RAPO: RESPONSIBILITY ATTRIBUTION-DRIVEN PROMPT OPTIMIZATION FOR LLM-BASED MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based multi-agent systems show strong potential for complex collaborative tasks, but their performance depends heavily on the design and optimization of role prompts. Existing optimization methods typically rely on label-free feedback and randomly sampled task cases, which can introduce misaligned signals and agent-irrelevant noise, leading to unstable prompt updates. We propose a Responsibility Attribution-driven Prompt Optimization (RAPO) framework for multi-agent systems. Its two key designs—replacing unreliable LLM self-evaluation with gold-answer guidance and constructing agent-specific sample sets via error attribution—jointly enable stable prompt updates with smaller sample budgets. A formal bias–variance analysis justifies their complementary roles. Extensive experiments on code generation, mathematical reasoning, and scientific QA show that our method outperforms the state-of-the-art MASPO, and ablation studies confirm each component's contribution. Cross-backbone evaluations further demonstrate the stability and generality of the approach.

1 INTRODUCTION

Recent advances in large language models (LLMs) (OpenAI, 2023) have given rise to increasingly capable task-solving systems. LLM-based multi-agent systems (MASs) extend the single-agent paradigm by assigning specialized roles (Li et al., 2023) to multiple agents. These systems have demonstrated strong performance on code generation (Mao et al., 2025), mathematical reasoning (Liang et al., 2024) and scientific question answering (Smit et al., 2024).

Despite these promising results, it is widely known that the prompt assigned to each agent in MASs has a substantial impact on overall performance (Li et al., 2023), making prompt optimization critically important. Although prompt optimization has been extensively studied for single LLMs (Fernando et al., 2024; Guo et al., 2025; Agrawal et al., 2026), optimizing prompts in MASs is particularly challenging. This is because updating one agent's prompt can affect its own behavior and downstream agents' inputs or decisions (Wang et al., 2026). Thus, recent work has increasingly focused on modeling inter-agent dependencies during the optimization process. For instance, MASPO (Wang et al., 2026) models inter-agent dependencies through rewards for local adherence, downstream utility and global alignment; MAPRO (Zhang et al., 2026) represents prompt dependencies as node and edge factors over the agent topology and propagates them via max-product belief propagation; Under budget constraints, MASPOB (Hong et al., 2026) learns a GNN-based surrogate over the agent graph to capture topology-induced coupling during bandit search. Despite their effectiveness, these methods still suffer from two key limitations.

First, existing methods mainly rely on label-free feedback signals, such as LLM-based preference judgments, language-guided node/edge scores, or black-box system-level execution feedback, to estimate the quality of candidate prompts. Such feedback does not directly measure whether a candidate improves task correctness with respect to the gold answer. As a result, the optimization signal may be misaligned with the target task, especially when the model lacks sufficient task knowledge or when local improvements do not translate into final-answer gains. Second, when optimizing a target agent, these methods depend on a small set of task samples to guide prompt optimization; yet, they do not verify whether the errors in those outputs actually originate from the target agent. Conse-

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.

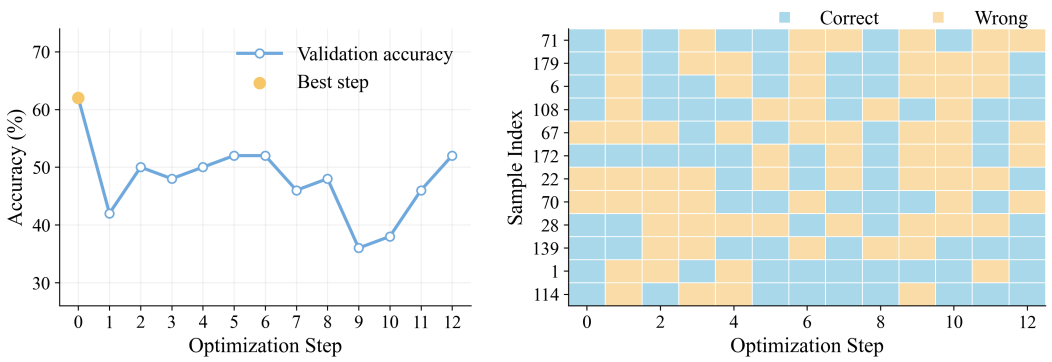


Figure 1: Performance of MASPO on the GPQA dataset. Left: validation accuracy fluctuates across optimization rounds. Right: sample correctness flips across optimization rounds.

quently, they may include irrelevant samples or errors propagated from upstream agents, which can mislead prompt updates. Together, these two limitations result in suboptimal optimization outcomes.

Taking the recent state-of-the-art method MASPO as an example, Figure 1 illustrates its performance during the prompt optimization process. On the GPQA (Rein et al., 2023) dataset, validation accuracy fluctuates substantially across optimization rounds. The right panel further reveals that an update may correct some errors while breaking previously correct samples. These observations indicate that the optimization process suffers from substantial instability.

Based on these observations, we propose a **Responsibility Attribution-driven Prompt Optimization** (RAPO) method for MASs. Our approach comprises two key designs. First, it introduces gold answer labels for a small set of task samples. By evaluating whether candidate prompts improve final-answer correctness, the method provides more reliable signals to guide prompt updates. Second, it attributes errors in failed trajectories to the responsible agents. When optimizing the prompt of a target agent, it uses those cases where the errors are attributed to that agent. By filtering out irrelevant cases, this enables more targeted and stable prompt optimization.

Furthermore, we provide a theoretical analysis that explains the rationale behind our method's two key designs. This analysis offers a bias–variance perspective on estimating the performance gain of candidate prompts. Specifically, gold-answer guidance reduces feedback bias by aligning evaluation signals with the true task objective, while error attribution reduces variance by selecting only those failed trajectories that are relevant to the target agent. Together, these two account for the more stable optimization under a limited sample budget. Our contributions can be summarized as follows:

1. We propose a responsibility attribution-driven prompt optimization method that replaces unreliable LLM self-evaluation with gold-answer-guided rewards and uses error attribution to construct agent-specific optimization samples. By filtering misleading feedback and irrelevant cases, it achieves more stable optimization with smaller update batches.
2. We provide a bias–variance analysis that formally explains how our two key designs interact to improve optimization stability. Gold-answer guidance reduces the bias of the optimization signal, while error attribution filtering reduces the variance caused by irrelevant samples, offering a principled rationale for the method's effectiveness.
3. Across diverse tasks—code generation, math reasoning, and scientific QA—our method outperforms MASPO on most benchmarks. Further analysis shows that our method improves the stability of the optimization process. Ablation studies confirm the contribution of each component, and cross-backbone experiments further validate its generality.

2 Related Work

Multi-Agent Prompt Optimization. Existing multi-agent prompt optimization methods typically follow a pipeline of candidate generation, system-level evaluation, and prompt selection. MASPO (Wang et al., 2026) optimizes prompt configurations through multi-granularity rewards

and trajectory-guided search, while MAPRO (Zhang et al., 2026) and MASPOB (Hong et al., 2026) model inter-prompt dependencies and optimization as structured search problems. MAS-PromptBench (Bai & Shi, 2026) further studies the conditions under which prompt optimization benefits MASs. Recent works also explore contribution estimation and credit assignment for multi-agent prompt optimization (Xia et al., 2026; Shen et al., 2025; Seo et al., 2026).

Although these methods address MAS prompt optimization, they differ in optimization signals and settings. Following the fixed-topology role-prompt optimization setting (Wang et al., 2026), we introduce two key differences: gold-answer supervision for final-task evaluation and responsibility-conditioned sample construction for agent-specific prompt updates. These designs improve optimization stability by providing more reliable feedback and targeted optimization signals.

Error Attribution. Error attribution in MASs aims to identify the agents or steps responsible for failures in complex trajectories. Existing studies explore automatic attribution through failure-step localization, such as Who&When (Zhang et al., 2025), coarse-to-fine search strategies such as SDBL (Sun et al., 2026), and temporal-semantic trajectory analysis such as StepFinder (Zhu et al., 2026). Unlike prior work that focuses on locating the source step that leads to failure (step-level), we focus on attributing errors in failed trajectories to the responsible agents from a sample-selection perspective (instance-level)—we use responsibility attribution to filter optimization samples.

Self-Evolution. Self-evolution focuses on improving LLM-based systems through self-generated or environmental feedback. For instance, Self-Refine (Madaan et al., 2023) iteratively revises outputs using its own feedback, while Reflexion (Shinn et al., 2023) stores feedback as reusable natural-language reflections. At the prompt and workflow levels, OPRO (Yang et al., 2024) generates candidate prompts from optimization histories and scores, and TextGrad (Yüksekgönül et al., 2024) uses textual gradients to optimize prompts, code, and other language-based variables. Prior multi-agent prompt optimization methods such as MASPO (Wang et al., 2026), MAPRO (Zhang et al., 2026), and MASPOB (Hong et al., 2026) can be understood within this broader self-evolution paradigm, as they automatically refine prompt configurations using label-free feedback signals.

3 Task Definition

Given a MAS driven by LLMs, its topology is modeled as a directed acyclic graph $G = (V, E)$, where $V = \{v_i\}_{i=1}^{N}$ denotes the node set consisting of N agents, and E denotes the information-passing relations among agents. Each agent v_i consists of an LLM M_i and a role prompt p_i. The prompts of all agents constitute the prompt configuration: $P = \{p_i\}_{i=1}^{N}$.

For an input question q, each agent is executed according to the topological order of G. The question q is provided to each node as a global input. Agent v_i receives the local context C_i, which is formed by the outputs of its direct predecessor nodes, and generates: $o_i = M_i(q, C_i, p_i)$. The final system output is uniformly denoted as: $o_{\text{glob}} = \Phi(q; G, P)$, where Φ denotes the composite execution process jointly induced by the multi-agent topology and agent inference functions.

On a benchmark dataset $\mathcal{D} = \{(q_k, y_k)\}_{k=1}^{K}$, the goal is to optimize the prompt configuration without updating the parameters of LLMs, so as to maximize the consistency between the system output and the gold answer: $\max_{P \in \mathcal{P}^N} \frac{1}{K} \sum_{k=1}^{K} R_{\text{task}}\big(\Phi(q_k; G, P), y_k\big)$, where $\mathcal{P}$ denotes the prompt space, and R_{task} denotes the task score computed from the system output and the reference answer.

4 Method

This section introduces our RAPO framework for MAS. Figure 2 provides an overview of it.

4.1 Gold-Guided Error Sample–Agent Assignment

In this stage, the MAS is evaluated on a small training set against gold answers. Trajectories are partitioned into correct and incorrect cases, and each failure is attributed to the agents responsible.

Gold-Guided Trajectory Partitioning. Given a multi-agent topology G and a prompt configuration P_t, the system performs inference on a small training set $\mathcal{D}_{\text{train}} = \{(q_k, y_k)\}_{k=1}^{N}$, where q_k denotes the input question and y_k is its corresponding gold answer. For each sample (q_k, y_k), it produces a

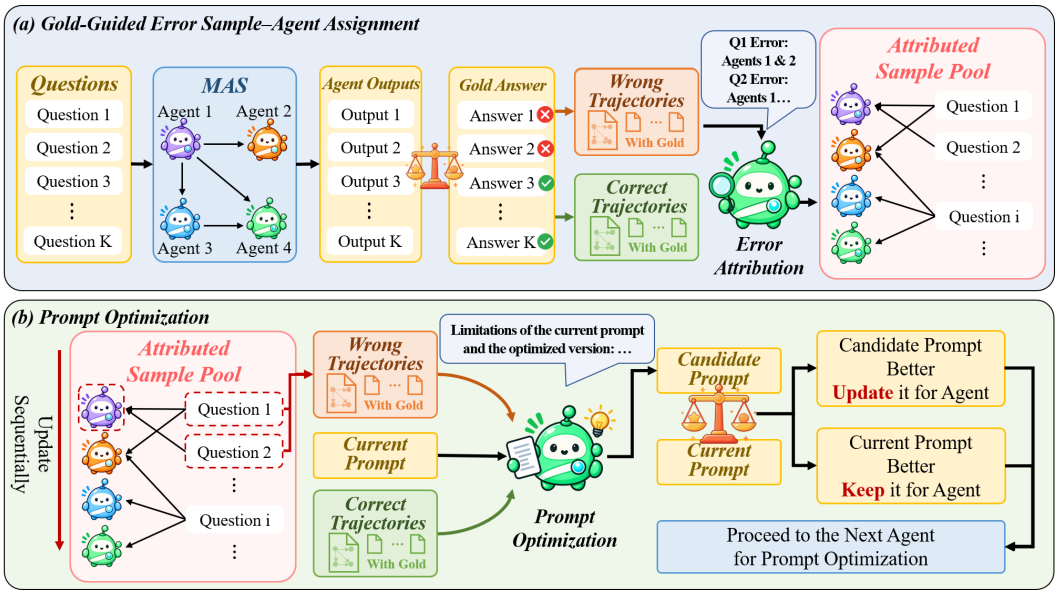


Figure 2: Overview of our RAPO framework. (a) Gold-Guided Error Sample–Agent Assignment: trajectories are evaluated against gold answers; erroneous ones are then attributed to the responsible agents, forming attributed sample pool. (b) Prompt Optimization: each agent is sequentially optimized using its assigned samples.

global output $o_{\text{glob},k,t} = \Phi(q_k; G, P_t)$, while recording the local context and output of each agent, together with the complete execution trajectory $\tau_{k,t}$.

The final output is then evaluated against the gold answer y_k from the training set to determine its correctness: $r_{k,t} = R_{\text{task}}(o_{\text{glob},k,t}, y_k)$, where $r_{k,t} = 1$ indicates a correct trajectory and 0 an incorrect one. Based on $r_{k,t}$, the execution trajectories are partitioned into correct and incorrect sets:

$$\mathcal{T}_t^+ = \{\tau_{k,t} \mid r_{k,t} = 1\}, \quad \mathcal{T}_t^- = \{\tau_{k,t} \mid r_{k,t} = 0\}.$$

Attributed Sample Pool Construction. For each incorrect trajectory $\tau_{k,t} \in \mathcal{T}_t^-$, an attribution LLM A_{LLM} identifies the responsible agents by examining intermediate contexts and outputs:

$$R_{k,t} = A_{\text{LLM}}\big(\tau_{k,t}, o_{\text{glob},k,t}, y_k; P_{\text{attr}}\big), \qquad R_{k,t} \subseteq V,$$

where P_{attr} denotes the prompt that guides error attribution using the gold answer, and $R_{k,t}$ is the set of identified agents. Multiple agents may be attributed to an incorrect trajectory.

For agent i, the attributed sample pool $\mathcal{D}_{i,t}^-$ and solved sample pool $\mathcal{D}_{i,t}^+$ are defined as:

$$\mathcal{D}_{i,t}^- = \big\{(q_k, \tau_{k,t}, y_k) \;\big|\; \tau_{k,t} \in \mathcal{T}_t^-,\ i \in R_{k,t}\big\}, \quad \mathcal{D}_{i,t}^+ = \big\{(q_k, \tau_{k,t}, y_k) \;\big|\; \tau_{k,t} \in \mathcal{T}_t^+,\ i \in V\big\}.$$

The agent-specific optimization set is then $\mathcal{D}_{i,t} = \mathcal{D}_{i,t}^- \cup \mathcal{D}_{i,t}^+$.

4.2 Prompt Optimization

In this stage, each agent's prompt is sequentially refined using its agent-specific optimization set. This process is repeated over multiple rounds, yielding the final optimized configuration P_T.

Candidate Prompt Generation. For a target agent v_i, the optimizer LLM generates a set of candidate prompts based on its current prompt and the optimization set: $\mathcal{C}_{i,t} = \{p_{i,t}'^{(1)}, \ldots, p_{i,t}'^{(K)}\} \sim \mathcal{O}(p_{i,t}, \mathcal{D}_{i,t})$. For a candidate prompt $p'_{i,t} \in \mathcal{C}_{i,t}$, replacing only $p_{i,t}$ in the current configuration $P_t = \{p_{1,t}, \ldots, p_{i,t}, \ldots, p_{N,t}\}$ yields the candidate updated configuration $P_t^{(i \leftarrow p'_{i,t})} = \{p_{1,t}, \ldots, p'_{i,t}, \ldots, p_{N,t}\}$.

For each question q_k in the training set, we obtain the correctness of the output before and after the replacement: $r_{k,t} = R_{\text{task}}(\Phi(q_k; G, P_t), y_k), r'_{k,t} = R_{\text{task}}(\Phi(q_k; G, P_t^{(i \leftarrow p'_{i,t})}), y_k)$.The transition score is then defined as:

$$s_{\text{trans}}(q_k) = \begin{cases} 1, & r_{k,t} = 0,\ r'_{k,t} = 1, \\ -\eta, & r_{k,t} = 1,\ r'_{k,t} = 0, \\ 0, & \text{otherwise}, \end{cases}$$

where hyper-parameter $\eta > 0$ penalizes the degradation from a correct answer to an incorrect one.

The final-answer score over a batch B is defined as:

$$S_{\text{ans}}(p'_{i,t} \mid p_{i,t}) = \frac{1}{|B|} \sum_{q_k \in B} s_{\text{trans}}(q_k).$$

Prompt Update. Since updating a single agent's prompt may impair downstream collaboration in the multi-agent system, we do not optimize final-answer gains in isolation. Following MASPO's joint-evaluation design (Wang et al., 2026) to model inter-agent dependencies, we retain two auxiliary scores: S_{succ}, which measures the usefulness of the updated output to successor agents, and S_{prompt}, which evaluates the preservation of the agent's role, output constraints, and task requirements. The overall score is therefore defined as:

$$S_{\text{overall}}(p'_{i,t} \mid p_{i,t}) = S_{\text{ans}}(p'_{i,t} \mid p_{i,t}) + \lambda_{\text{succ}} S_{\text{succ}}(p'_{i,t} \mid p_{i,t}) + \lambda_{\text{prompt}} S_{\text{prompt}}(p'_{i,t} \mid p_{i,t}).$$

where the hyperparameters λ_{succ} and λ_{prompt} control the contributions of successor compatibility and prompt-quality regularization, respectively.

For a target agent, the optimizer generates a candidate set $\mathcal{C}_{i,t}$. The current prompt $p_{i,t}$ is also included as a fallback candidate with zero improvement. We first select the best candidate and then update the prompt only if the selected candidate yields a positive improvement:

$$\hat{p}_{i,t} = \arg\max_{p' \in \mathcal{C}_{i,t}} S_{\text{overall}}(p' \mid p_{i,t}), \quad p_{i,t+1} = \begin{cases} \hat{p}_{i,t}, & S_{\text{overall}}(\hat{p}_{i,t} \mid p_{i,t}) > 0, \\ p_{i,t}, & \text{otherwise}. \end{cases}$$

The prompts for each agent are updated sequentially in a predefined order. This updating process can be repeated over multiple rounds, yielding the final configuration P_T. During inference, P_T is deployed to the multi-agent system to perform inference on the test samples.

5 Theoretical Analysis

We use candidate-gain estimation to explain why answer guidance and attribution filtering can improve prompt optimization under a small sample budget. Let $P_t^{(i \leftarrow p'_{i,t})}$ be the configuration obtained by replacing only the prompt of agent v_i. The expected task gain of the candidate prompt is:

$$\Delta(p'_{i,t}) = \mathbb{E}_{(q,y)\sim\mathcal{D}} \left[R_{\text{task}}\big(\Phi(q; G, P_t^{(i \leftarrow p'_{i,t})}), y\big) - R_{\text{task}}\big(\Phi(q; G, P_t), y\big) \right].$$

Since the data distribution of $\mathcal{D}$ is inaccessible, optimization relies on a finite-batch estimate $\hat{\Delta}(p'_{i,t})$. The estimation error can be decomposed as:

$$\mathbb{E}\left[(\hat{\Delta}(p'_{i,t}) - \Delta(p'_{i,t}))^2 \right] = \Big(\mathbb{E}[\hat{\Delta}(p'_{i,t})] - \Delta(p'_{i,t}) \Big)^2 + \text{Var}[\hat{\Delta}(p'_{i,t})],$$

where the first term measures the feedback bias and the second term captures the finite-sample variability. A detailed derivation is provided in the Appendix C.1.

This decomposition reveals two failure modes in recent methods. First, when candidate prompts are evaluated solely by generic LLM preferences (without access to gold labels), the feedback can favor longer, more plausible, or stylistically better outputs even when final task correctness does not improve. This introduces a bias, as the proxy objective is not fully aligned with the gold-label evaluation R_{task}. Gold-answer guidance addresses this by estimating the gain in task correctness:

$$\hat{\Delta}_{\text{gold}}(p'_{i,t}) = \frac{1}{|B|} \sum_{q_k \in B} \left[R_{\text{task}}\big(\Phi(q_k; G, P_t^{(i \leftarrow p'_{i,t})}), y_k\big) - R_{\text{task}}\big(\Phi(q_k; G, P_t), y_k\big) \right],$$

where the estimate $\hat{\Delta}_{\text{gold}}(p'_{i,t})$ should align more closely with the expected task gain $\Delta(p'_{i,t})$ than label-free LLM feedback does, thereby yielding lower bias. A formal analysis is in Appendix C.2.

Second, randomly sampled failure trajectories constitute a mixture of failures attributable to different agents. When optimizing a specific agent, trajectories caused by other agents introduce irrelevant supervision into prompt generation. Such supervision does not consistently reflect the target agent's failure modes, thereby increasing the dispersion of the induced prompt updates. Since these updates may exhibit inconsistent effects across evaluation samples, their estimated gains tend to have high variance. In contrast, responsibility-aware sampling conditions prompt optimization on failures attributed to the target agent, thereby increasing the relevance and directional consistency of the optimization signal. This filtering effect suppresses agent-irrelevant variation and leads to more stable gains across evaluation samples. A formal analysis is provided in Appendix C.3.

6 Experiments

6.1 Experimental Setup

Models and Benchmarks. Following MASPO (Wang et al., 2026), we use Qwen3-8B (Team, 2025b) as the MAS backbone and Gemini-2.5-Pro (Team, 2025a) as the prompt optimizer. We evaluate accuracy on three categories of benchmarks: math (MATH-500 (MATH) (Hendrycks et al., 2021), AQuA (Ling et al., 2017), AGIEval-MATH Level-5 (AGIEval) (Zhong et al., 2024)), complex reasoning (GPQA-Diamond (GPQA) (Rein et al., 2023)), and code generation (MBPP (Austin et al., 2021), HumanEval-ET (HumanEval) (Chen et al., 2021)).

Baselines. We primarily compare with the recent state-of-the-art method MASPO, as we follow the same experimental setting that adopts a fixed-topology role-prompt optimization. While the original MASPO paper reports results from a single run, we rerun both MASPO and our method with three random seeds and report their average performance to ensure a fair and rigorous comparison. Other results, including Vanilla, CoT (Wei et al., 2022), SC-CoT (Wang et al., 2023), Self-Refine (Madaan et al., 2023), and AgentDropout (Wang et al., 2025) are retrieved from MASPO's paper.

Evaluation Protocol. We conduct evaluation under both Sequential and Hierarchical MAS architectures. The Sequential MAS uses four agents in a Predictor-Reflector-Predictor-Reflector pipeline. The Hierarchical MAS includes code, science, and math sub-agents, followed by an aggregation agent. For MBPP, we adhere to the original benchmark protocol (Austin et al., 2021) and include test examples in the input, rather than adopting the MASPO setting that omits them. Omitting the test examples would cause the optimization to focus on guessing function names.

Implementation Details. As in MASPO, we randomly sample 50 instances to form the training set. For optimizing an agent's prompt, we use 4 samples to guide candidate prompt generation. Specifically, two samples are drawn from the agent's attributed sample pool—if the pool contains more than two, two are chosen at random; otherwise, all available ones are used. The remaining slots are filled by random draws from the solved sample pool. When generating a candidate prompt, we perform nine independent attempts to construct the candidate set. These settings are shared across both MAS architectures. Further details are provided in Appendix D.

6.2 Main Results

The performance comparison is presented in Table 1. Even without optimization, both the Sequential and Hierarchical MAS methods generally outperform single-agent methods on average, highlighting the value of multi-agent collaboration. MASPO further improves the average performance over the basic MAS methods under both architectures; however, its gains are not consistent across tasks. In particular, it degrades GPQA performance in both settings. In fact, all methods perform poorly on this dataset, with accuracy below 50, underscoring the difficulty of the science domain. This result indicates that the backbone LLM lacks sufficient domain-specific knowledge, and its limited capability prevents self-generated preferences from providing reliable optimization directions.

Our method achieves the best average performance under both MAS architectures. It outperforms MASPO on five of six benchmarks with Sequential MAS (all except MATH) and on five benchmarks with Hierarchical MAS, while matching MASPO on AQuA. These improvements are obtained using

Table 1: Performance comparison across six benchmarks. Best results are shown in bold.

Method	Prompt Opt.	Joint Opt.	Math			Science	Code		Avg.
			MATH	AGIEval	AQuA	GPQA	MBPP	HumanEval	
Vanilla	×	×	74.80	55.86	79.53	45.96	63.47	71.95	65.26
CoT	×	×	75.40	54.69	81.89	46.72	64.17	72.26	65.85
SC-CoT	×	×	75.50	56.64	82.17	47.52	64.49	72.56	66.48
Self-Refine	×	×	76.20	56.62	82.28	47.73	64.17	70.73	66.28
AgentDropout	×	×	76.80	59.77	86.23	47.98	58.09	72.44	66.88
Sequential MAS	×	×	82.66	66.01	83.46	49.32	77.36	67.88	71.11
+MASPO	✓	✓	**83.06**	69.40	85.56	45.62	81.73	69.71	72.51
+Ours	✓	✓	82.66	**72.52**	**87.92**	**49.83**	**82.35**	**72.96**	**74.70**
Hierarchical MAS	×	×	83.93	66.53	87.53	48.48	78.14	74.39	73.16
+MASPO	✓	✓	84.20	70.96	**87.66**	48.31	82.59	74.79	74.75
+Ours	✓	✓	**85.66**	**71.87**	**87.66**	**49.49**	**85.40**	**75.81**	**75.98**

Table 2: Ablation study. **Gold**, **Attr.**, and k denote gold-answer guidance, error attribution, and the number of samples used to guide prompt generation, respectively.

Method	Gold	Attr.	k	Math			Science	Code		Avg.
				MATH	AGIEval	AQuA	GPQA	MBPP	HumanEval	
Sequential MAS	×	×	/	82.66	66.01	83.46	49.32	77.36	67.88	71.11
+MASPO	×	×	10	83.06	69.40	85.56	45.62	81.73	69.71	72.51
+LABEL	✓	×	10	**84.53**	67.96	87.79	**52.02**	**83.68**	67.27	73.87
+LABEL	✓	×	4	81.46	68.61	86.87	45.45	82.50	67.47	72.06
+Ours	✓	✓	4	82.66	**72.52**	**87.92**	49.83	82.35	**72.96**	**74.70**
Hierarchical MAS	×	×	/	83.93	66.53	87.53	48.48	78.14	74.39	73.16
+MASPO	×	×	10	84.20	70.96	**87.66**	48.31	82.59	74.79	74.75
+LABEL	✓	×	10	85.40	67.96	87.27	**50.16**	77.82	73.78	73.73
+LABEL	✓	×	4	85.26	65.75	85.82	49.15	83.84	72.56	73.73
+Ours	✓	✓	4	**85.66**	**71.87**	**87.66**	49.49	**85.40**	**75.81**	**75.98**

only four samples per optimization step, compared to ten for MASPO, demonstrating higher sample efficiency. Notably, our method surpasses all prior approaches on GPQA, highlighting the effect of gold-label supervision for prompt optimization in this challenging domain.

For the weaker performance of our method on MATH under the Sequential MAS architecture, we find that, due to the limited capability of the attribution LLM, the attributed samples remain highly similar across optimization rounds, causing the optimization to focus on a narrow subset of failures. This feedback may fail to cover the full diversity of reasoning errors, thereby limiting generalization. Detailed statistics on the overlap of attributed samples are provided in Appendix B.

6.3 Ablation Study

To isolate the effects of gold-answer guidance and error attribution, we compare our method with three baselines—MAS, MASPO, and a variant called LABEL—under both Sequential and Hierarchical MAS architectures on six benchmarks, as reported in Table 2. Specifically, MAS is the basic multi-agent method; MASPO extends it with prompt optimization; LABEL further incorporates gold-label guidance; and our method adds error attribution on top of LABEL to select samples. Here, k denotes the number of samples used to guide prompt generation.

We first compare MASPO and LABEL with $k = 10$ under the Sequential MAS architecture. LABEL, which adds gold-answer guidance, yields mixed results across benchmarks: it outperforms MASPO on MATH, AQuA, GPQA, and MBPP, but underperforms on AGIEval and HumanEval. Interestingly, on the datasets where LABEL surpasses MASPO, our method ($k = 4$) tends to underperform LABEL; conversely, on those where LABEL trails MASPO, our method tends to outperform LABEL. Similar trends are observed under the Hierarchical MAS architecture.

A possible reason is that, on benchmarks where LABEL already outperforms MASPO, randomly selected samples appear useful—gold-label guidance alone already points the optimizer in the right

Table 3: Stability and efficiency comparison of different methods.

Metric	AGIEval			GPQA			HumanEval			Avg.		
	MASPO	LABEL	Ours	MASPO	LABEL	Ours	MASPO	LABEL	Ours	MASPO	LABEL	Ours
Best Acc.	70.00	82.00	82.00	62.00	54.00	58.00	80.00	80.00	74.00	70.66	**72.00**	71.33
Final Acc.	70.00	76.00	78.00	42.00	46.00	58.00	76.00	78.00	74.00	62.66	66.66	**70.00**
Range	12.00	14.00	8.00	20.00	12.00	14.00	12.00	12.00	10.00	14.66	12.66	**10.66**
Std.	3.27	3.49	2.82	5.45	3.52	4.08	3.54	3.27	2.88	4.08	3.42	**3.26**
Max Drop	-8.00	-10.00	-6.00	-14.00	-8.00	-10.00	-4.00	-6.00	-6.00	-8.66	-8.00	**-7.33**
Time (min)	489.13	495.43	514.62	321.63	520.07	336.22	109.56	128.01	186.54	**306.77**	381.17	345.79

Table 4: Performance comparison of MASPO and our method under different LLM backbones.

Dataset	Optimizer				Generator			
	Gemini-2.5-Pro		DeepSeek-V4.1-Flash		Gemini-2.5-Flash		DeepSeek-V4.1-Flash	
	MASPO	Ours	MASPO	Ours	MASPO	Ours	MASPO	Ours
AGIEval	70.96	**71.87**	63.28	**67.57**	**91.92**	91.27	**95.18**	**95.18**
GPQA	48.31	**49.49**	46.12	**49.15**	78.45	**79.46**	77.77	**81.48**
HumanEval	74.79	**75.81**	68.69	**71.54**	84.56	**86.78**	83.94	**84.75**
Avg.	64.68	**65.72**	59.36	**62.75**	84.97	**85.83**	85.63	**87.13**

direction. In such cases, error attribution filtering can inadvertently reduce sample diversity and discard complementary signals, thus hurting performance. Conversely, on benchmarks where LABEL degrades, the samples are likely dominated by irrelevant failures that mislead optimization. Attribution filtering removes these noisy samples and retains more informative ones, allowing our method to improve. These observations highlight a relevance–coverage trade-off: attribution filtering improves sample relevance, but may sacrifice the diversity of the optimization set.

Nonetheless, Ours ($k = 4$) outperforms both MASPO ($k = 10$) and LABEL ($k = 10$) in average accuracy, highlighting its efficiency. When LABEL is also evaluated with $k = 4$, our method remains superior across all datasets. Overall, these results indicate that our method generally yields more effective prompt-update signals under a fixed sample budget.

6.4 Further Analysis

Due to space limitations, all analyses use Hierarchical MAS on GPQA, AGIEval, and HumanEval.

Optimization Stability Analysis. Table 3 reports, for each method, the best and final accuracy (Best Acc. and Final Acc.), the accuracy range (Range), the standard deviation (Std), and the maximum drop (Max Drop) observed on the validation set during optimization. Here, Best Acc. denotes the peak validation accuracy; Final Acc. is the accuracy after the last optimization round; Range is the difference between the best and worst accuracy; Std is the round-wise standard deviation; and Max Drop captures the largest accuracy decrease between two consecutive rounds.

First, the margin between Best Acc. and Final Acc. is the smallest for our method on average. Combined with its lower Max Drop, this suggests that our method better preserves performance once the best validation result is reached. Second, for Range and Std, our method slightly lags behind LABEL on GPQA, but outperforms both MASPO and LABEL on the other two datasets, achieving the best average scores. Overall, these observations indicate that our method exhibits superior stability during optimization. Finally, with respect to Time (i.e., the optimization cost over 12 rounds in minutes), our method increases the cost over MASPO by 1.1 times on average. Specifically, on GPQA, LABEL tends to produce more detailed reasoning; combined with GPQA's long, knowledge-intensive questions, this leads to a substantially higher time cost.

Performance on Different LLM Backbones. To evaluate the generalization of our method across LLM backbones, we vary either the optimizer or the generator while keeping the remaining components and optimization settings fixed. Here, the Optimizer refers to the model that performs error

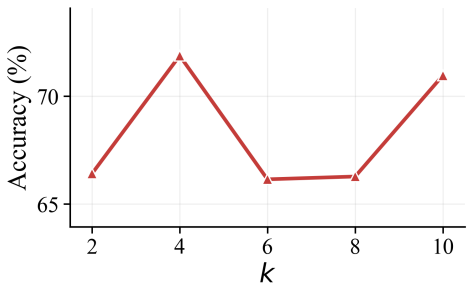


Figure 3: Impact of k on accuracy for the AGIEval dataset.

Table 5: Average number of samples attributed to the agent being optimized across four agents (A1–A4 represent Agent 1 through Agent 4).

Dataset	Sequential				Hierarchical			
	A1	**A2**	**A3**	**A4**	**A1**	**A2**	**A3**	**A4**
AGIEval	5.08	5.08	5.83	6.75	4.83	4.66	4.41	3.75
AQuA	3.00	1.41	1.58	2.58	3.00	2.50	3.25	1.33
GPQA	7.50	7.83	6.75	6.83	7.58	6.41	7.00	0.66
HumanEval	2.25	1.50	2.66	6.25	6.00	4.66	4.83	3.25
MATH	2.33	3.41	3.58	6.16	3.00	2.00	3.16	2.50
MBPP	2.83	2.33	1.50	3.75	5.16	6.83	6.25	4.91
Avg.	3.83	3.59	3.65	5.38	4.92	4.51	4.81	2.73

Table 6: Performance comparison of different sample composition strategies.

Sample Composition	AGIEval	GPQA	HumanEval	Avg.
Correct Only	66.27	50.00	73.17	63.14
Random Errors Only	67.57	**50.67**	74.79	64.34
Attributed Errors Only	69.01	49.49	**76.42**	64.97
Attributed Errors + Correct Samples (Ours)	**71.87**	49.49	75.81	**65.72**

attribution and prompt optimization in our method, while the Generator refers to the inference LLM used by each agent. The results are presented in Table 4. Our method consistently outperforms MASPO on average across all four configurations. When the optimizer is varied, it improves all three benchmarks for both Gemini-2.5-Pro and DeepSeek-V4.1-Flash, with a larger average gain under the latter. When the generator is varied, our method improves GPQA and HumanEval in both settings. On AGIEval, it slightly trails MASPO with Gemini-2.5-Flash and remains unchanged with DeepSeek-V4.1-Flash. These results prove the generality of our method across different LLMs.

Impact of k on Performance. Performance under different values of k (i.e., the number of samples used to guide prompt generation) on AGIEval is presented in Figure 3, with $k = 2, 4, 6, 8, 10$. Our method achieves the best performance at $k = 4$. As k increases, the performance fluctuates. Table 5 provides insight into why $k = 4$ works best. On average, only about 4.2 samples are attributed to the agent being optimized across the six benchmarks and four agents. We set $k = 4$ for our method, so that each update focuses primarily on agent-relevant errors while limiting the influence of unrelated samples. This choice offers a balance between update quality and efficiency.

Sample Composition Analysis. In addition to attributed samples (failed trajectories), our method randomly includes a few solved samples (positive references) to form the agent-specific optimization set. To validate this composition strategy, Table 6 compares three variants under the Hierarchical MAS: (1) "Correct Only" uses only solved samples; (2) "Random Errors Only" uses only randomly selected failed samples; (3) "Attributed Errors Only" uses only attributed failed samples. "Random Errors Only" outperforms "Correct Only", indicating that failed samples are more important in prompt optimization. "Attributed Errors Only" surpasses "Random Errors Only" in average accuracy, suggesting that error attribution helps select more informative samples. Our method achieves the best in average, demonstrating the benefit of incorporating solved samples as regularization.

7 Conclusion

We propose a responsibility attribution-driven prompt optimization framework for MASs. Its two key designs—replacing unreliable LLM self-evaluation with gold-answer guidance and constructing agent-specific sample sets via error attribution—jointly enable stable prompt updates with smaller sample budgets. A formal bias–variance analysis theoretically justifies their complementary roles. Across code generation, math reasoning and scientific QA, our method outperforms MASPO, with ablation and cross-backbone studies confirming the contribution and generality of each design. These results demonstrate that reliable supervision and agent-specific feedback form a principled path toward sample-efficient and robust multi-agent prompt optimization.

AI USE STATEMENT

In this work, we used generative AI tools for the generation or revision of analysis scripts and plotting scripts. We have not used generative AI tools for research idea generation, experimental design decisions, or the formulation of scientific claims, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for language polishing, draft translation, assistance with theoretical derivations and proof writing, LaTeX and manuscript formatting, table formatting, and consistency checking between result files and the tables reported in the manuscript. We reviewed all AI-assisted work: all AI-assisted text was checked and revised by the authors; generated or revised code was verified and tested for correctness; AI-assisted theoretical derivations and proofs were independently checked by the authors; and all reported results, interpretations, claims, and final manuscript content were independently reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, derivations, proofs, code, or other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies prompt optimization for LLM-based multi-agent systems on public academic benchmarks, including mathematical reasoning, scientific question answering, and code generation datasets. The experiments do not involve human subjects, private user data, personally identifiable information, or newly collected sensitive data. The proposed method is intended to improve the stability and sample efficiency of prompt optimization under a fixed multi-agent topology. All datasets are used according to their intended research purposes, and we do not release new datasets containing sensitive or restricted information.

REPRODUCIBILITY STATEMENT

We make the experimental setting explicit throughout the paper and appendix to support reproducibility. Section 6.1 describes the evaluated datasets, model settings, evaluation protocols, and compared methods. Appendix D provides implementation details, including optimization budgets, sample construction, scoring weights, and architecture settings. Appendix F reports the actual prompt templates and the fields provided to the optimizer, attribution module, reward scorer, and LLM judge. Appendix C gives the complete theoretical derivation. The experiments use fixed random seeds where applicable.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: reflective prompt evolution can outperform reinforcement learning. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust (eds.), *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0e9e708b6f48e14fd0ac29e167413f76-Abstract-Conference.html.

Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Juyang Bai and Laixi Shi. Mas-promptbench: When does prompt optimization improve multi-agent LLM systems? *CoRR*, abs/2606.23664, 2026. doi: 10.48550/ARXIV.2606.23664. URL https://doi.org/10.48550/arXiv.2606.23664.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan,

Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Evoprompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers, 2025. URL https://arxiv.org/abs/2309.08532.

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 Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Zhi Hong, Qian Zhang, Jiahang Sun, Zhiwei Shang, Mingze Kong, Xiangyi Wang, Yao Shu, and Zhongxiang Dai. MASPOB: bandit-based prompt optimization for multi-agent systems with graph neural networks. *CoRR*, abs/2603.02630, 2026. doi: 10.48550/ARXIV.2603.02630. URL https://doi.org/10.48550/arXiv.2603.02630.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: communicative agents for "mind" exploration of large language model society. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html.

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 Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 17889–17904. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.992. URL https://doi.org/10.18653/v1/2024.emnlp-main.992.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 158–167. Association for Computational Linguistics, 2017. doi: 10.18653/V1/P17-1015. URL https://doi.org/10.18653/v1/P17-1015.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and

Peter Clark. Self-refine: Iterative refinement with self-feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Kehao Mao, Baokun Hu, Ruixin Lin, Zewen Li, Guanyu Lu, and Zhengyu Zhang. Blueprint2code: a multi-agent pipeline for reliable code generation via blueprint planning and repair. *Frontiers Artif. Intell.*, 8, 2025. doi: 10.3389/FRAI.2025.1660912. URL https://doi.org/10.3389/frai.2025.1660912.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022, 2023. doi: 10.48550/ARXIV.2311.12022. URL https://doi.org/10.48550/arXiv.2311.12022.

Wonduk Seo, Juhyeon Lee, Junseo Koh, Wonseok Choi, Hyunjin An, Jian Park, Seunghyun lee, Haihua Chen, and Yi Bu. Ma-sapo: Multi-agent reasoning for score-aware prompt optimization, 2026. URL https://arxiv.org/abs/2510.16635.

Ming Shen, Raphael Shu, Anurag Pratik, James Gung, Yubin Ge, Monica Sunkara, and Yi Zhang. Optimizing llm-based multi-agent system with textual feedback: A case study on software development. *CoRR*, abs/2505.16086, 2025. doi: 10.48550/ARXIV.2505.16086. URL https://doi.org/10.48550/arXiv.2505.16086.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Andries P. 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 Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 45883–45905. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/smit24a.html.

Kai Sun, Wenqiang Li, Bo Dong, Yuxin Lin, Jingyao Zhang, and Bin Shi. Scope delineation before localization: A two-stage framework for enhancing failure attribution in multi-agent systems. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 33108–33116. AAAI Press, 2026. doi: 10.1609/AAAI.V40I39.40594. URL https://doi.org/10.1609/aaai.v40i39.40594.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261, 2025a. doi: 10.48550/ARXIV.2507.06261. URL https://doi.org/10.48550/arXiv.2507.06261.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025b. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

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 *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

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 Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 24013–24035. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.1170. URL https://doi.org/10.18653/v1/2025.acl-long.1170.

Zhexuan Wang, Xuebo Liu, Li Wang, Zifei Shan, Yutong Wang, Zhenxi Song, and Min Zhang. MASPO: joint prompt optimization for llm-based multi-agent systems. *CoRR*, abs/2605.06623, 2026. doi: 10.48550/ARXIV.2605.06623. URL https://doi.org/10.48550/arXiv.2605.06623.

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 Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Yihan Xia, Taotao Wang, Shengli Zhang, Zhangyuhua Weng, Bin Cao, and Soung Chang Liew. Hivemind: Contribution-guided online prompt optimization of LLM multi-agent systems. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 29767–29774. AAAI Press, 2026. doi: 10.1609/AAAI.V40I35.40222. URL https://doi.org/10.1609/aaai.v40i35.40222.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

Mert Yüksekgönül, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text. *CoRR*, abs/2406.07496, 2024. doi: 10.48550/ARXIV.2406.07496. URL https://doi.org/10.48550/arXiv.2406.07496.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Zheyuan Zhang, Lin Ge, Hongjiang Li, Weicheng Zhu, Chuxu Zhang, and Yanfang Ye. MAPRO: recasting multi-agent prompt optimization as maximum a posteriori inference. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Findings of the Association for Computational Linguistics: EACL 2026, Rabat, Morocco, March 24-29, 2026*, Findings of ACL, pp. 4458–4480. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.FINDINGS-EACL.233. URL https://doi.org/10.18653/v1/2026.findings-eacl.233.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Findings of*

the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024, volume NAACL 2024 of *Findings of ACL*, pp. 2299–2314. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.149. URL https://doi.org/10.18653/v1/2024.findings-naacl.149.

Taiyu Zhu, Yifan Wu, Weilin Jin, Ying Li, and Gang Huang. Stepfinder: A temporal semantic framework for failure attribution in multi-agent systems. In Won Kim, Jae-Gil Lee, Kyuseok Shim, James Caverlee, David F. Gleich, Xiangliang Zhang, Qi He, Faisal Farooq, Rómer Rosales, Sayan Ranu, Haixun Wang, Yan Liu, Carl Yang, Leman Akoglu, Nitesh V. Chawla, and Vagelis Papalexakis (eds.), *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2, KDD 2026, Jeju Island, Korea, August 9-13, 2026*, pp. 6946–6957. ACM, 2026. doi: 10.1145/3770855.3817991. URL https://doi.org/10.1145/3770855.3817991.

A Additional Experimental Results

A.1 Full Optimization Stability Analysis

To examine whether the optimization-stability trends observed in the main paper generalize across datasets and MAS architectures, we extend the analysis to all six benchmarks under both the Sequential and Hierarchical MAS. We compare MASPO, LABEL, and our method using the complete processes recorded during prompt optimization. Tables 7 and 8 report the corresponding results. In addition to Best Acc., Final Acc., Range, Std., and Max Drop defined in Section 6.4, we further report Gap and Avg. Step Change. Gap is the difference between Best Acc. and Final Acc., measuring how much of the peak performance is lost by the end of optimization, while Avg. Step Change is the mean absolute accuracy change between adjacent rounds.

Under the Sequential MAS, our method achieves the highest average Final Acc. and the smallest average Gap, reducing the best-to-final gap from 7.66 for MASPO and 6.00 for LABEL to 4.00. It also obtains the smallest Avg. Step Change, indicating that successive prompt updates lead to more moderate performance fluctuations. Although LABEL achieves slightly better average Range, Std., and Max Drop, our method preserves a larger fraction of the best discovered performance while maintaining competitive stability.

The stability advantage is more pronounced under the Hierarchical MAS. Our method achieves the highest average Final Acc. and reduces the average Gap to 1.00, compared with 5.66 for MASPO and 5.00 for LABEL. It also obtains the smallest average Range, Avg. Step Change, and Max Drop, while its Std. remains close to LABEL. These results show that attribution-guided optimization is particularly effective at preserving improvements and reducing optimization fluctuations in the Hierarchical architecture.

A.2 Supplementary Efficiency Analysis

To further evaluate the efficiency of prompt optimization, we compare MASPO, LABEL, and our method across all six datasets under both the Sequential and Hierarchical MAS architectures. Table 9 reports the optimization cost (Time), final accuracy (Final Acc.), and round-wise standard deviation (Std.) for each method. Our method uses 4 samples per prompt update, whereas MASPO and LABEL use 10 samples; all methods use the same training set. Here, Time denotes the optimization cost over 12 rounds in minutes. Together, these metrics characterize the trade-off between optimization cost, final performance, and stability.

First, despite using fewer samples per update, our method achieves the highest average Final Acc. under both architectures, reaching 77.33 under the Sequential MAS and 80.33 under the Hierarchical MAS, respectively. Second, this improvement comes with only a moderate increase in optimization time over MASPO. Compared with LABEL, our method achieves higher final accuracy while requiring less average optimization time under both architectures. Finally, in terms of Std., our method is more stable than MASPO on average, while remaining close to LABEL. These results indicate that attribution-guided sample selection provides a favorable accuracy–efficiency trade-off, achieving stronger final performance with a smaller optimization sample budget while maintaining competitive stability.

Table 7: Full stability analysis under the Sequential MAS.

Dataset	Method	Best Acc.	Final Acc.	Gap	Range	Std.	Avg. Step Change	Max Drop
MATH	MASPO	84.00	80.00	4.00	10.00	3.19	3.66	-10.00
	LABEL	90.00	90.00	0.00	8.00	2.82	3.83	-6.00
	Ours	92.00	88.00	4.00	8.00	2.40	2.83	-8.00
AGIEval	MASPO	70.00	64.00	6.00	8.00	2.33	2.83	-6.00
	LABEL	84.00	76.00	8.00	14.00	3.67	4.66	-8.00
	Ours	82.00	78.00	4.00	18.00	4.11	4.66	-12.00
AQuA	MASPO	92.00	88.00	4.00	12.00	3.37	3.33	-8.00
	LABEL	92.00	86.00	6.00	10.00	2.69	3.00	-6.00
	Ours	92.00	90.00	2.00	8.00	2.43	2.66	-4.00
GPQA	MASPO	62.00	52.00	10.00	26.00	6.44	5.83	-20.00
	LABEL	54.00	52.00	2.00	10.00	3.56	5.33	-8.00
	Ours	56.00	48.00	8.00	20.00	5.01	5.66	-12.00
MBPP	MASPO	86.00	82.00	4.00	6.00	2.07	3.33	-6.00
	LABEL	90.00	84.00	6.00	10.00	3.47	2.66	-4.00
	Ours	90.00	84.00	6.00	10.00	2.37	3.33	-8.00
HumanEval	MASPO	82.00	64.00	18.00	18.00	6.14	8.83	-18.00
	LABEL	76.00	62.00	14.00	14.00	4.06	5.00	-10.00
	Ours	76.00	76.00	0.00	16.00	4.46	4.00	-10.00
Avg.	MASPO	79.33	71.66	7.66	13.33	3.92	4.63	-11.33
	LABEL	81.00	75.00	6.00	11.00	3.37	4.08	-7.00
	Ours	81.33	77.33	4.00	13.33	3.46	3.85	-9.00

Table 8: Full stability analysis under the Hierarchical MAS.

Dataset	Method	Best Acc.	Final Acc.	Gap	Range	Std.	Avg. Step Change	Max Drop
MATH	MASPO	86.00	82.00	4.00	8.00	2.95	3.66	-6.00
	LABEL	94.00	92.00	2.00	12.00	3.28	3.83	-10.00
	Ours	92.00	92.00	0.00	10.00	2.78	2.66	-4.00
AGIEval	MASPO	70.00	70.00	0.00	12.00	3.27	3.33	-8.00
	LABEL	82.00	76.00	6.00	14.00	3.49	3.83	-10.00
	Ours	82.00	78.00	4.00	8.00	2.82	2.83	-6.00
AQuA	MASPO	96.00	94.00	2.00	10.00	3.03	2.00	-4.00
	LABEL	94.00	90.00	4.00	8.00	2.24	3.00	-6.00
	Ours	94.00	92.00	2.00	8.00	2.33	1.66	-6.00
GPQA	MASPO	62.00	42.00	20.00	20.00	5.45	6.33	-14.00
	LABEL	54.00	46.00	8.00	12.00	3.52	4.50	-8.00
	Ours	58.00	58.00	0.00	14.00	4.08	4.66	-10.00
MBPP	MASPO	88.00	84.00	4.00	12.00	2.84	2.16	-4.00
	LABEL	90.00	82.00	8.00	10.00	2.67	3.00	-10.00
	Ours	88.00	88.00	0.00	14.00	4.19	2.50	-4.00
HumanEval	MASPO	80.00	76.00	4.00	12.00	3.54	2.66	-4.00
	LABEL	80.00	78.00	2.00	12.00	3.27	2.83	-6.00
	Ours	74.00	74.00	0.00	10.00	2.88	2.16	-6.00
Avg.	MASPO	80.33	74.66	5.66	12.33	3.51	3.35	-6.66
	LABEL	82.33	77.33	5.00	11.33	3.07	3.49	-8.33
	Ours	81.33	80.33	1.00	10.66	3.18	2.74	-6.00

B TRACE-BASED FAILURE MODE ANALYSIS

To examine how responsibility attribution affects the construction of agent-specific optimization batches, we analyze the samples selected during prompt optimization across all six datasets under both the Sequential and Hierarchical MAS architectures. Table 10 reports attribution coverage, batch

Table 9: Efficiency and stability statistics under Sequential and Hierarchical MAS.

Sequential MAS					Hierarchical MAS				
Dataset	**Method**	**Time (min)**	**Final Acc.**	**Std.**	**Dataset**	**Method**	**Time (min)**	**Final Acc.**	**Std.**
MATH	MASPO	599.85	80.00	3.19	MATH	MASPO	357.82	82.00	2.95
	LABEL	748.54	90.00	2.82		LABEL	408.89	92.00	3.28
	Ours	692.99	88.00	2.40		Ours	460.65	92.00	2.78
AGIEval	MASPO	777.79	64.00	2.33	AGIEval	MASPO	489.13	70.00	3.27
	LABEL	662.42	76.00	3.67		LABEL	495.43	76.00	3.49
	Ours	722.96	78.00	4.11		Ours	514.62	78.00	2.82
AQuA	MASPO	441.07	88.00	3.37	AQuA	MASPO	231.34	94.00	3.03
	LABEL	633.34	86.00	2.69		LABEL	242.20	90.00	2.24
	Ours	515.85	90.00	2.43		Ours	255.85	92.00	2.33
GPQA	MASPO	647.72	52.00	6.44	GPQA	MASPO	321.63	42.00	5.45
	LABEL	1037.68	52.00	3.56		LABEL	520.07	46.00	3.52
	Ours	773.14	48.00	5.01		Ours	336.22	58.00	4.08
MBPP	MASPO	336.92	82.00	2.07	MBPP	MASPO	132.51	84.00	2.84
	LABEL	370.69	84.00	3.47		LABEL	151.95	82.00	2.67
	Ours	416.99	84.00	2.37		Ours	136.49	88.00	4.19
HumanEval	MASPO	280.68	64.00	6.14	HumanEval	MASPO	109.56	76.00	3.54
	LABEL	379.25	62.00	4.06		LABEL	128.01	78.00	3.27
	Ours	358.64	76.00	4.46		Ours	186.54	74.00	2.88
Avg.	MASPO	514.00	71.66	3.92	Avg.	MASPO	273.67	74.66	3.51
	LABEL	638.65	75.00	3.37		LABEL	324.43	77.33	3.07
	Ours	580.09	77.33	3.46		Ours	315.06	80.33	3.18

composition, sample reuse, and error shortages. **Related/Diagnosed** denotes the average proportion of diagnosed failures that are attributed to the target agent in each round. **Selected Error** and **Correct Fill** denote the average proportions of attributed error samples and correct samples in each optimization round, respectively. **Unique Selected** denotes the number of distinct selected examples divided by the total number of sample-selection slots across optimization rounds, while **Max Repeat** records the maximum number of times that any single example is reused. **Error Shortage** denotes the number of optimization rounds in which the available attributed error samples are insufficient to fill the designated error slots.

First, Related/Diagnosed ranges from 35.83% to 72.50%, indicating that a substantial fraction of observed failures is not attributed to the agent currently being optimized. This supports the need to filter failed trajectories according to agent responsibility rather than treating all failures as equally relevant. Second, Error Shortage occurs in most settings, and Selected Error is correspondingly supplemented by Correct Fill. This shows that attributed failures alone are often insufficient to construct the optimization batch, while correct examples provide complementary references that maintain the size and preserve successful behaviors.

Finally, attribution-guided sampling results in a relatively concentrated selection pattern. Unique Selected ranges from 50.00% to 66.66%, with individual examples reused up to seven times. By comparison, randomly selecting 10 samples from the same pool of 50 over 12 optimization events would cover approximately 93.13% of the pool, or about 47 unique samples, in expectation, whereas our method selects four samples per event and covers roughly 25–33 unique examples. These observations show that responsibility attribution concentrates prompt optimization on a smaller set of agent-relevant cases while reducing exposure to unrelated failures. However, excessive concentration may also reduce sample diversity, potentially limiting generalization when optimization repeatedly focuses on a narrow subset of failures, as observed for Sequential MATH in Table 1.

Table 10: Attribution-Guided Sampling Statistics across Datasets and Agent Topologies.

Graph	Dataset	Related/ Diagnosed	Selected Error	Correct Fill	Unique Selected	Max Repeat	Error Shortage
Sequential	AGIEval	53.77	47.91	52.08	56.25	5	1
	AQuA	36.64	41.66	58.33	60.41	4	3
	GPQA	72.50	50.00	50.00	66.66	4	0
	HumanEval	35.83	37.50	62.50	60.41	5	4
	MATH	46.01	47.91	52.08	60.41	7	1
	MBPP	43.86	43.75	56.25	54.16	4	2
Hierarchical	AGIEval	43.37	47.91	52.08	66.66	3	1
	AQuA	50.03	45.83	54.16	58.33	7	2
	GPQA	56.66	39.58	60.41	64.58	3	3
	HumanEval	50.83	50.00	50.00	54.16	5	0
	MATH	41.12	43.75	56.25	56.25	4	3
	MBPP	66.46	50.00	50.00	50.00	4	0

C Theoretical Derivation

C.1 Bias–Variance Decomposition of Candidate Prompt Gain Estimation Error

We provide the detailed derivation of the candidate prompt gain estimation error used in Section 5. Let the current prompt configuration be P_t, and let candidate prompt $p'_{i,t}$ replace only agent v_i, resulting in the candidate configuration $P_t^{(i\leftarrow p'_{i,t})}$. The expected gain of the candidate prompt on the true data distribution $\mathcal{D}$ is:

$$\Delta(p'_{i,t}) = \mathbb{E}_{(q,y)\sim\mathcal{D}}\left[R_{\text{task}}\big(\Phi(q;G,P_t^{(i\leftarrow p'_{i,t})}),y\big) - R_{\text{task}}\big(\Phi(q;G,P_t),y\big)\right].$$

Practical optimization cannot directly compute $\Delta(p'_{i,t})$, and can only construct an estimator $\hat{\Delta}(p'_{i,t})$ based on a batch B. Therefore, we care about the mean squared error between the estimator and the true gain:

$$\text{MSE}(\hat{\Delta}(p'_{i,t})) = \mathbb{E}\left[(\hat{\Delta}(p'_{i,t}) - \Delta(p'_{i,t}))^2\right],$$

where the expectation is taken over the sampling and stochasticity involved in evaluating a fixed candidate prompt.

For simplicity, we denote $\hat{\Delta}(p'_{i,t})$ and $\Delta(p'_{i,t})$ as $\hat{\Delta}$ and Δ, respectively. Then:

$$\hat{\Delta} - \Delta = \big(\hat{\Delta} - \mathbb{E}[\hat{\Delta}]\big) + \big(\mathbb{E}[\hat{\Delta}] - \Delta\big).$$

Squaring both sides and taking expectation gives:

$$\begin{aligned}\mathbb{E}\left[(\hat{\Delta}-\Delta)^2\right] &= \mathbb{E}\left[\Big(\hat{\Delta} - \mathbb{E}[\hat{\Delta}] + \mathbb{E}[\hat{\Delta}] - \Delta\Big)^2\right] \\ &= \mathbb{E}\left[\Big(\hat{\Delta} - \mathbb{E}[\hat{\Delta}]\Big)^2\right] + \mathbb{E}\left[\Big(\mathbb{E}[\hat{\Delta}] - \Delta\Big)^2\right] \\ &\quad + 2\mathbb{E}\left[\Big(\hat{\Delta} - \mathbb{E}[\hat{\Delta}]\Big)\Big(\mathbb{E}[\hat{\Delta}] - \Delta\Big)\right].\end{aligned}$$

Since Δ is a constant for a fixed candidate prompt, the second term can be written as:

$$\Big(\mathbb{E}[\hat{\Delta}] - \Delta\Big)^2.$$

Meanwhile, the cross term satisfies:

$$\mathbb{E}\left[\hat{\Delta} - \mathbb{E}[\hat{\Delta}]\right] = 0.$$

Therefore:

$$\text{MSE}(\hat{\Delta}) = \Big(\mathbb{E}[\hat{\Delta}] - \Delta\Big)^2 + \text{Var}[\hat{\Delta}].$$

This derivation shows that candidate-prompt gain estimation can be characterized from both bias and variance perspectives. In our method, gold-answer guidance reduces the bias caused by misalignment between the feedback objective and the true task objective. Complementarily, attribution-based failure filtering reduces agent-irrelevant variation during prompt construction, thereby producing candidate prompts whose estimated gains exhibit lower variance on the same evaluation batch.

C.2 Gold-Answer Guidance and Candidate-Gain Bias

Consider a candidate prompt update from $p_{i,t}$ to $p'_{i,t}$. For an example (q, y), define the true task-level improvement as

$$\delta(q) = \left[R_{\text{task}}\big(\Phi(q; G, P_t^{(i \leftarrow p'_{i,t})}), y\big) - R_{\text{task}}\big(\Phi(q; G, P_t), y\big) \right],$$

where $\Phi(q; G, P_t)$ and $\Phi(q; G, P_t^{(i \leftarrow p'_{i,t})})$ are the outputs before and after the update, respectively. The true expected gain is

$$\Delta = \mathbb{E}_{(q,y)\sim\mathcal{D}}[\delta(q)].$$

For analysis, we model the proxy feedback used for candidate evaluation improvement as

$$\widetilde{\delta}(q) = \delta(q) + b(q) + \epsilon(q),$$

where $b(q)$ is the systematic mismatch between the proxy objective and task correctness, and $\epsilon(q)$ is zero-mean noise. For a batch B,

$$\hat{\Delta}_{\text{proxy}} = \frac{1}{|B|} \sum_{q_k \in B} \widetilde{\delta}(q_k).$$

Since $\mathbb{E}[\epsilon(q)] = 0$,

$$\mathbb{E}[\hat{\Delta}_{\text{proxy}}] = \mathbb{E}[\delta(q)] + \mathbb{E}[b(q)] = \Delta + \mathbb{E}[b(q)],$$

and hence

$$\text{Bias}(\hat{\Delta}_{\text{proxy}}) = \mathbb{E}[b(q)].$$

Our method instead evaluates candidates using the gold answer:

$$\hat{\Delta}_{\text{gold}} = \frac{1}{|B|} \sum_{q_k \in B} \left[R_{\text{task}}\big(\Phi(q_k; G, P_t^{(i \leftarrow p'_{i,t})}), y_k\big) - R_{\text{task}}\big(\Phi(q_k; G, P_t), y_k\big) \right].$$

If task correctness is evaluated exactly, then

$$\mathbb{E}[\hat{\Delta}_{\text{gold}}] = \Delta, \qquad \text{Bias}(\hat{\Delta}_{\text{gold}}) = 0.$$

More generally, let the residual bias of a gold-conditioned evaluator be b_{gold}. If it is better aligned with task correctness than the generic proxy feedback, namely

$$|b_{\text{gold}}| < |\mathbb{E}[b(q)]|,$$

then

$$\text{Bias}^2(\hat{\Delta}_{\text{gold}}) < \text{Bias}^2(\hat{\Delta}_{\text{proxy}}).$$

Thus, gold-answer guidance reduces candidate-gain estimation bias by using an evaluation signal that is more closely aligned with the final task objective.

C.3 Responsibility Attribution and Candidate-Gain Variance

We next analyze how responsibility attribution affects the variability of the estimated gains of candidate prompts constructed from different failure samples. Unlike the preceding bias analysis, which considers a fixed candidate prompt, here the candidate prompt varies with the sampled failure trajectories.

Since the solved sample pool is randomly sampled in the same manner across different optimization runs, we focus on the variation introduced by the sampled failure trajectories. For a target agent v_i, $\mathcal{F}$ denotes the sampled failure trajectories used for prompt optimization, and let

$$p'_{i,t}(\mathcal{F}) \sim \mathcal{O}(p_{i,t}, \mathcal{F})$$

denote the resulting candidate prompt. For a fixed batch B, the estimated gain of the candidate prompt constructed from failure set $\mathcal{F}$ is denoted by

$$\hat{\Delta}_{\text{gold}}(\mathcal{F}; B) = \frac{1}{|B|} \sum_{q_k \in B} \left[R_{\text{task}}\big(\Phi(q_k; G, P_t^{(i \leftarrow p'_{i,t}(\mathcal{F}))}), y_k\big) - R_{\text{task}}\big(\Phi(q_k; G, P_t), y_k\big) \right].$$

Throughout the following analysis, B is held fixed, and the randomness arises only from the sampled failure trajectories $\mathcal{F}$ used for candidate-prompt construction.

We model the estimated gain on the fixed batch B as

$$\hat{\Delta}_{\text{gold}}(\mathcal{F}; B) = \mu_i + \epsilon_{\mathcal{F}} + I_{\mathcal{F}} \xi_{\mathcal{F}},$$

where μ_i denotes the expected gain of agent-relevant prompt optimization, $\epsilon_{\mathcal{F}}$ captures the ordinary variation induced by different agent-relevant failure sets, and $I_{\mathcal{F}} \xi_{\mathcal{F}}$ represents the additional variation caused by agent-irrelevant failures entering prompt construction. Here, $I_{\mathcal{F}} = 1$ if agent-irrelevant failures affect the generated candidate prompt and $I_{\mathcal{F}} = 0$ otherwise.

Conditioned on the batch B, assume that

$$\mathbb{E}_{\mathcal{F}}[\epsilon_{\mathcal{F}} \mid B] = \mathbb{E}_{\mathcal{F}}[\xi_{\mathcal{F}} \mid B] = 0,$$

with

$$\text{Var}_{\mathcal{F}}[\epsilon_{\mathcal{F}} \mid B] = \sigma_\epsilon^2, \qquad \text{Var}_{\mathcal{F}}[\xi_{\mathcal{F}} \mid B] = \sigma_\xi^2.$$

We further assume that $I_{\mathcal{F}}$ is independent of $\xi_{\mathcal{F}}$, and that $\epsilon_{\mathcal{F}}$ is independent of $I_{\mathcal{F}} \xi_{\mathcal{F}}$.

Let $p = P(I_{\mathcal{F}} = 1)$ denote the probability that agent-irrelevant failures affect candidate-prompt construction. Then

$$\text{Var}_{\mathcal{F}} \left[\hat{\Delta}_{\text{gold}}(\mathcal{F}; B) \mid B \right] = \sigma_\epsilon^2 + p\sigma_\xi^2.$$

Under random failure sampling, failures attributable to other agents may enter $\mathcal{F}$, resulting in a larger probability p_{rand}. Responsibility-aware sampling instead constructs $\mathcal{F}$ from failures attributed to the target agent, resulting in a smaller probability p_{attr}. When attribution is informative,

$$p_{\text{attr}} < p_{\text{rand}}.$$

Assuming that the baseline variation induced by agent-relevant failures is the same under the two sampling strategies, we obtain

$$\begin{aligned} &\text{Var}_{\mathcal{F}} \left[\hat{\Delta}^{\text{rand}}_{\text{gold}}(\mathcal{F}; B) \mid B \right] - \text{Var}_{\mathcal{F}} \left[\hat{\Delta}^{\text{attr}}_{\text{gold}}(\mathcal{F}; B) \mid B \right] \\ &\quad = (p_{\text{rand}} - p_{\text{attr}})\, \sigma_\xi^2 > 0. \end{aligned}$$

Therefore,

$$\text{Var}_{\mathcal{F}} \left[\hat{\Delta}^{\text{attr}}_{\text{gold}}(\mathcal{F}; B) \mid B \right] < \text{Var}_{\mathcal{F}} \left[\hat{\Delta}^{\text{rand}}_{\text{gold}}(\mathcal{F}; B) \mid B \right].$$

Thus, when candidate prompts generated from different failure sets are evaluated on the same batch B, random failure sampling leads to larger gain variation because agent-irrelevant failures may perturb prompt construction. Responsibility attribution filters such failures, making the resulting candidate prompts more consistently targeted to the responsible agent and yielding more stable estimated gains on the same evaluation batch.

D Implementation Details

D.1 Models and Inference Settings

All experiments keep the parameters of the underlying LLMs fixed. The multi-agent backbone model is Qwen3-8B. Both the optimizer and the error-attribution module use Gemini-2.5-Pro. The inference temperature of all MAS agents is set to 0, so that the optimization curves mainly reflect prompt changes rather than sampling randomness. The optimizer temperature is set to 0.7, which encourages candidate prompt diversity. All models are deployed in non-thinking inference mode.

D.2 Optimization Hyperparameters

As in MASPO, we randomly sample 50 instances to form the training set. For optimizing an agent's prompt, we use a total of 4 samples to guide candidate prompt generation. Specifically, two samples are drawn from the agent's attributed sample pool—if the pool contains more than two, two are chosen at random; otherwise, all available ones are used. The remaining slots are filled by random draws from the positive references. For MASPO and LABEL, the default optimization sample size is 10. Candidate prompts are evaluated by the joint reward consisting of final-answer transition, successor compatibility, and prompt quality. The weights of successor compatibility and prompt quality are set to $\lambda_{\text{succ}} = 0.2$ and $\lambda_{\text{prompt}} = 0.1$, respectively. A regression penalty of $\eta = 1.5$ is applied when a candidate prompt turns an originally correct sample into a wrong one. We optimize the four agents sequentially over three cycles, resulting in 12 prompt-update rounds in total. Each prompt update uses a four-sample optimization batch constructed according to the sampling strategy described above. Within each round, the optimizer performs three independent candidate-generation attempts using the same optimization batch. Therefore, each agent undergoes three prompt updates and nine candidate-generation attempts in total.

D.3 Evaluation Protocol

For MATH and AGIEval, final answers can take diverse forms, making rule-based normalization potentially unreliable. We therefore use an LLM judge to evaluate answer equivalence. The judge receives the problem, the gold answer, and the MAS output, and outputs only `True` or `False`. For GPQA and AQuA, we extract the predicted answer option and compare it directly with the gold option after standardization. Because the answer space is limited to discrete choices, this procedure avoids ambiguity from diverse answer formats. For code tasks, we always provide the function name. Since MBPP problem statements do not specify the target function name, we include the test examples to make it explicit and prevent the optimization from shifting toward guessing function names rather than improving code-generation performance. This also follows the original MBPP protocol(Austin et al., 2021). In contrast, HumanEval problem statements already include the function name, so no test examples are provided.

D.4 Multi-Agent Architectures

The Sequential architecture uses an alternating refinement workflow with four agents, following the order Predictor $\rightarrow$ Reflector $\rightarrow$ Predictor $\rightarrow$ Reflector. The Hierarchical architecture uses three parallel sub-agents followed by an aggregation agent. The sub-agents independently generate candidate solutions from different role perspectives, and the aggregation agent synthesizes their outputs into the final answer.

E Case Study

To further analyze the influence of different prompt optimization strategies on the multi-agent system, we select a probability problem from AGIEval as a case study. The problem asks to independently randomly select integers a, b, c, d from 0 to 2007 and compute the probability that $ad - bc$ is even.

The interval $[0, 2007]$ contains 2008 integers, among which the number of odd and even integers is both 1004. Therefore, for any randomly selected integer x,

$$P(x \text{ is odd}) = P(x \text{ is even}) = 1/2.$$

The product ad is odd if and only if both a and d are odd, so

$$P(ad \text{ is odd}) = 1/4.$$

Thus,

$$P(ad \text{ is even}) = 3/4.$$

Similarly,

$$P(bc \text{ is odd}) = 1/4, P(bc \text{ is even}) = 3/4.$$

Since $ad - bc$ is even if and only if ad and bc have the same parity, that is, both are even or both are odd. Also, since ad is determined by a, d, bc is determined by b, c, and the four variables are independent, the parities of ad and bc are independent. Therefore,

$$P(ad - bc \text{ is even}) = 3/4 \cdot 3/4 + 1/4 \cdot 1/4 = 5/8.$$

On this example, MASPO outputs $9/16$, LABEL outputs 1, and Ours produces the correct answer, $5/8$. Although some sub-agents in MASPO—and even multiple sub-agents in LABEL—generate the correct answer, their aggregation agents still select incorrect results. This suggests that fixed-sample optimization or gold-answer feedback alone cannot ensure reliable aggregation under conflicting reasoning paths. In contrast, Ours correctly selects $5/8$ despite disagreement among sub-agents. This case illustrates how answer-guided feedback and attribution filtering can help preserve task-consistent outputs under conflicting reasoning paths.

F Prompt Templates

We report the faithful prompt-template skeletons used in our implementation. To improve readability, we omit only lengthy role instructions directly inherited from the MAS backbone and minor wording variations across task types. All input fields, supervision signals, output constraints, and executable placeholders are otherwise kept unchanged. In particular, we preserve the optimizer, attribution, and evaluator prompts faithfully to the code. Across all templates, {question} and {context} remain executable placeholders. The sanitizer rejects any optimized prompt that removes required placeholders or introduces unsupported ones.

F.1 Initial Agent Prompt Templates

For mathematical and reasoning tasks, predictor-style agents are initialized with prompts that require step-by-step reasoning and a minimal final answer enclosed in <answer> tags:

> Let's think step by step. Show your final answer bracketed between <answer> and </answer> tags.
> {context}
> Question: {question}
> Reasoning:

Reflector-style agents receive the previous agent outputs as context and are initialized to critique or refine them:

> Please review the solution above and criticize where it might be wrong. Show your final answer bracketed between <answer> and </answer>.
> Question: {question}
> Solution: {context}
> Feedback:
> Correct answer:

For code generation tasks, agents are initialized as Python programming agents.

> You are an expert Python programmer. Write clean, efficient, and well-documented code.
> Problem: {question}
> {context}
> Provide a complete and correct code implementation in Python.
> Code:

In Hierarchical MAS, the aggregation agent receives the question and outputs from preceding agents. It is prompted to synthesize these responses and produce the final answer in the task-specific required format:

> You are a task summarizer. Use the input question and responses from previous agents as reference to derive the final answer.
> Input Question: {question}
> Content from Previous Agents:
> {context}
> Your response:

For mathematical reasoning, the summarizer is required to reason step by step and place its final answer in <answer> tags. For multiple-choice tasks, it must output only the selected option (e.g., <answer>A</answer>) within the tags. For code-generation tasks, it must return an efficient, self-contained Python function in a Markdown code block.

F.2 Prompt Optimization Template Before Answer Guidance

The original MASPO-style optimizer generates a new prompt from sampled execution traces without explicitly exposing the gold answer. The implementation uses task-specific variants. For mathematical tasks, the template is:

> You are optimizing a prompt for a specific agent in a multi-agent mathematical reasoning system.
> CRITICAL: The agent's core role and responsibilities MUST be preserved in the optimized prompt.
> Agent Type: {agent_type}
> Current System Role: {role_description}
> Sample Execution Traces (Question + Context + Agent Output)
> {samples}
> Requirements:
> {requirements}
> Reference prompt:
> {prompt}
> Provide your analysis, optimization points, and the complete optimized prompt using the following XML format:
> <analyse>Analyse what drawbacks exist in the results produced by the reference prompt and how to improve them.</analyse>
> <modification>One sentence summary of the key improvement</modification>
> <prompt>Provide the complete optimized prompt</prompt>

For code tasks, the corresponding template additionally asks the optimizer to inspect coding errors, while keeping the optimized prompt under a concise length budget.

F.3 Gold-Guided Prompt Optimization Template

With answer guidance, the prompt optimizer receives the same structural information as above, but each selected sample is augmented with a gold signal and a sample role. The gold signal is never provided to the MAS during final inference.

The gold-guided optimization template is:

> GOLD-GUIDED PROMPT OPTIMIZATION
> You are optimizing a prompt for a specific agent in a multi-agent mathematical reasoning system.
> Use the gold signal only to diagnose general failure patterns; never copy any gold answer or sample-specific shortcut into the optimized prompt.
> Agent Type: {agent_type}
> Current System Role: {role_description}
> Gold-Guided Sample Execution Traces(Question + Context + Agent Output + Gold Answer)
> {samples}
> Optimization Requirements:
> {requirements}
> Reference prompt:
> {prompt}
> Optimization procedure:
> 1. Compare the agent output and gold-guided outcome to identify the general failure type.
> 2. Determine the underlying issue responsible for the failure.
> 3. Rewrite the prompt to fix the general failure pattern while preserving the agent role and placeholders.
> 4. If this is an intermediate agent, improve the usefulness of its output for successor agents instead of forcing it to produce a final answer.
> 5. Do not include gold answers, test cases, sample-specific facts, or dataset-specific

shortcuts in the optimized prompt.
Return exactly:
<analyse>Gold-guided diagnosis of the prompt weakness and the general fix.</analyse>
<modification>One sentence summary of the prompt change.</modification>
<prompt>The complete optimized prompt.</prompt>

F.4 Actual Sample Block Used by Our Method

Trace attribution is used only to identify the target agent and select informative failure cases. Its repair suggestions are not provided to the prompt optimizer. Instead, each selected correct or failed execution trace is formatted as a sample block and inserted into the {samples} field of the gold-guided optimization template:

Problem {i}:
{question}
Gold Answer:
{gold_answer}
Context:
{local_context}
Agent Output:
{agent_output}
Sample Role: correct_exemplar or failure_case
Sample Guidance: {sample_instruction}

For a sample in solved sample pool, the sample instruction is:

> This agent output passes the available gold/test signal. Use it as a positive example of behavior to preserve; do not over-correct it.

For a failure case, the sample instruction is:

> This agent output fails the available gold/test signal. Use it to infer a transferable prompt weakness; do not copy the gold/test content.

The following requirement block is inserted into the {requirements} field of the gold-guided optimization template:

> Ensure the agent's role, responsibilities, and input format remain consistent. Use gold-guided feedback only to identify general failure modes; never copy a gold answer, test case, or sample-specific shortcut into the prompt. Treat correct_exemplar samples as behavior to preserve and failure_case samples as behavior to repair. Do not let failure cases override stable correct behavior unless they reveal a general weakness. For intermediate agents, improve how their output supports successor agents instead of forcing them to output a final answer when that is not their role. Preserve collaboration with predecessor and successor prompts. The optimized prompt should make a substantive, role-preserving change from the reference prompt when the samples reveal any actionable weakness; do not simply restate the reference prompt.

Thus, error attribution influences optimization indirectly through sample selection: it determines which execution traces are formatted and inserted into {samples}.

F.5 Trace-Level Error Attribution Template

For wrong trajectories, the attribution module receives the problem, the final extracted output, the gold answer or tests, and a compact full trace containing each agent's local output. The actual attribution template is:

> You are performing trace-level failure attribution for a multi-agent prompt optimization experiment. Your goal is not to solve the problem again. Inspect the complete compact execution trace, compare the final output with the gold answer/tests, and identify which agent(s) or stage most likely introduced the failure.

> Task type: {task_type}
> Graph type: {graph_type}
> Problem: {problem}
> Gold answer / tests / reference: {gold_signal}
> Final extracted output: {final_output}
> Execution trace: {trace_block}
> Rules: use only trace evidence; do not invent hidden behavior; multiple responsible agents are allowed; for Hierarchical graphs, do not assign the summarizer merely because it produced the wrong final answer; do not copy the gold answer or tests into repair advice.
> Return ONLY a JSON object with fields confidence, responsible_agents, failure_stage, failure_cause, evidence, agent_repair_advice, and prompt_repair_advice.

Although the attribution prompt asks for repair advice, the main sample-filter-only setting uses only responsible_agents to select failure cases for the current target agent.

F.6 Reward Scoring and Judge Templates

Candidate prompts are evaluated by replacing only the target agent prompt and comparing the resulting behavior with the current prompt configuration. The joint reward contains the final-answer transition score, successor compatibility, and intrinsic prompt quality. The successor-compatibility judge receives the problem, gold signal, downstream agent type and prompt, and two upstream outputs:

> You are evaluating two candidate outputs from an upstream agent in a multi-agent system. The upstream agent may produce intermediate reasoning, a plan, critique, summary, code, or a final answer. Do NOT require the upstream output itself to contain the final answer. Choose the output that is more compatible with the downstream agent and more useful for eventually reaching the gold answer. Prefer outputs that preserve relevant constraints, expose useful intermediate state, avoid misleading facts, and are easy for the downstream prompt to consume. Respond ONLY with "A" or "B".

The intrinsic prompt-quality judge compares the old and candidate prompts directly:

> Evaluate the prompt text itself, not the sampled answer quality. Prefer the candidate only if it preserves the original role, input placeholders, and responsibilities; gives clearer, more actionable guidance without hard-coding any specific training sample or gold answer; helps collaboration with predecessor and successor agents; and avoids excessive verbosity, over-specific dataset tricks, and instructions that could harm downstream agents. Respond ONLY with "A" or "B".

When LLM-based answer judgment is used for mathematical or choice tasks, the judge prompt is:

> Below is a problem, its standard answer, and a candidate solution. Determine whether the candidate solution's final answer is mathematically equivalent to the standard answer. Accept equivalent forms such as simplified radicals, fractions/decimals, reordered sets, and equivalent interval notation. Do not require identical wording or formatting. Only output True or False, nothing else.
> Problem: {problem}
> Standard answer: {gold}
> Candidate solution: {raw_output}
> Response: