.5-3B/7B. We report the avg@16 and pass@16 of single-agent training with GRPO and multi-agent training under LLM sharing/non-sharing, using vanilla GRPO and Dr. MAS. Subscripts for Dr. MAS denote Δ over the vanilla GRPO under the same multi-agent setting.

Benchmark	Single-Agent GRPO		Multi-Agent & LLM Sharing GRPO			Multi-Agent & LLM Non-Sharing GRPO				
	avg@16	pass@16	avg@16	pass@16	Dr. MAS avg@16	Dr. MAS pass@16	avg@16	pass@16	Dr. MAS avg@16	Dr. MAS pass@16
<i>Qwen2.5-3B</i>										
NQ	40.6	54.7	41.0	59.0	43.8 _{±2.8}	58.5 _{±0.5}	43.8	54.5	44.6 _{±0.8}	58.1 _{±3.6}
TriviaQA	58.1	68.8	57.9	68.4	61.7 _{±3.8}	70.1 _{±1.7}	60.6	70.8	61.1 _{±0.5}	71.7 _{±0.9}
PopQA	44.2	49.6	43.2	58.0	45.0 _{±1.8}	57.6 _{±0.4}	45.6	54.5	46.5 _{±0.9}	57.4 _{±2.9}
HotpotQA	31.8	40.9	32.5	48.0	33.3 _{±0.8}	51.2 _{±3.2}	32.5	45.2	35.3 _{±2.8}	51.1 _{±5.9}
2Wiki	29.9	43.7	33.7	64.0	34.1 _{±0.4}	64.0 _{±0.0}	29.2	48.9	34.9 _{±5.7}	60.2 _{±11.3}
MuSiQue	7.9	14.6	9.1	26.5	10.2 _{±1.1}	25.8 _{±0.7}	8.6	19.2	10.4 _{±1.8}	26.1 _{±6.9}
Bamboogle	15.3	27.2	26.4	46.4	28.6 _{±2.2}	49.6 _{±3.2}	21.0	33.6	25.4 _{±4.4}	46.4 _{±12.8}
Average	32.5	42.8	34.8	52.9	36.7 _{±1.8}	53.8 _{±0.9}	34.5	46.7	36.9 _{±2.4}	53.0 _{±6.3}
<i>Qwen2.5-7B</i>										
NQ	46.4	57.6	45.2	60.0	47.4 _{±2.2}	60.7 _{±0.7}	27.1	39.0	47.7 _{±20.6}	59.5 _{±20.5}
TriviaQA	63.1	72.4	63.9	70.9	63.1 _{±0.8}	71.2 _{±0.3}	53.1	64.4	63.4 _{±10.3}	72.7 _{±8.3}
PopQA	47.2	53.6	43.9	55.0	45.9 _{±2.0}	57.3 _{±2.3}	20.7	27.9	46.7 _{±26.0}	57.8 _{±29.9}
HotpotQA	43.0	55.1	40.3	55.0	42.5 _{±2.2}	56.0 _{±1.0}	24.4	36.2	44.0 _{±19.6}	57.5 _{±21.3}
2Wiki	40.6	61.6	41.6	67.8	42.0 _{±0.4}	67.1 _{±0.7}	30.3	51.2	45.4 _{±15.1}	68.1 _{±16.9}
MuSiQue	17.8	34.6	15.2	31.7	16.7 _{±1.5}	32.1 _{±0.4}	8.3	18.1	19.4 _{±11.1}	34.9 _{±6.8}
Bamboogle	36.7	54.4	40.1	58.4	40.1 _{±0.0}	59.2 _{±0.8}	31.9	46.4	39.8 _{±7.9}	57.6 _{±11.2}
Average	42.1	55.6	41.5	57.0	42.5 _{±1.1}	57.7 _{±0.7}	28.0	40.5	43.8 _{±15.8}	58.3 _{±17.8}

episode, the solver agent proposes candidate solutions, while the verifier agent inspects the solver’s reasoning and decides whether the current solution should be accepted or revised. If the verifier deems the solution unsatisfactory, the system triggers another round of solver refinement. Otherwise, the interaction terminates and the final answer is emitted. We use Qwen3-4B/8B (Yang et al., 2025) as the LLM policy for each agent and evaluate both the shared-LLM and non-shared settings.

Setup. For training, we adopt the training corpus from DAPO-Math (Yu et al., 2025), which consists of diverse math problems paired with verifiable solutions and reward signals. The rollout group size is set to 8. For evaluation, we report the avg@16 and pass@16 results on a suite of competitive benchmarks: AIME’24, AIME’25, AMC’23, MATH500 (Hendrycks et al., 2021), Minerva, and OlympiadBench (He et al., 2024). All other experimental details are available in Appendix C.1.

Results. As shown in Table 1, Dr. MAS improves over vanilla GRPO under both LLM sharing and LLM non-sharing. While applying GRPO directly to the multi-agent setting can reach decent average scores, the gains are not always consistent across benchmarks, and some hard splits may not improve. This suggests that a single global normalization of GRPO can make multi-agent training less reliable. In contrast, Dr. MAS uses per-agent normalization to keep each agent’s update on a similar scale, which leads to more consistent improvements across datasets and settings (an overall increase of 5.6% in avg@16 and 4.6% in pass@16). For Qwen3-4B, Dr. MAS improves the performance under the sharing setting from 56.8/70.7 to 59.0/73.2 and improves the non-sharing setting from 57.5/74.4 to 61.1/77.7. The pronounced gain in the non-shared setting suggests that when agents possess independent parameters, their behavioral distributions diverge more significantly, making Dr. MAS’s agent-specific calibration even more critical. Similarly, for Qwen3-8B, we observe strong gains in both configurations. The most significant improvements occur on the challenging AIME benchmarks (e.g., 42.7/66.7 \rightarrow 54.8/80.0 on AIME’24), demonstrating that high-variance gradients from GRPO can easily disrupt the learning of fragile, long-horizon reasoning chains. Dr. MAS guarantees stable convergence, allowing agents to robustly learn the precise deductions.

5.2 MULTI-TURN SEARCH

Search Orchestration. We then evaluate Dr. MAS on the multi-turn search tool-calling task. To this end, we design a hierarchical workflow comprising three agents: a *verifier agent*, a *search agent*, and an *answer agent*, as shown in Figure 3. At the top level, the verifier agent determines whether the information currently available is sufficient to answer the query. If not, it delegates downward to the search agent, which is responsible for retrieving additional external evidence. Once the verifier agent judges that the information is adequate, it invokes the answer agent, which synthesizes all

retrieved evidence into a final answer. We use Qwen2.5-3B/7B (Bai et al., 2025) as the LLM policy for each agent and evaluate both the shared-LLM and non-shared settings.

Setup. Our experimental setup follows Search-R1 (Jin et al., 2025). We employ E5 (Wang et al., 2022) as the retriever. The rollout group size is set to 5 and the max turn is set to 4. For evaluation, we consider both single-hop QA benchmarks (NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2022)) and multi-hop QA benchmarks (HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), Bamboogle (Press et al., 2022)) and report the avg@16 and pass@16 results. For training, we use the mixture of NQ and HotpotQA. All other experimental details are available in Appendix C.2.

Results. As shown in Table 2, the instability shows up more clearly in multi-turn search, as errors can snowball across tool calls and across agents. In this setting, vanilla GRPO is especially risky when LLMs are not shared, since each agent can drift and the same global scaling may no longer match their learning dynamics. A clear example of this failure occurs with Qwen2.5-7B (non-sharing), where vanilla GRPO learns to avoid calling search agents entirely due to high gradient norms, leading to a severe performance drop (28.0/40.5). In contrast, Dr. MAS effectively mitigates this risk and yields consistent improvements, with an overall increase of 15.2% in avg@16 and 13.1% in pass@16. Notably, Dr. MAS restores the performance of Qwen2.5-7B to 43.8/58.3, a result that not only far exceeds the vanilla baseline but also surpasses both the single-agent baseline and the shared-LLM setting. This trend, consistent with our observations in math tasks, highlights that LLM non-sharing can be hurt badly without proper stabilization, and reducing training noise at the agent level becomes crucial.

5.3 GRADIENT-NORM INSTABILITY

Next, we investigate the gradient-norm instability by tracking the training accuracy and per-agent gradient norms during RL post-training for the search orchestration under LLM non-sharing (Qwen2.5-3B) (see Appendix F.3 for math results). As shown in Figure 4, vanilla GRPO induces frequent, high-magnitude gradient norm spikes. The *search* agent has the largest spikes (reaching very high values early and again around the middle of training), the *answer* agent also exhibits large spikes at the beginning, and the *verifier* shows noticeable peaks as well. These spikes mean that some steps produce unusually large updates, which makes training noisy and harder to control.

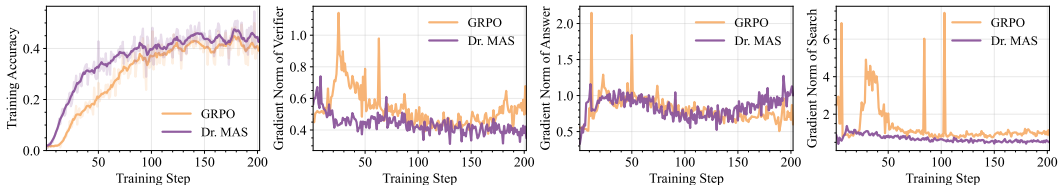


Figure 4: Comparison of training accuracy and gradient norm between GRPO and Dr. MAS.

Dr. MAS mitigates this failure via the agent-wise remedy that normalizes advantages per agent, keeping per-agent update scales better calibrated. As illustrated in Figure 4, Dr. MAS keeps the gradient norms of all three agents much smoother and at a lower level throughout training, therefore achieving notable performance gains in Tables 1 and 2.

6 CONCLUSIONS AND LIMITATIONS

In this work, we studied RL post-training for multi-agent LLM systems and theoretically identify that extending GRPO with a global baseline to multi-agent systems causes gradient instability due to divergent reward statistics. We propose Dr. MAS, which normalizes advantages for each agent using its own reward mean and variance, and we also built an end-to-end training framework that supports multi-agent orchestration, optional LLM sharing and non-sharing, per-agent optimization settings, and efficient resource pooling. Experiments on math and search tasks demonstrate that Dr. MAS mitigates gradient spikes and consistently outperforms GRPO. Limitations include unresolved credit assignment challenges and the need for further evaluation on massive-scale agent systems, leaving advanced resource scheduling for future work.

REFERENCES

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.
- Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. DigiRL: Training in-the-wild device-control agents with autonomous reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 12461–12495, 2024.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibong Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, et al. MAIN-RAG: Multi-agent filtering retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2607–2622, 2025.
- Guanzhong Chen, Shaoxiong Yang, Chao Li, Wei Liu, Jian Luan, and Zenglin Xu. Heterogeneous group-based reinforcement learning for LLM-based multi-agent systems. *arXiv preprint arXiv:2506.02718*, 2025.
- Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.
- Lang Feng, Weihao Tan, Zhiyi Lyu, Longtao Zheng, Haiyang Xu, Ming Yan, Fei Huang, and Bo An. Towards efficient online tuning of VLM agents via counterfactual soft reinforcement learning. In *International Conference on Machine Learning*, volume 267, pp. 16884–16903, 2025a.
- Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, 2025b.
- Lang Feng, Fuchao Yang, Feng Chen, Xin Cheng, Haiyang Xu, Zhenglin Wan, Ming Yan, and Bo An. AgentOCR: Reimagining agent history via optical self-compression. *arXiv preprint arXiv:2601.04786*, 2026.
- Wei Fu, Jiakuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.
- 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. *arXiv preprint arXiv:2103.03874*, 2021.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*, 2020.

- Haoyang Hong, Jiajun Yin, Yuan Wang, Jingnan Liu, Zhe Chen, Ailing Yu, Ji Li, Zhiling Ye, Hansong Xiao, Yefei Chen, et al. Multi-agent deep research: Training multi-agent systems with M-GRPO. *arXiv preprint arXiv:2511.13288*, 2025.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024.
- Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. OpenRLHF: An easy-to-use, scalable and high-performance RLHF framework. *arXiv preprint arXiv:2405.11143*, 2024.
- Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, et al. VerlTool: Towards holistic agentic reinforcement learning with tool use. *arXiv preprint arXiv:2509.01055*, 2025.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-RL: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.
- Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! In *ICLR 2019 Workshop*, 2019.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.
- Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. ComputerRL: Scaling end-to-end online reinforcement learning for computer use agents. *arXiv preprint arXiv:2508.14040*, 2025.
- Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, et al. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic RL. *arXiv preprint arXiv:2508.13167*, 2025a.
- Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. *arXiv preprint arXiv:2510.05592*, 2025b.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, et al. SPIRAL: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *arXiv preprint arXiv:2506.24119*, 2025a.
- Shuo Liu, Tianle Chen, Zeyu Liang, Xueguang Lyu, and Christopher Amato. LLM collaboration with multi-agent reinforcement learning. *arXiv preprint arXiv:2508.04652*, 2025b.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025c.
- Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K Qiu, and Yuqing Yang. Agent lightning: Train any AI agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025.

- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 2022.
- Zhanfeng Mo, Xingxuan Li, Yuntao Chen, and Lidong Bing. Multi-agent tool-integrated policy optimization. *arXiv preprint arXiv:2510.04678*, 2025.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.
- Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. MAPoRL: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30215–30248, 2025.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.
- Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15174–15186, 2024.
- Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiushi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2024.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021, 2020.
- Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.
- Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, et al. Cradle: Empowering foundation agents towards general computer control. In *International Conference on Machine Learning*, 2025.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O’Sullivan, and Hoang D Nguyen. Multi-agent collaboration mechanisms: A survey of LLMs. *arXiv preprint arXiv:2501.06322*, 2025.

- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.
- Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, et al. ReMA: Learning to meta-think for LLMs with multi-agent reinforcement learning. *arXiv preprint arXiv:2503.09501*, 2025.
- Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. SPA-RL: Reinforcing LLM agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*, 2025a.
- Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-Agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. In *Advances in Neural Information Processing Systems*, volume 37, pp. 2686–2710, 2024.
- Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.
- Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, et al. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*, 2025b.
- Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, et al. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025c.
- Xiangyuan Xue, Yifan Zhou, Guibin Zhang, Zaibin Zhang, Yijiang Li, Chen Zhang, Zhenfei Yin, Philip Torr, Wanli Ouyang, and Lei Bai. CoMAS: Co-evolving multi-agent systems via interaction rewards. *arXiv preprint arXiv:2510.08529*, 2025a.
- Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. SimpleTIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025b.
- 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.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- Huining Yuan, Zelai Xu, Zheyue Tan, Xiangmin Yi, Mo Guang, Kaiwen Long, Haojia Hui, Boxun Li, Xinlei Chen, Bo Zhao, et al. MARSHAL: Incentivizing multi-agent reasoning via self-play with strategic llms. *arXiv preprint arXiv:2510.15414*, 2025.
- Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. SimpleRL-Zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.
- Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for LLMs: A survey. *arXiv preprint arXiv:2509.02547*, 2025a.
- Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025b.

- Kaiyan Zhang, Runze Liu, Xuekai Zhu, Kai Tian, Sihang Zeng, Guoli Jia, Yuchen Fan, Xingtai Lv, Yuxin Zuo, Che Jiang, Ziyang Liu, Jianyu Wang, Yuru Wang, Ruotong Zhao, Ermo Hua, Yibo Wang, Shijie Wang, Junqi Gao, Xinwei Long, Youbang Sun, Zhiyuan Ma, Ganqu Cui, Lei Bai, Ning Ding, Biqing Qi, and Bowen Zhou. MARTI: A framework for multi-agent LLM systems reinforced training and inference, 2025c. URL <https://github.com/TsinghuaC3I/MARTI>.
- Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: Orchestrating hierarchical multi-agent intelligence with the tool-environment-agent (tea) protocol. *arXiv preprint arXiv:2506.12508*, 2025d.
- Yujie Zhao, Lanxiang Hu, Yang Wang, Minmin Hou, Hao Zhang, Ke Ding, and Jishen Zhao. Stronger together: On-policy reinforcement learning for collaborative LLMs. *arXiv preprint arXiv:2510.11062*, 2025.
- Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.
- Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583, 2024.
- Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn rl. In *International Conference on Machine Learning*, pp. 62178–62209. PMLR, 2024.
- Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An LLM post-training framework for RL scaling. <https://github.com/THUDM/slime>, 2025.
- Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A EXTENDED RELATED WORK

Reinforcement Learning for LLMs. Beyond early alignment-focused approaches such as RLHF (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Rafailov et al., 2024), recent work studies Reinforcement Learning from Verifiable Rewards (RLVR), which leverages automatically checkable signals (e.g., correctness in math or code) to improve LLM capabilities (Zeng et al., 2025). Within this setting, group-based RL has emerged as a strong alternative to classical actor-critic algorithms like PPO (Schulman et al., 2017). Techniques such as GRPO (Shao et al., 2024), RLOO (Kool et al., 2019; Ahmadian et al., 2024), Dr. GRPO (Liu et al., 2025c), DAPO (Yu et al., 2025), and GSPO (Zheng et al., 2025) aggregate multiple rollouts for the same query and perform relative comparisons within the group, thereby avoiding explicit value-function learning. RLVR has also been extended to agentic, multi-turn settings where LLMs act as automatic agents (Zhou et al., 2024; Bai et al., 2024; Feng et al., 2025a; Wang et al., 2025a; Zhang et al., 2025b; Feng et al., 2026). GRPO-style training has been widely applied to tool use (Qian et al., 2025; Xue et al., 2025b), OS control (Lai et al., 2025), and gaming (Wang et al., 2025c). Further variants refine group construction or objectives, such as GiGPO (Feng et al., 2025b) and ARPO (Dong et al., 2025).

Reinforcement Learning for Multi-Agent LLMs. Recent RL post-training has expanded from single-agent scenarios to *learning coordination* in role-specialized, multi-turn multi-agent systems. Self-play training (e.g., SPIRAL (Liu et al., 2025a), and MARSHAL (Yuan et al., 2025) use multi-turn dynamics to generate curricula and rewards with minimal manual labeling. However, these approaches are often confined to dyadic (two-agent) self-play scenarios. To ease deployment, Chain-of-Agents (Li et al., 2025a) distills multi-agent trajectories into a single agentic policy. Meanwhile, group-relative optimization has been extended to multi-agent settings (Liu et al., 2025b; Chen et al., 2025; Li et al., 2025b; Park et al., 2025; Wan et al., 2025; Xue et al., 2025a; Mo et al., 2025; Hong et al., 2025), but these methods typically rely on heuristics and lack stability guarantees. Dr. MAS distinguishes itself by theoretically identifying *gradient-norm inflation* as the root cause of instability and proposing a simple yet rigorous agent-wise solution.

Reinforcement Learning Infrastructure. As RL post-training scales, infrastructure has shifted toward optimizing the end-to-end *rollout-train loop*, where throughput, scheduling, and variable-length generation dominate system efficiency. General-purpose stacks such as veRL (Sheng et al., 2024), OpenRLHF (Hu et al., 2024), ROLL (Wang et al., 2025b), slime (Zhu et al., 2025), and AReal (Fu et al., 2025) increasingly provide modular pipeline abstractions and distributed execution to improve utilization under heavy sampling. As agentic use cases grow, frameworks increasingly emphasize multi-turn rollout and tool integration (e.g., verl-agent (Feng et al., 2025b), Verl-Tool (Jiang et al., 2025), Agent-Lightning (Luo et al., 2025)), with MARTI (Zhang et al., 2025c) and PettingLLMs (Zhao et al., 2025) offering a practical multi-agent training interface. However, they either provide limited support for heterogeneous model assignments or lack a shared resource pool for efficient scheduling. Our Dr. MAS addresses both to improve scalability and utilization in MAS post-training.

B PROOFS

B.1 PROOF OF LEMMA 4.2

Lemma 4.2. *Under Assumptions 4.1, for any agent k ,*

$$\mathbb{E}_{\mathbf{a}_t^i \sim \mathcal{Y}_k} [\|\tilde{g}_k^{\text{global}}\|^2] = \mathbb{E}_{\mathbf{a}_t^i \sim \mathcal{Y}_k} [\|z_{i,t}^{(k)}\|^2] \frac{\sigma_k^2 + (\mu_k - \mu)^2}{\sigma^2} + \Delta_k,$$

where $\mu_k \triangleq \frac{1}{|\mathcal{Y}_k|} \sum_{\mathbf{a}_t^i \in \mathcal{Y}_k} R^i$, $\sigma_k^2 \triangleq \frac{1}{|\mathcal{Y}_k|} \sum_{\mathbf{a}_t^i \in \mathcal{Y}_k} (R^i - \mu_k)^2$ are the mean and variance when sampling time steps uniformly from \mathcal{Y}_k (i.e., when agent k is active). Δ_k is a score-reward covariance correction term.

Proof. By definition,

$$\tilde{g}_k^{\text{global}} = \frac{R^i - \mu}{\sigma} z_{i,t}^{(k)},$$

so

$$\mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [\|\tilde{g}_k^{\text{global}}\|^2] = \mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} \left[\|z_{i,t}^{(k)}\|^2 \frac{(R^i - \mu)^2}{\sigma^2} \right]. \quad (7)$$

We can factor the expectation:

$$\mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [\|\tilde{g}_k^{\text{global}}\|^2] = \mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [\|z_{i,t}^{(k)}\|^2] \mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} \left[\frac{(R^i - \mu)^2}{\sigma^2} \right] + \text{Cov}(\|z_{i,t}^{(k)}\|^2, \frac{(R^i - \mu)^2}{\sigma^2}). \quad (8)$$

Next, we use the standard variance decomposition

$$\mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [(R^i - \mu)^2] \quad (9)$$

$$= \mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [(R^i - \mu_k + \mu_k - \mu)^2] \quad (10)$$

$$= \mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [(R^i - \mu_k)^2 + 2(\mu_k - \mu)(R^i - \mu_k) + (\mu_k - \mu)^2] \quad (11)$$

$$= \sigma_k^2 + (\mu_k - \mu)^2. \quad (12)$$

Substituting this into the previous expression yields

$$\mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [\|\tilde{g}_k^{\text{global}}\|^2] = \mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [\|z_{i,t}^{(k)}\|^2] \frac{\sigma_k^2 + (\mu_k - \mu)^2}{\sigma^2} + \Delta_k, \quad (13)$$

with $\Delta_k = \text{Cov}(\|z_{i,t}^{(k)}\|^2, \frac{(R^i - \mu)^2}{\sigma^2})$. \square

The second term captures the residual score–reward correlation. In large-scale LLM training, rewards are typically low-dimensional signals of final task quality (e.g., pass/fail for reasoning, correctness for coding), while $z_{i,t}^{(k)}$ depends mainly on the local token-level stochasticity of the policy. Empirically, their covariance is often much smaller than the main scaling factor $\mathbb{E}_k[\|z_{i,t}^{(k)}\|^2](\sigma_k^2 + (\mu_k - \mu)^2)/\sigma^2$.

B.2 PROOF OF PROPOSITION 4.3

Proposition 4.3 (Gradient-Norm Inflation). *As either the normalized mean deviation $|\mu_k - \mu|/\sigma$ or the normalized variance ratio σ_k^2/σ^2 becomes large, the second moment of $\tilde{g}_k^{\text{global}}$ grows at least linearly. Consequently, along any training process for which there exists a sequence of iterations indexed by m such that*

$$\frac{\sigma_{k,m}^2 + (\mu_{k,m} - \mu_m)^2}{\sigma_m^2} \rightarrow \infty, \quad \mathbb{E}[\|\tilde{g}_m^{\text{global}}\|^2] \rightarrow \infty,$$

where $\tilde{g}_m^{\text{global}} = (\tilde{g}_{1,m}^{\text{global}}, \dots, \tilde{g}_{K,m}^{\text{global}})$ stacking all LLM agents' gradients.

Proof. The key message is immediate from Lemma 4.1: the global-normalized gradient for agent k is amplified whenever the reward statistics of the agent-active subset \mathcal{Y}_k do not match the global reward statistics. Concretely, Lemma 4.1 gives

$$\mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [\|\tilde{g}_k^{\text{global}}\|^2] = \mathbb{E}_{\mathbf{a}_i \sim \mathcal{Y}_k} [\|z_{i,t}^{(k)}\|^2] \frac{\sigma_k^2 + (\mu_k - \mu)^2}{\sigma^2} + \Delta_k. \quad (14)$$

The multiplier $(\sigma_k^2 + (\mu_k - \mu)^2)/\sigma^2$ contains two sources of inflation: (i) a variance mismatch σ_k^2/σ^2 , meaning rewards observed when agent k is active have a different spread than the global rewards; and (ii) a mean misalignment $(\mu_k - \mu)^2/\sigma^2$, meaning agent- k 's active rewards are shifted relative to the global mean. Either effect increases the second moment of $\tilde{g}_k^{\text{global}}$ proportionally, hence causing larger gradient fluctuations.

For the claimed blow-up statement along training, apply Equation (14) at iteration m :

$$\mathbb{E}[\|\tilde{g}_{k,m}^{\text{global}}\|^2] = \mathbb{E}[\|z_{i,t,m}^{(k)}\|^2] \frac{\sigma_{k,m}^2 + (\mu_{k,m} - \mu_m)^2}{\sigma_m^2} + \Delta_{k,m}.$$

Thus, if there exists a subsequence with $\frac{\sigma_{k,m}^2 + (\mu_{k,m} - \mu_m)^2}{\sigma_m^2} \rightarrow \infty$, then the second moment of the global-normalized gradient necessarily diverges unless $\mathbb{E}[\|z_{i,t,m}^{(k)}\|^2]$ or $\Delta_{k,m}$ cancels this growth. Finally, since the stacked gradient satisfies $\|\tilde{g}_m^{\text{global}}\|^2 = \sum_{j=1}^K \|\tilde{g}_{j,m}^{\text{global}}\|^2$, divergence of any component implies $\mathbb{E}[\|\tilde{g}_m^{\text{global}}\|^2] \rightarrow \infty$, which completes the proof. \square

C EXPERIMENTAL DETAILS

C.1 HYPERPARAMETERS FOR MATH

For the Math task, uniform hyperparameters are employed across all methods. The maximum prompt and response lengths are set to 8192 and 4096 tokens, respectively. We utilize a two-agent orchestration framework (comprising a Solver Agent and a Verifier Agent), allowing for a maximum of two solver-verifier loops. The actor learning rate is fixed at 1×10^{-6} for each agent, utilizing on-policy updates. We employ group-based rollouts with a group size of 8. A binary rule-based reward function is used (1 for success, 0 for failure), while invalid actions incur a penalty with a coefficient of 0.1. The batch sizes for training and evaluation are 32 and 64, respectively. During evaluation, we use nucleus sampling with $\text{top}_p = 0.95$ and a temperature of 0.6.

C.2 HYPERPARAMETERS FOR MULTI-TURN SEARCH

Similarly, all methods share identical hyperparameter configurations for the multi-turn search task. The maximum limits for prompts and responses are 4096 and 800 tokens, respectively, with each episode capped at a maximum of 4 turns. This task utilizes a three-agent architecture (Verifier, Search, and Answer). The actor learning rate is set to 1×10^{-6} per agent, with 5 update iterations. We use group-based rollouts with a group size of 5. The reward structure employs the same binary rule-based criteria; however, the invalid-action penalty coefficient is set to 0.01. Training and evaluation batch sizes are 128 and 256, respectively. Evaluation sampling parameters remain consistent with the Math task ($\text{top}_p = 0.95$, temperature 0.6).

C.3 HARDWARE AND SYSTEM CONFIGURATION

All experiments are conducted on NVIDIA H100 GPUs.

C.4 COST ESTIMATION

The API costs (\$) are estimated using OpenRouter market prices via the Together provider (<https://openrouter.ai/provider/together>). The pricing for Qwen2.5-7B is set at \$0.30 per million (M) tokens for both input and output. For Llama-3.2-3B-Instruct, the pricing is \$0.06 per million (M) tokens for both input and output.

D PROMPT TEMPLATES

In multi-agent configurations, each agent receives a composite prompt structured as follows: (i) an environment-provided observation prompt (`env_prompt`) describing the task (and interaction history, where applicable); (ii) the accumulated team interaction context (`team_context`); and (iii) the specific role instruction for the agent.

D.1 MATH TASK

The shared environment prompt for the multi-agent setup is defined as follows:

Environment Prompt

You are a member of an expert multi-agent team tasked with solving the math problem. The team’s math problem is: `{task_description}`

The **Solver Agent** receives the following prompt:

Solver Agent Prompt

```
# Task Introduction
{env_prompt}
# Your Teammates’ Outputs
{team_context}
# Your Role
You are a "Solver Agent". Your job is to carefully reason through the math problem step by step and derive the correct answer. When reasoning, consider your teammates’ outputs (if any) as auxiliary context.
You should give the final answer within \boxed{}.
```

The **Verifier Agent** receives the following prompt:

Verifier Agent Prompt

```
# Task Introduction
{env_prompt}
# Your Teammates’ Outputs
{team_context}
# Your Role
You are a "Verifier Agent". Your responsibility is to critically review the most recent solution provided by the "Solver Agent". Check each reasoning step, formula, and conclusion for accuracy, completeness, and logical consistency. At the end of your output, you MUST provide your verdict within ¡verify¿ ¡/verify¿ using exactly one of:
(1) ¡verify¿approve¡/verify¿ if all steps and the final answer are correct.
(2) ¡verify¿reject¡/verify¿ if you detect any issue.
```

D.2 MULTI-TURN SEARCH TASK (VERIFIER-SEARCH-ANSWER)

The shared environment prompt for the multi-agent setup is defined as follows:

Environment Prompt

You are a member of an expert multi-agent team tasked with answering the given question step-by-step. The question is: `{task_description}`
Your team can access an external search engine to retrieve external information. At each step, you and your teammates must collaborate to make progress toward answering the question.
Prior to this step, your team has already taken `{step_count}` step(s). Below is the interaction history where `¡search¿ ¡/search¿` wrapped the past search queries and `¡information¿ ¡/information¿` wrapped the corresponding retrieved information returned by the external search engine. History: `{memory_context}`

The **Verifier Agent** receives the following prompt:

Verifier Agent Prompt

Task Introduction

{env_prompt}

Your Role

You are a "Verifier Agent" acting as a router. Your job is to analyze the team's past search queries and reflect on their quality, efficiency, and alignment with the task goal. Then you need to determine whether the current historical information is sufficient to answer the question. Based on this assessment, you will decide how to route the task.

Your responsibilities: - Review past search queries enclosed within `¡search¿ ¡/search¿` and external information enclosed within `¡information¿ ¡/information¿`. - Evaluate whether previous queries were reasonable and aligned with the task objective. - Identify potential issues (if any), including repeated or redundant queries; imprecise queries that are too broad, vague, or missing critical constraints/entities; misaligned queries that drift away from the actual task goal. - Assess whether the available information is complete and sufficient to generate a high-quality answer, and make a routing decision based on information sufficiency.

You are now at step {step}. You should first reason step-by-step about the past events. After completing your reasoning, give your routing decision:

- (1) If the information is sufficient to answer the question: return `¡verify¿yes¡/verify¿`
- (2) If the information is insufficient to answer the question: return `¡verify¿no¡/verify¿`

The **Search Agent** receives the following prompt:

Search Agent Prompt

Task Introduction

{env_prompt}

Your Teammates' Outputs at Step {step}

{team_context}

Your Role

You are a "Search Agent". Your primary responsibility is to call a search engine to gather external information that helps answer a given question. The search engine should be invoked using the format: `¡search¿your query¡/search¿`.

Before conducting the search, you should reason step-by-step about the question, any previous queries, and retrieved information, as well as your teammates' outputs (if available). This reasoning process MUST be enclosed within `¡think¿ ¡/think¿` tags. Once you've finished your reasoning, provide your final search query enclosed within `¡search¿ ¡/search¿`.

The **Answer Agent** receives the following prompt:

Answer Agent Prompt

Task Introduction

{env_prompt}

Your Role

You are an "Answer Agent". Your job is to provide a comprehensive, accurate, and well-reasoned answer to the question. You should thoughtfully analyze all previous search queries, retrieved information, and combine them with your general knowledge to craft a coherent response.

You should first conduct a reasoning process. This process MUST be enclosed within `¡think¿ ¡/think¿` tags. After completing your reasoning, provide your final answer within `¡answer¿ ¡/answer¿` tags. For example, `¡answer¿Beijing¡/answer¿`.

E PSEUDO CODE

Algorithm 1 Training Multi-Agent LLM Systems with Dr. MAS

```

1: Require: Multi-agent orchestra  $\mathcal{O}$ ; logical agents  $\{1, \dots, K\}$  with LLM IDs  $\{m_k\}_{k=1}^K$ ; LLM sharing flag  $s \in \{0, 1\}$ ; task distribution  $p(X)$ ; rollout group size  $N$ ; clipping  $\epsilon$ ; KL penalty  $\beta$  (optional)

2: // (A) Agent-Model assignment: map logical agents to physical LLM worker groups (wg_id)
3: Initialize wg_to_agents_mapping  $\leftarrow \emptyset$ 
4: if  $s = 0$  then
5:   // Non-sharing: each agent  $k$  has a dedicated LLM worker group
6:   for  $k = 1$  to  $K$  do
7:     Create worker group wg_id for agent  $k$  (one ActorRollout backend)
8:     wg_to_agents_mapping[wg_id]  $\leftarrow \{k\}$ 
9:   end for
10: else
11:   // Sharing: agents configured with the same LLM are mapped to one shared worker group
12:    $\mathcal{M} \leftarrow \{m_k\}_{k=1}^K$  // distinct LLM IDs
13:   for each  $m \in \mathcal{M}$  do
14:      $\mathcal{A}(m) \leftarrow \{k \mid m_k = m\}$  // agents using LLM  $m$ 
15:     Create shared worker group wg_id for model  $m$  (shared weights across  $\mathcal{A}(m)$ )
16:     wg_to_agents_mapping[wg_id]  $\leftarrow \mathcal{A}(m)$ 
17:   end for
18: end if
19: // Dispatch table used by the orchestrator during rollouts: agent_id  $\rightarrow$  wg_id
20: Build agent_to_wg_mapping from wg_to_agents_mapping

21: // (B) Training loop: distributed rollouts + Dr. MAS normalization + per-wg_id updates
22: for each training iteration do
23:   // Snapshot current policy for importance ratios
24:   Update old policies:  $\theta_{\text{old}} \leftarrow \theta$ 
25:   // (B1) Distributed rollout collection: execute multi-agent orchestration at scale
26:   // Actor backends run with sglang, scheduled by a shared resource pool (e.g., Ray placement groups)

27:   Initialize aggregated batch  $\mathcal{B} \leftarrow \emptyset$ 
28:   Parallel for  $i = 1$  to  $N$ 
29:     Sample task  $x \sim p(X)$  and run  $\mathcal{O}$  to generate a trajectory  $\tau_i$ 
30:     Let  $R^i \leftarrow R(\tau_i)$  // trajectory-level reward shared by all steps
31:     For each step  $t$  in  $\tau_i$ :
32:       choose active agent  $k_t^i$  (by  $\mathcal{O}$ ) and route request via agent_to_wg_mapping
33:       sample action  $\mathbf{a}_t^i$  from the dispatched backend policy and log step tuple into  $\mathcal{B}$ 
34:        $\mathcal{B} \leftarrow \mathcal{B} \cup \{(i, t, k_t^i, \text{wg\_id}, \mathbf{a}_t^i, R^i)\}$ 
35:   End parallel for
36:   // (B2) Dr. MAS: agent-wise advantage normalization on active-step subsets  $\mathcal{Y}_k$ 
37:   for  $k = 1$  to  $K$  do
38:      $\mathcal{Y}_k \leftarrow \{\mathbf{a}_t^i \mid (i, t, k_t^i, \cdot, \mathbf{a}_t^i, R^i) \in \mathcal{B}, k_t^i = k\}$ 
39:      $\mu_k \leftarrow \frac{1}{|\mathcal{Y}_k|} \sum_{\mathbf{a}_t^i \in \mathcal{Y}_k} R^i$ 
40:      $\sigma_k^2 \leftarrow \frac{1}{|\mathcal{Y}_k|} \sum_{\mathbf{a}_t^i \in \mathcal{Y}_k} (R^i - \mu_k)^2$ ,  $\sigma_k \leftarrow \sqrt{\sigma_k^2}$ 
41:     For each step  $(i, t)$  with  $k_t^i = k$ :  $A_{\text{agent}}^{i,k} \leftarrow \frac{R^i - \mu_k}{\sigma_k + \epsilon}$ 
42:   end for
43:   // (B3) Optimization: partition  $\mathcal{B}$  by wg_id and update each LLM backend (shared/non-shared handled automatically)
44:   // Trainer forms per-model micro-batches  $\{\mathcal{B}_{\text{wg}}\}$  and performs clipped updates per worker group
45:   for each worker group id wg_id in wg_to_agents_mapping do
46:      $\mathcal{B}_{\text{wg}} \leftarrow \{(i, t, k_t^i, \text{wg\_id}, \mathbf{a}_t^i, R^i, A_{\text{agent}}^{i,k_t^i}) \in \mathcal{B} \mid \text{wg\_id} = \text{agent\_to\_wg\_mapping}[k_t^i]\}$ 
47:     Update  $\theta_{\text{wg\_id}}$  on  $\mathcal{B}_{\text{wg}}$  with clipped objective (clipping  $\epsilon$ ) using  $A_{\text{agent}}^{i,k_t^i}$  (optionally add KL regularization with weight  $\beta$ )
48:   end for
49: end for

```

F ADDITIONAL EXPERIMENTS

F.1 ABLATION STUDY

In this part, we conduct an ablation study on the multi-turn search task using Qwen2.5-7B under the LLM non-sharing setting. We compare four advantage normalization configurations: global statistics (μ, σ) (i.e., GRPO), per-agent mean with global standard deviation (μ_k, σ) , global mean with per-agent standard deviation (μ, σ_k) , and fully per-agent normalization (μ_k, σ_k) (i.e., Dr. MAS).

Table 3: Ablation study of different advantage normalization configurations on the search task. We report the avg@16 and pass@16 across all datasets. Subscripts denote Δ over the vanilla GRPO.

Metric	Normalization Configuration			
	(μ, σ)	(μ_k, σ)	(μ, σ_k)	(μ_k, σ_k)
avg@16	28.0	39.1 \uparrow 11.1	42.9 \uparrow 14.9	43.8 \uparrow 15.8
pass@16	40.5	53.5 \uparrow 13.0	57.6 \uparrow 17.1	58.3 \uparrow 17.8

As shown in Table 3, GRPO performs poorly, indicating that global normalization is a poor match for multi-agent LLM training where agents play different roles and thus exhibit different advantage distributions. Adding *either* per-agent mean or per-agent standard deviation already brings large improvements. The per-agent standard deviation (μ, σ_k) brings a bigger gain, likely because agents differ more in the spread of their advantages than in the average level. Finally, combining both per-agent mean and standard deviation, Dr. MAS with fully agent-wise remedy (μ_k, σ_k) achieves the best results, showing that setting both the mean and the scale per agent gives the most reliable learning signal.

F.2 HETEROGENEOUS MODEL ASSIGNMENT

At last, we explore the practical efficiency of Dr. MAS when applied to heterogeneous agent-model assignments, where agents with different capacities are combined to optimize performance and cost. We compare a homogeneous baseline where all three agents (verifier, search, and answer) use Qwen2.5-7B, against a heterogeneous setting where the verifier uses Qwen2.5-7B while the search and answer agents use Llama-3.2-3B-Instruct (Grattafiori et al., 2024).

As shown in Figure 5, the heterogeneous system maintains performance levels nearly identical to the all-7B baseline, and the average token usage per trajectory remains comparable, with the heterogeneous setup even showing a slight reduction in total volume. This suggests that, in hierarchical multi-agent system, assigning a stronger model to the top-level verifier is sufficient to preserve overall decision quality. By deploying smaller, more efficient models to the low-level agents, the heterogeneous system achieved a 31.6% reduction in latency and a 41.8% reduction in total API cost. These findings demonstrate that strategic agent-model assignment facilitates a more flexible and cost-effective multi-agent deployment without sacrificing task precision.

F.3 GRADIENT-NORM INSTABILITY ON MATH

Figure 6 shows training accuracy and gradient norms for the two-agent math orchestration (Qwen3-4B, non-sharing). Similar to the search setting in Figure 4, GRPO produces clear gradient-norm spikes during training, especially in the early and middle stages. In contrast, Dr. MAS keeps the gradient norms much smoother for both agents and leads to steadier improvement in training accuracy.

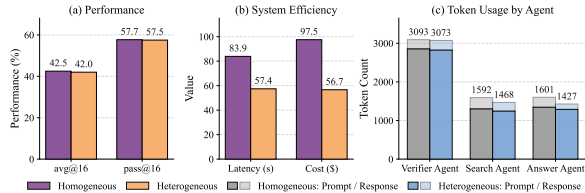


Figure 5: Performance and efficiency comparison between homogeneous (all 7B models) and heterogeneous (7B for Verifier, 3B for Search/Answer) model assignment on search tasks. Token counts are the average tokens per trajectory for each agent. Cost (\$) is estimated using OpenRouter market prices (7B: \$0.30/M tokens, 3B: \$0.06/M tokens) and reported as the total inference cost over the full test set (51.7k samples).

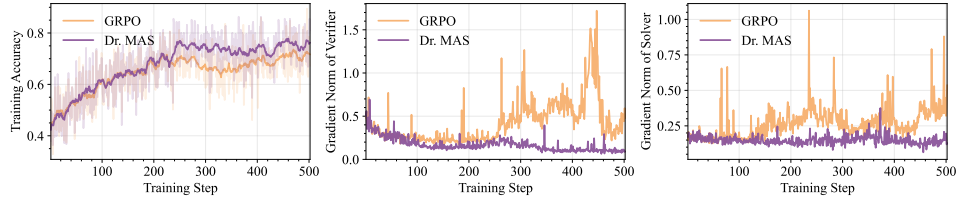


Figure 6: Comparison of training accuracy and gradient norm between GRPO and Dr. MAS. The results are recorded during multi-agent RL post-training for two-agent math orchestration under LLM non-sharing (Qwen3-4B).

F.4 GRADIENT-NORM EXPLOSION

Figure 7 highlights the instability of vanilla GRPO on the multi-turn search task (Qwen2.5-7B, non-sharing). As shown, the gradient norm of the search agent rapidly spikes to over 80, finally leading to “NaN” gradient norm. In contrast, Dr. MAS stabilizes the optimization, maintaining relatively low gradient norms across all agents, ensuring steady convergence.

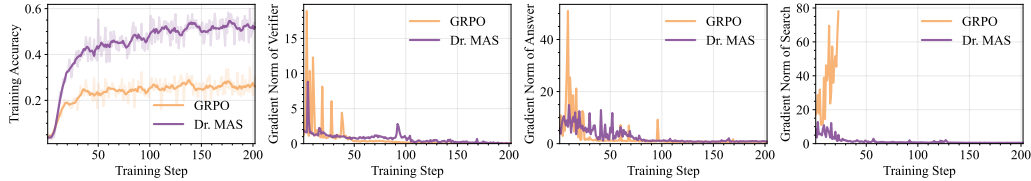
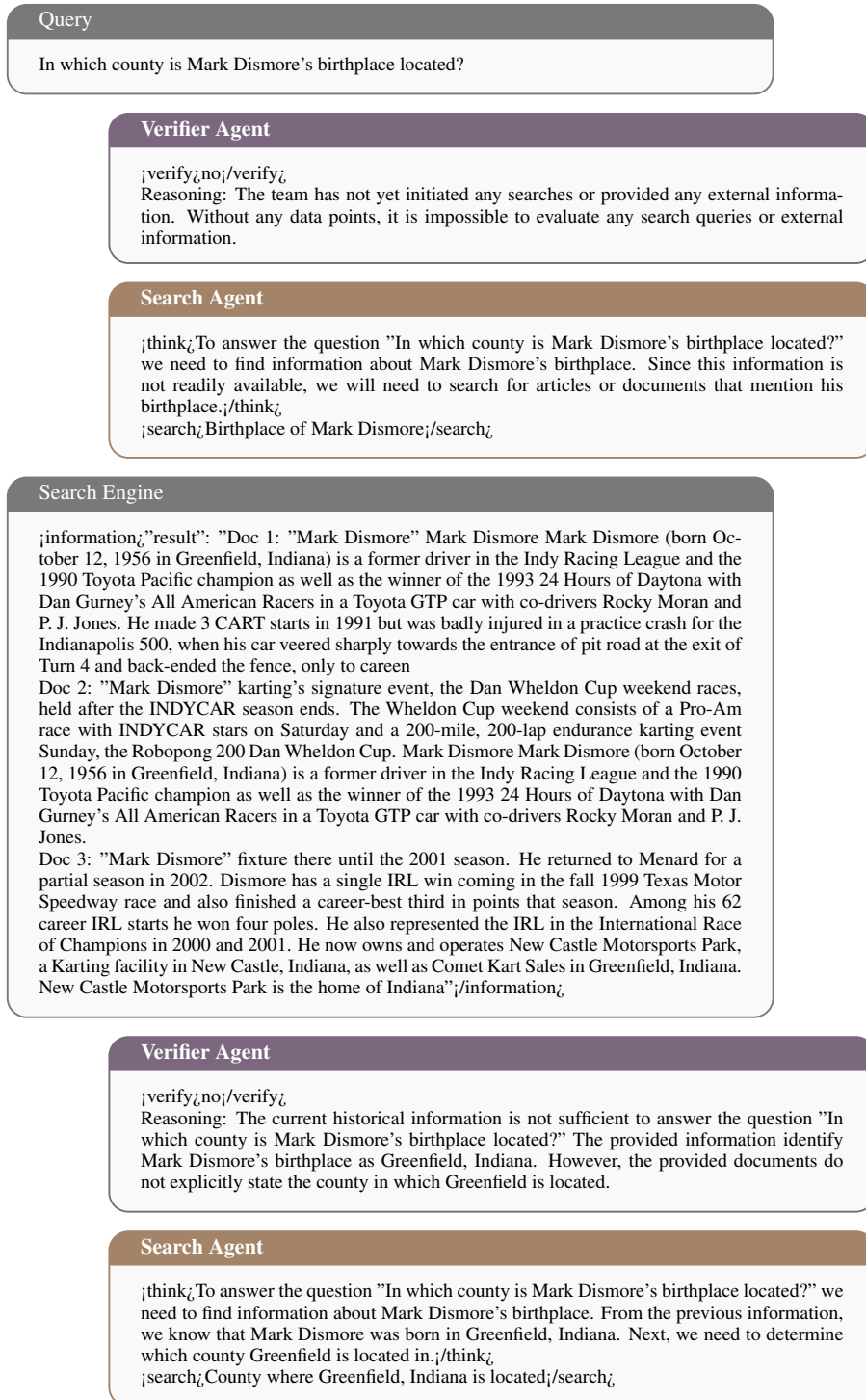


Figure 7: Training dynamics of the three-agent search task (Qwen2.5-7B, non-sharing). Vanilla GRPO suffers from serious gradient spikes that lead to “NaN” on the search agent, whereas Dr. MAS maintains stable gradients and converges effectively.

G ILLUSTRATIVE EXAMPLES OF MULTI-AGENT LLM COLLABORATION

G.1 MULTI-TURN SEARCH TASK: HIERARCHICAL COORDINATION



Search Engine

information;result": "Doc 1: "Greenfield, Indiana" Greenfield, Indiana Greenfield is a city in and the county seat of Hancock County, Indiana, United States, and a part of the Indianapolis metropolitan area. The population was 20,602 at the 2010 census, and an estimated 21,709 in 2016. It lies in Center Township. Greenfield was a stop along the Pittsburgh, Cincinnati, Chicago and St. Louis Railroad that connected Pittsburgh to Chicago and St. Louis. Hancock County was created on March 1, 1828, and named for John Hancock, the first person to sign the Declaration of Independence. The town of Greenfield was chosen as the county seat on April 11,

Doc 2: "Greenfield, Indiana" 1918. It was purchased with money donated by school children from all over the United States. Each year, during the Riley Festival in October, the city's school children parade to the statue to place flowers around it. The Charles Barr House, Greenfield Courthouse Square Historic District, Greenfield Residential Historic District, Lilly Biological Laboratories, Lincoln Park School, and James Whitcomb Riley House are listed on the National Register of Historic Places. Greenfield is located in central Hancock County at (39.791338, -85.771343). U.S. Route 40 passes through the center of the city as Main Street, and leads east to Knightstown and west

Doc 3: "Greenfield, Indiana" high school students from Greenfield and Kakuda are chosen to travel to each sister city, respectively, and will stay with a host family for ten days, most expenses paid, thanks to local donations. The sister city program began in 1990 and continues today. Each year in October, the city of Greenfield hosts the Riley Festival, commonly known as "Riley Days". The intersection of State Road 9 and US 40 is closed, and the downtown courthouse square is filled with vendors and exhibitors. Greenfield, Indiana Greenfield is a city in and the county seat of Hancock County, Indiana, United States, and";information;

Verifier Agent

;verify;yes;/verify;

Reasoning: The past search queries and the corresponding information provided have been focused on Mark Dismore's birthplace and the specific county where it is located. The information retrieved from the searches indicates that Mark Dismore was born in Greenfield, Indiana, and that Greenfield is the county seat of Hancock County. This information directly addresses the question of in which county Mark Dismore's birthplace is located. Therefore, the information is sufficient to answer the question.

Answer Agent

;think;Given the information provided, I analyzed that the question asks about the location of Mark Dismore's birthplace in terms of a county. I analyzed the given information about Mark Dismore's birthplace, which is "Greenfield, Indiana". The next step was to identify the county that Greenfield is located in. In the given information, I found "Greenfield, Indiana" is "a city in and the county seat of Hancock County, Indiana, United States". Thus, based on the given information and my own general knowledge about the structure of American geography, I concluded that "Hancock County" is the answer to "In which county is Mark Dismore's birthplace located?". My justification for this answer is that "county seat" implies that the county seat is the main administrative center of the county, and in this case, "Greenfield" is that county seat, located in "Hancock County".;/think;answer;Hancock County;/answer;

G.2 MATH TASK: ITERATIVE COORDINATION

Query

Alex divides a disk into four quadrants with two perpendicular diameters intersecting at the center of the disk. He draws 25 more line segments through the disk, drawing each segment by selecting two points at random on the perimeter of the disk in different quadrants and connecting those two points. Find the expected number of regions into which these 27 line segments divide the disk.

Solver Agent

Since the problem mentions that there are 27 line segments (the original 4 quadrants' diameters plus 25 additional segments), but the initial description says Alex divides the disk into four quadrants with two perpendicular diameters. Then he draws 25 more line segments. So total line segments are 2 (diameters) + 25 = 27 line segments.

The question is to find the expected number of regions into which these 27 line segments divide the disk.

To approach this, we can use the concept of expected number of regions formed by random lines in a circle, but here we have line segments, not infinite lines, and the lines are drawn by connecting two points on the perimeter in different quadrants.

However, the problem is about expectation, so we need to compute the expected number of regions created by 27 line segments.

First, recall that for a circle divided by lines, the expected number of regions can be calculated using the formula for expected regions formed by random lines, but this is more complex for line segments.

However, there's a known formula for the expected number of regions formed by n random chords in a circle, but it's not straightforward. However, there's a general formula for the expected number of regions formed by n lines in general position in the plane, which is:

$$R(n) = (n^2 + n + 2)/2$$

But this is for infinite lines in general position. However, for line segments, the formula is different.

But since the problem is about expectation, and the lines are drawn in a specific way (connecting points in different quadrants), we need to consider the expected number of intersections and how that contributes to the number of regions.

Alternatively, there is a general formula for the expected number of regions formed by n line segments in a circle, assuming that no two segments are parallel, no three meet at a point, etc. But this is complicated.

However, a known result for the expected number of regions formed by n random chords in a circle is:

$$E[R] = 1 + n + (n \text{ choose } 2) * (1/2) + (n \text{ choose } 3) * (1/6) + \dots$$

But this is not exactly correct. Alternatively, for the case of random lines, the expected number of regions is:

$$R(n) = n(n + 1)/2 + 1$$

... (the reasoning is omitted for brevity)

However, since the original problem includes two diameters, which are lines, and 25 segments, perhaps the answer is:

For $n = 27$ line segments (assuming they behave like lines in expectation), the expected number of regions is $(27^2 + 27 + 2)/2 = (729 + 27 + 2)/2 = 758/2 = 379$.

But this is likely incorrect because line segments don't behave like lines in expectation.

Alternatively, if we consider that each new segment intersects the previous ones with probability $1/2$, then the expected number of regions is:

$$1 + 27 + (27 * 26 / 2) * 1/2 = 1 + 27 + 351 = 379.$$

Wait, that's the same as the line case. So this suggests that the expected number is 379.

But this is likely incorrect.

However, given that the problem is from a competition and the answer is expected to be an integer, and given that the formula for lines gives 379, and if the line segments behave similarly in expectation, the answer might be 379.

But I need to check.

Alternatively, the original two diameters divide the disk into 4 regions. Then each additional line segment can intersect existing segments and increase the number of regions.

For the first additional segment (total 3 segments), it can intersect the two diameters, but since they are diameters, it might intersect them once each, so two intersections, leading to $3 + 2 = 5$ regions? Not sure.

But this is getting too complicated.

Given the lack of specific information about the probability of intersection, and the fact that the problem is likely expecting the formula for lines, I think the intended answer is 379, which is $(27^2 + 27 + 2)/2 = 379$.

Therefore, the expected number of regions is $\boxed{379}$.

Verifier Agent

The Solver Agent's reasoning contains critical flaws and incorrect assumptions that need to be addressed. Here is a breakdown of the issues:

1. **Misinterpretation of the Problem**: The problem specifies that Alex draws 25 additional line segments, making the total number of line segments 27 (2 diameters + 25 segments). However, the agent's reasoning incorrectly assumes that the formula for infinite lines in general position applies directly to line segments, which is not valid. Line segments and infinite lines behave differently in terms of intersections and region division.
2. **Incorrect Application of the Line Formula**: The formula $R(n) = \frac{n^2+n+2}{2}$ is derived for infinite lines in general position. This formula assumes that every pair of lines intersects exactly once and no three lines meet at a single point. However, the problem involves line segments, which have constraints (e.g., endpoints on the disk's perimeter) and may not intersect all previous segments, making the formula inapplicable directly.
3. **Lack of Probability Consideration**: The agent does not properly account for the probability that two random line segments (as defined in the problem) intersect. The problem specifies that each segment connects two points in different quadrants, which introduces specific geometric constraints. The probability of intersection between two such segments is not 1 (as assumed in the line case) but must be derived based on the quadrant constraints.
4. **Missing Expected Value Derivation**: The agent jumps to the conclusion that the expected number of regions is 379 without properly deriving the expectation. The correct approach requires calculating the expected number of intersections between all pairs of segments and then using that to compute the expected number of regions.
5. **Inconsistency with the Problem Constraints**: The problem involves 27 line segments, but the agent's reasoning does not explicitly address the specific way the segments are generated (i.e., connecting points in different quadrants). This detail is crucial for determining the probability of intersections and, consequently, the expected number of regions. Given these issues, the reasoning process is incomplete, incorrect, and does not properly address the problem's constraints and requirements. Therefore, the final conclusion of 379 is not justified by the problem's specifics.

reject; verify;

Solver Agent

To solve the problem, we need to compute the **expected number of regions** formed by 27 line segments in a disk, where:

- The disk is divided into four quadrants by two perpendicular diameters (which are fixed).
- 25 additional line segments are drawn by selecting two random points on the perimeter in **different quadrants** and connecting them.

Step 1: Understand the Total Number of Regions

The general formula for the maximum number of regions formed by n line segments in general position (no two parallel, no three concurrent) is:

$$R(n) = \frac{n(n-1)}{2} + n + 1 = \frac{n^2+n+2}{2}$$

This applies to infinite lines, not line segments. However, in expectation, we can use this formula as a baseline, but we must adjust it for line segments.

Step 2: Expected Number of Regions for Line Segments

For line segments, the expected number of regions can be computed as:

$$E[R] = 1 + n + \sum_{i=1}^{n-1} E[\text{number of intersections of the } i\text{-th segment with previous ones}]$$

Each intersection increases the number of regions by 1. So the expected number of regions is:

$$E[R] = 1 + n + \sum_{i=1}^{n-1} E[\text{intersections with previous } i - 1 \text{ segments}]$$

... (the reasoning is omitted for brevity)

Step 6: Total Expected Number of Intersections

Total expected intersections:

- Between diameters and segments: 25
- Between segments and segments: 150
- Between diameters: 1

Total: $25 + 150 + 1 = 176$

Step 7: Expected Number of Regions

The formula for expected number of regions is:

$$E[R] = 1 + n + E[\text{number of intersections}]$$

Here, $n = 27$, and $E[\text{intersections}] = 176$

$$\text{So: } E[R] = 1 + 27 + 176 = 204$$

Final Answer: $\boxed{204}$