MAMR-GRPO: REINFORCEMENT LEARNING WITH EXECUTION-STRUCTURED CREDIT FOR MULTI-AGENT LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based multi-agent systems (MAS) extend the capabilities of language models by decomposing complex tasks into specialized roles, enabling parallel reasoning, information gathering, and iterative refinement. In such systems, the final answer is produced collaboratively by multiple agents or roles whose outputs interact and influence subsequent executions. However, prevailing reinforcement learning methods such as Group Relative Policy Optimization (GRPO) typically assign credit at the rollout level, making it difficult to distinguish the contributions of different executions within a multi-agent trajectory. We propose Multi-Agent Multi-Round GRPO (MAMR-GRPO), which retains rollout-level comparisons while introducing local credit assignment aligned with the execution structure of MAS. Specifically, round boundaries, role dependencies, and parallel outputs provide references for comparing task assignment, subtask execution, and information aggregation. These local credits are combined with the terminal rollout advantage and applied to the corresponding generated segments of a shared LLM policy. Across seven search-augmented QA benchmarks, MAMR-GRPO reaches 46.7 average exact match, 3.1 points above trajectory-level GRPO on the same graph. Budget ablations show further gains with additional subagents and rounds.

1 INTRODUCTION

LLM-based multi-agent systems (MAS) solve complex tasks by coordinating multiple agent executions for planning, subtask execution, and information aggregation (Wu et al., 2024; Hong et al., 2024). Parallel execution and repeated delegation make the final answer depend on both the work performed by individual agents and how their outputs guide later work (Xu et al., 2026). Training this collaboration therefore requires assigning credit across interdependent executions, although outcome supervision typically provides only a terminal reward. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) compares rewards within a group of responses to the same task and has been extended to multi-agent training (Mo et al., 2025; Xu et al., 2026).

However, in trajectory-level GRPO, a complete rollout receives one group-relative advantage that is shared by all of its generated segments. This feedback reflects the quality of the overall rollout but does not distinguish the contributions of individual executions. As a result, useful and unhelpful executions within the same trajectory can receive identical updates: an erroneous intermediate decision may be reinforced when the final answer succeeds, while a useful subtask solution may be penalized when other parts of the trajectory fail. Such coarse credit assignment is particularly problematic in MAS, where different roles and rounds can have substantially different effects on subsequent computation and the final answer.

Recent methods assign finer credit. GiGPO compares actions at repeated environment states (Feng et al., 2025). M-GRPO gives separate advantages to main-agent and subagent trajectories; its subagent reward uses LLM evaluation of execution quality (Hong et al., 2025). SHARP estimates worker contributions by counterfactual removal (Li et al., 2026c). LLM-as-a-Judge assessments can favor verbose outputs and misgrade reasoning answers (Zheng et al., 2023). Such errors can enter local rewards, and each judgment adds a model call. We need a credit-assignment method that retains GRPO's within-group comparison while distinguishing contributions across roles and rounds.

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.

We therefore propose MAMR-Framework, a multi-round collaboration framework, together with Multi-Agent Multi-Round GRPO (MAMR-GRPO), a reinforcement learning method designed to train agents within this framework. A rollout in MAMR-Framework proceeds through multiple rounds, each consisting of a META AGENT that assigns tasks, parallel SUB AGENT executions that perform the assigned work, and a CONTROLLER AGENT that aggregates information and decides whether another round is needed. All three roles share a single LLM policy with role-specific instructions.

Our local credit assignment uses counterfactual value estimation. After each collaboration round, we introduce a *virtual stopping point*: before the actual rollout proceeds, we counterfactually terminate the collaboration at the current round and generate a final answer from the information available so far. The reward of this counterfactual answer defines the *round value*. The actual rollout then follows the CONTROLLER AGENT's decision to stop or continue. For task assignment, the preceding round value serves as the baseline, and the local credit is defined as the increase in round value from the previous round to the current round, with zero as the first-round reference. For subtask execution, we construct removal counterfactuals by deleting subagent outputs and measuring the resulting change in round value. Unlike SHARP (Li et al., 2026c), which evaluates individual worker removal, we also remove sampled groups of subagents to capture contributions obscured by redundant information. For aggregation, the current round value serves as the reference for discounted value gains in subsequent rounds conditioned on the generated summary. The update for each generated segment combines its role-specific local credit with the terminal group-relative advantage.

We evaluate MAMR-GRPO on seven search-augmented QA benchmarks with Qwen3-4B as the backbone. Under the same execution graph, MAMR-GRPO improves average exact match (EM) from 43.6 to 46.7 over trajectory-level GRPO. It ranks first on four benchmarks, second on Bamboogle and PopQA, and fourth on Natural Questions in Tab. 1. Our ablations further show that increasing the execution budget can improve EM, indicating that MAMR-GRPO can benefit from additional collaborative computation beyond the training-time configuration. These results demonstrate the effectiveness of role-specific credit assignment and its ability to support larger collaboration budgets.

Our contributions are:

- We introduce MAMR-Framework, a multi-round MAS framework with three execution roles: task assignment, parallel subtask execution, and information aggregation.
- We propose MAMR-GRPO, which assigns role-specific local credits using counterfactual round values and output removal, including joint deletion to account for redundant outputs.
- We evaluate MAMR-GRPO on seven search-augmented QA benchmarks, achieving a 3.1-point average EM gain over trajectory-level GRPO and further gains with larger budgets.

2 RELATED WORK

2.1 MULTI-AGENT COORDINATION AND TRAINING

Recent multi-agent RL methods jointly optimize planning and execution. MATPO uses a shared LLM for planner and worker trajectories under different role prompts and extends GRPO to their joint optimization (Mo et al., 2025). WideSeek-R1 similarly shares parameters between a lead agent and parallel subagents, assigning a common terminal advantage within each rollout (Xu et al., 2026). MAMR-Framework follows this shared-policy design but introduces aggregation as a separate execution role: the CONTROLLER AGENT summarizes subagent outputs and provides context for subsequent rounds. MAMR-GRPO then assigns role-specific local credits to task assignment, subtask execution, and aggregation while retaining the terminal advantage.

2.2 LOCAL CREDIT ASSIGNMENT

Several methods refine credit below the rollout level. GiGPO compares actions taken at repeated environment states (Feng et al., 2025), GRPO-MA estimates thought-level values from sampled answer continuations (Wang et al., 2026), and M-GRPO introduces separate feedback for subagents using answer correctness and LLM-based evaluation (Hong et al., 2025). TRACE instead derives temporal value changes along a trajectory and combines them with terminal group-relative advantages (Tao et al., 2026). MAMR-GRPO similarly combines local and terminal feedback, but defines

counterfactual round values at collaboration boundaries to credit both current task assignment and subsequent progress enabled by aggregation.

Counterfactual methods estimate contribution by intervening on selected executions. SHARP uses individual worker removal to compute marginal credits (Li et al., 2026c), while C3 samples alternative decisions and evaluates their regenerated downstream trajectories (Chen et al., 2026). MAMR-GRPO evaluates both individual and joint subagent deletion within a round, allowing it to capture contributions hidden by redundant outputs, while also assigning separate temporal credits to task assignment and aggregation. Appendix B compares further methods.

3 MAMR-FRAMEWORK

Fig. 1-A shows the execution order within a round: a META AGENT assigns subqueries, parallel SUB AGENT executions return outputs, and a CONTROLLER AGENT summarizes the outputs and chooses whether to continue or stop. A continuation passes the summary and original task to the next META AGENT execution; a stop triggers final-answer generation. Each agent execution instantiates one protocol role using the shared LLM policy π_θ with a role-specific instruction template. For an input x, execution produces $T \leq T_{\max}$ rounds and an answer z. Let h_t be the summary after round t, with $h_0 = \varnothing$. We omit fixed role instructions and tool-environment state from policy conditionals.

Context-dependent task assignment. At round t, the META AGENT uses the task and previous summary to generate subtask specifications:

$$\mathcal{D}_t \sim \pi_\theta(\cdot \mid x, h_{t-1}), \qquad \mathcal{D}_t = \{d_{t,j}\}_{j=1}^{n_t}, \quad n_t \leq n_{\max}. \tag{1}$$

Both the number of subagents and their subtasks depend on the current task and round context. Each specification contains the assigned task, the context needed to execute it, and the required output. SUB AGENT j executes its specification in an isolated context and returns $y_{t,j}$. Its execution can include reasoning and tool interactions: generated tool requests are followed by observations appended to its context. Parallel subagents cannot access peer outputs.

Information aggregation. The CONTROLLER AGENT receives the returned outputs and their task descriptions. Writing $\bar{d}_{t,j}$ for the description passed to the CONTROLLER AGENT, including the task name and instructions, its input is

$$\mathcal{C}_t = \{(\bar{d}_{t,j}, y_{t,j})\}_{j=1}^{n_t}, \qquad c_t = (x, h_{t-1}, t, \mathcal{C}_t). \tag{2}$$

It generates a prefix p_t followed by a boundary decision in the same autoregressive sequence:

$$(p_t, \delta_t) \sim \pi_\theta(\cdot \mid c_t), \qquad \delta_t \in \{\texttt{continue}, \texttt{stop}\}. \tag{3}$$

The prefix p_t contains the CONTROLLER AGENT output, including an updated summary field h_t, which is extracted from p_t and carried forward to the next round; it may also contain reasoning or formatting text preceding the decision value. Current-round answering and forced-stop evaluation use c_t and p_t, including the current SUB AGENT records. Deletion-based evaluation uses a reduced CONTROLLER AGENT context and a regenerated prefix. If the rollout continues, the next META AGENT receives the extracted summary h_t together with the original task. Later subagents also receive metadata for earlier retrievals.

Continuation and answering. If $\delta_t = \texttt{continue}$ and $t < T_{\max}$, another round begins from (x, h_t). Otherwise, execution ends at $T = t$ and produces

$$z \sim \pi_\theta(\cdot \mid c_T, p_T, \texttt{stop}). \tag{4}$$

At the round limit, the decision is set to `stop` regardless of the sampled value.

These boundaries define the references used in Section 4: the state before assignment, records with or without selected outputs, and summary-conditioned continuation. Appendix D specifies prompts and tool context; Appendix C illustrates a full rollout.

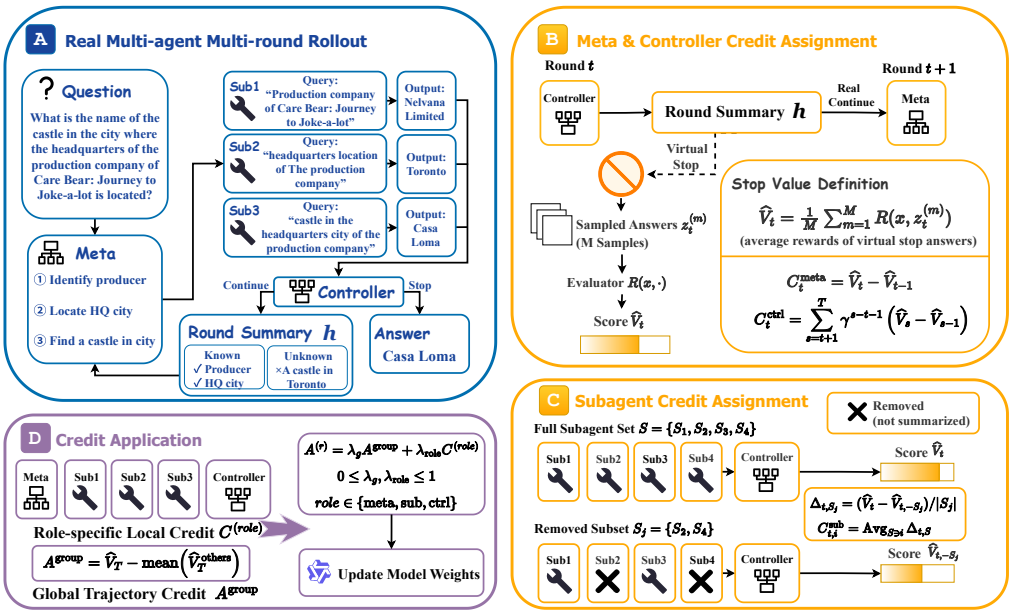

Figure 1: Execution and credit assignment in MAMR-GRPO. (A) A shared policy assigns tasks, runs parallel SUB AGENTs, and carries a CONTROLLER AGENT summary into later rounds. (B) Forced-stop answers estimate round values for META AGENT and CONTROLLER AGENT credits. (C) Removing sampled SUB AGENT records and regenerating aggregation and answers estimates subagent credit. (D) Each segment's update combines local credit and group-relative advantage.

4 MAMR-GRPO

Fig. 1-B–D summarizes MAMR-GRPO: forced-stop answers estimate round values, output deletion evaluates conditional removal effects, and role-specific credits combine with a terminal group advantage on generated segments. The definitions below state which context is retained or regenerated in each evaluation. In Search-QA, local credits use task-level answer reward without format feedback. The terminal group advantage includes format feedback.

4.1 ROUND VALUE

During rollout collection and counterfactual evaluation, we hold the shared policy π_θ fixed. Optimization updates θ after these samples have been collected. At a completed round t, the *round value* is the expected task reward of an answer generated after a virtual forced stop from the current CONTROLLER AGENT context c_t and prefix p_t (Fig. 1-B):

$$V_t = V(c_t, p_t) = \mathbb{E}_{z\sim\pi_\theta(\cdot|c_t,p_t,\texttt{stop})}\left[R_{\text{task}}(x,z)\right], \tag{5}$$

Here R_{task} is the answer reward, and $V(c,p)$ extends the definition to any CONTROLLER AGENT context and prefix. Following GRPO-MA (Wang et al., 2026), we reduce sampling variance by averaging the rewards of M answers sampled at fixed (c_t, p_t):

$$\widehat{V}_t = \frac{1}{M}\sum_{m=1}^{M} R_{\text{task}}(x, z_t^{(m)}), \tag{6}$$

This counterfactual evaluation samples only answer continuations; the collected rollout and its boundary decision remain unchanged. We set $\widehat{V}_0 = 0$ as the first-round reference.

4.2 LOCAL CREDIT ASSIGNMENT

Task assignment. At the beginning of round t, the META AGENT assigns subtasks to the SUB AGENTs. We evaluate this assignment by the change in round value after these subtasks are executed

and their outputs are aggregated (Fig. 1-B):

$$C_t^{\text{meta}} = \widehat{V}_t - \widehat{V}_{t-1}. \tag{7}$$

For $t > 1$, $\widehat{V}_{t-1}$ estimates the answer quality available before the new assignment. Subtracting it makes the credit reflect progress during the current round. An assignment that repeats already available information can therefore receive little local credit even when the answer remains correct. The first round uses $\widehat{V}_0 = 0$ as the reference for the value reached after its assignment.

Subtask execution. We evaluate subagent outputs by counterfactual deletion (Fig. 1-C). Let $\mathcal{J}_t = \{1, \ldots, n_t\}$ denote the set of subagent indices in round t. We randomly sample a collection $\mathcal{F}_t$ of nonempty proper subsets of $\mathcal{J}_t$, where each $S \in \mathcal{F}_t$ specifies a group of subagents whose outputs are removed in one counterfactual evaluation. For a deletion subset S, we remove the outputs and task descriptions of all subagents in S, hold the surviving records fixed, and recompute the counterfactual round value following the same procedure as in Eq. 6, by regenerating the CONTROLLER AGENT prefix and the forced-stop answers from the modified context. We denote the resulting counterfactual estimate by $\widehat{V}_{t,-S}$ and compare it with the value of the original round.

For subagent j, we define

$$\mathcal{F}_{t,j} = \{S \in \mathcal{F}_t \mid j \in S\}, \tag{8}$$

the collection of sampled deletion subsets that contain j. We distribute the value change caused by deleting S equally among its members and average over all sampled subsets containing j:

$$\Delta_{t,S} = \frac{\widehat{V}_t - \widehat{V}_{t,-S}}{|S|}, \qquad C_{t,j}^{\text{sub}} = \text{Avg}_{S \in \mathcal{F}_{t,j}} \Delta_{t,S}. \tag{9}$$

Unlike SHARP (Li et al., 2026c), which evaluates credit through individual worker removal, we additionally consider joint deletion of multiple subagents. Individual deletion can underestimate a useful subagent when its contribution is redundant with those of other subagents: removing it alone may leave the round value nearly unchanged because the remaining outputs provide the same evidence. By evaluating joint as well as individual deletion, our credit estimator can recover contributions that are masked by such redundancy.

Information aggregation. The CONTROLLER AGENT aggregates evidence after the subagents finish. Its summary h_t becomes the input to the next round's task assignment, so information preserved in the summary can affect which evidence is collected next. Later assignments and subagent executions use this information to produce subsequent round-value gains. We use $\widehat{V}_t$ as the reference and assign these gains to the generating prefix p_t:

$$C_t^{\text{ctrl}} = \begin{cases} \displaystyle\sum_{s=t+1}^{T} \gamma^{s-t} \left(\widehat{V}_s - \widehat{V}_{s-1} \right), & \delta_t = \texttt{continue} \text{ and } t < T, \\ 0, & \text{otherwise}, \end{cases} \qquad \gamma \in [0, 1]. \tag{10}$$

The first term is the next round's value change after its assignment uses h_t; later terms add changes from further rounds. For our setting $\gamma = 1$, the sum equals $\widehat{V}_T - \widehat{V}_t$. Stops receive zero local credit.

4.3 TRAINING OBJECTIVE

For rollout i among $K \geq 2$ rollouts for the same task, let R_i^{term} denote its terminal reward. We use the leave-one-out group advantage

$$A_i^{\text{group}} = R_i^{\text{term}} - \frac{1}{K-1} \sum_{k \neq i} R_k^{\text{term}}. \tag{11}$$

For a generated segment of type $role \in \{\text{meta}, \text{sub}, \text{ctrl}\}$, the segment advantage is

$$A^{(role)} = \lambda_g A_i^{\text{group}} + \lambda_r C^{(role)}. \tag{12}$$

The terminal term supervises overall task success, while the local term provides credit specific to the corresponding execution. The weights λ_g and λ_r scale these terms; we set all four weights to

one. The CONTROLLER AGENT advantage applies to the generated prefix p_t, including the summary. Boundary decision values receive separate supervised action targets, and terminal answers receive answer-level leave-one-out advantages. All roles share the same policy parameters and are optimized jointly; local credits are fixed targets without additional centering.

Details of deletion sampling, answer-sampling budgets, auxiliary supervision, token masks, loss reduction, and optimization are provided in Appendix D.

5 EXPERIMENTS

5.1 EXPERIMENTAL PROTOCOL

Our locally trained systems use Qwen3-4B-Instruct-2507 (Yang et al., 2025). We train MAMR-GRPO on 49,938 questions: 15,000 from 2WikiMultiHopQA (Ho et al., 2020), 10,000 from HotpotQA (Yang et al., 2018), 19,938 from MuSiQue (Trivedi et al., 2022), and 5,000 from Natural Questions (Kwiatkowski et al., 2019). Each question supplies its dataset answer for reward computation. Rollouts retrieve from the Wiki2018 corpus. The task reward is token F1 (Rajpurkar et al., 2016); it is used to estimate round values and local credits. The terminal trajectory reward adds format feedback for role outputs and final answers and averages over sampled terminal answers.

We evaluate one answer per question on a collection of 51,713 examples from these four benchmarks and Bamboogle (Press et al., 2023), PopQA (Mallen et al., 2023), and TriviaQA (Joshi et al., 2017). We report normalized exact match (EM) by benchmark in Tab. 1. Its *Average* is the unweighted mean of the seven benchmark EM scores.

Training questions come from subsets of the four benchmarks' official training splits. Evaluation uses test or development sets from all seven benchmarks. The training and evaluation files share no exact question strings. We train each system on 8 NVIDIA A100 80GB GPUs using VERL for training and vLLM for inference. A single training run takes approximately seven days. In the main experiment, we allow up to $R = 3$ collaboration rounds, with each META AGENT initiating up to $S = 4$ SUB AGENT executions per round.

5.2 BASELINES

We organize the baselines in Tab. 1 into three groups.

Published results. We report selected Qwen2.5-7B results from Search-R1 (Jin et al., 2025) and ReSeek (Li et al., 2026a). The RAG, IRCoT, Search-o1, rejection sampling, and Search-R1 scores in Tab. 1 come from Search-R1's 7B evaluation. These published results use Qwen2.5-7B and the respective papers' training and evaluation protocols.

Locally reproduced methods and evaluations. We reproduce Search-R1 (Jin et al., 2025), C3 SubMessage Lite (Chen et al., 2026), and SHARP-MC (Li et al., 2026c) using Qwen3-4B-Instruct-2507. The purpose of these reproductions is to align major experimental factors, including the training data, backbone model, and training-time compute and retrieval budgets, so that differences more directly reflect the training algorithms. We evaluate the released WideSeek-R1-4B (Xu et al., 2026) with a matched inference budget, its official RLinf implementation, and Wiki2018 retrieval.

Comparisons on MAMR-Framework. We further construct baselines around the MAMR-Framework execution protocol to separate the effects of multi-agent execution and reinforcement learning. The inference-only comparisons include SingleAgent + Qwen3-4B-Ins, MATPO-Framework + Qwen3-4B-Ins (Mo et al., 2025), and MAMR-Framework + Qwen3-4B-Ins, all using the same Qwen3-4B-Instruct-2507 backbone. For the main controlled training comparison, we train MAMR-Framework + GRPO and MAMR-Framework + MAMR-GRPO on the same MAMR-Framework execution graph. MAMR-Framework + GRPO assigns the rollout-level group advantage to all generated agent segments, whereas MAMR-Framework + MAMR-GRPO uses role-specific local credits for intermediate agent outputs and leave-one-out advantages for final answers.

Table 1: Exact match (EM, %) on seven search-augmented QA benchmarks. Average is the unweighted mean of the seven EM scores. **Bold** and *underlined italics* mark the highest and second-highest scores in each column.

	Multi-Hop QA				General QA			
Method	2Wiki	Bamboogle	HotpotQA	MuSiQue	NQ	PopQA	TriviaQA	Avg.
Published reference results (Qwen2.5-7B)								
RAG (Lewis et al., 2020) (Instruct)	23.5	20.8	29.9	5.8	34.9	39.2	58.5	30.4
IRCoT (Trivedi et al., 2023) (Instruct)	14.9	22.4	13.3	7.2	22.4	30.1	47.8	22.6
Search-o1 (Li et al., 2025b) (Instruct)	17.6	29.6	18.7	5.8	15.1	13.1	44.3	20.6
Rejection Sampling (Jin et al., 2025) (Instruct)	29.6	35.5	33.1	12.3	36.0	38.0	59.2	34.8
Search-R1 (Jin et al., 2025) (Base, PPO)	38.2	43.2	*43.3*	*19.6*	**48.0**	45.7	63.8	43.1
Search-R1 (Jin et al., 2025) (Instruct, PPO)	41.4	36.8	37.0	14.6	39.3	39.7	61.0	38.5
ReSeek (Li et al., 2026a) (Instruct)	38.2	39.2	38.9	18.5	*46.9*	**50.1**	64.0	42.3
Local reproductions and evaluations								
Search-R1 (Jin et al., 2025)	47.0	48.8	43.2	17.9	40.2	41.8	64.6	43.3
C3 SubMessage Lite (Chen et al., 2026)	42.8	42.4	39.4	15.3	36.5	39.6	61.9	39.7
SHARP-MC (Li et al., 2026c)	46.3	42.4	39.4	16.7	37.7	41.3	62.8	40.9
WideSeek-R1-4B (Xu et al., 2026)	45.8	**50.4**	39.6	15.1	33.7	37.9	61.5	40.6
Framework comparisons								
SingleAgent + Qwen3-4B-Ins (Yang et al., 2025)	28.0	36.0	26.1	6.7	24.3	28.5	52.2	28.8
MATPO-Framework (Mo et al., 2025) + Qwen3-4B-Ins	25.4	31.2	30.7	7.2	33.9	36.8	60.4	32.2
MAMR-Framework + Qwen3-4B-Ins	34.7	45.6	34.6	12.7	35.2	37.1	60.5	37.2
MAMR-Framework + GRPO	*48.6*	44.0	42.7	17.7	41.5	45.5	*65.4*	*43.6*
MAMR-Framework + MAMR-GRPO	**55.2**	*49.6*	**46.1**	**22.4**	40.9	*46.7*	**66.1**	**46.7**

5.3 Main Results

MAMR-GRPO achieves the highest Average EM in Tab. 1. The controlled comparison on the same MAMR-Framework graph shows a 3.1-point gain over GRPO, with improvements on six of seven benchmarks. With roles and communication fixed, the gain shows that the training procedure affects how well the shared policy uses the execution structure. Within a rollout, task assignments, subagent outputs, and summaries can contribute differently to answer quality. GRPO gives these segments the same rollout-level advantage; MAMR-GRPO adds local credits that reflect their execution roles.

The gains concentrate on multi-hop QA, which requires combining dependent results. The average improvement over GRPO is 5.1 points on multi-hop benchmarks and 0.4 points on general QA. Intermediate progress may matter more when a retrieved entity or relation determines the next search. A subagent may recover a useful fact even when the rollout's final answer is wrong. Counterfactual value estimation can give that output local credit when its removal lowers estimated answer quality. Round-value changes also provide feedback on whether later task assignments make further progress.

The framework comparisons show gains from both execution structure and policy optimization. Before RL, MAMR-Framework already outperforms the SingleAgent and MATPO-Framework configurations with the same backbone, indicating that explicit task assignment and aggregation help organize the pretrained policy's search. Both training methods improve this starting point. The cross-method training statistics in Tab. 3 (Section 5.5) show that MAMR-GRPO assigns more parallel work within each round and uses fewer rounds than GRPO during steps 515–780. The paired checkpoint analysis and case study in that section compare MAMR-GRPO at steps 50 and 600 to examine how its rollout behavior changes during training.

5.4 Subagent and Round Budgets

At the near-end MAMR-GRPO checkpoint, we vary the subagent cap $S = 2, \ldots, 6$ at $R = 3$ and the round cap $R = 1, \ldots, 5$ at $S = 4$ (Figs. 2 and 3). All evaluations use the same 51,713 questions and forced final-answer generation; other inference settings remain fixed. Appendix E reports per-benchmark EM and average generation counts.

Additional subagents improve accuracy, but the benefit varies by benchmark. From $S = 2$ to $S = 6$, Average EM rises from 45.43 to 47.76 while completion tokens per question rise from 2,496 to 5,136. Bamboogle gains 8.00 points, accounting for 1.14 points of the 2.33-point increase in the seven-benchmark average. MuSiQue gains 0.41 points. The concentration is even stronger beyond

$S = 4$: from $S = 4$ to $S = 6$, Bamboogle contributes 1.03 points of the total 1.05-point increase in Average EM. Over the same range, completion tokens increase from 3,735 to 5,136 per question. Thus, increasing the subagent cap beyond four provides limited aggregate benefit outside Bamboogle despite substantially higher generation cost.

Increasing the round cap shows a different pattern. An additional round is especially useful when new subtasks can depend on evidence collected in the first round. Increasing R from 1 to 2 improves Average EM by 2.14 points, including gains of 5.41 on 2Wiki and 4.84 on MuSiQue. MuSiQue's response to a second round is much larger than its 0.41-point gain from increasing S from 2 to 6. The comparison shows why parallel capacity and sequential interaction serve different purposes: subagents within a round share the current task context, whereas a later round can use newly found entities and unresolved relations to assign new tasks. The paired case in Section 5.5 gives a concrete example of this dependency.

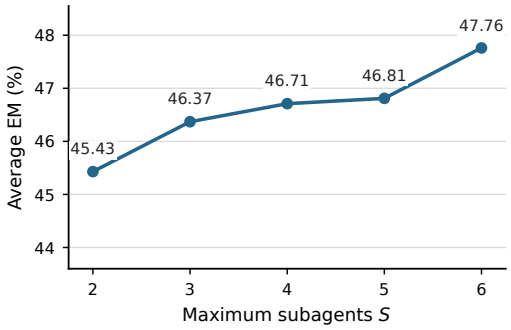

Figure 2: Near-end checkpoint Average EM by subagent cap at $R = 3$.

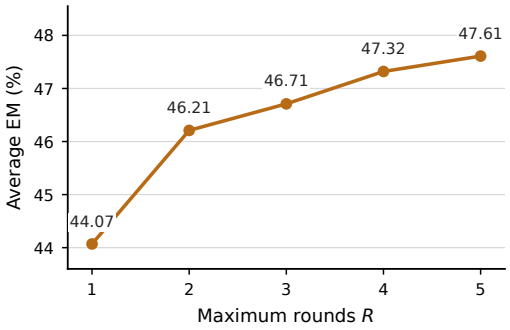

Figure 3: Near-end checkpoint Average EM by round cap at $S = 4$.

Most of the average gain from additional rounds occurs with the second round: its 2.14-point improvement accounts for about 60% of the total 3.54-point gain from $R = 1$ to $R = 5$. The final increase from $R = 4$ to $R = 5$ adds 0.29 points. Nevertheless, performance continues to improve beyond the three-round training limit. At $R = 5$, the policy executes only 1.961 rounds on average, indicating that the larger round cap is not routinely exhausted. Completion tokens increase from 2,058 at $R = 1$ to 4,457 at $R = 5$. Overall, larger budgets provide uneven returns: additional subagents beyond four mainly benefit Bamboogle, whereas most of the improvement from additional rounds comes from enabling a single follow-up round.

5.5 Changes in Rollout Behavior

Early and near-end checkpoints We compare the early (step 50) and near-end (step 600) checkpoints from the same MAMR-GRPO training run on 2,086 questions (Tab. 2). Micro EM rises from 41.4 to 52.7, subagent executions per realized round from 3.248 to 3.993, and first-round stops from 42.6% to 57.2%. Rounds per rollout fall from 1.836 to 1.626. Total subagent executions and tool calls rise by about 9%, while completion tokens remain near 3,600. Accuracy thus improves with more parallel work and fewer rounds at a similar generation budget.

Table 2: Test-set behavior of MAMR-GRPO at steps 50 and 600 on the same 2,086 questions.

Statistic	Step 50	Step 600
Micro EM (%)	41.4	52.7
Rounds / rollout	1.836	1.626
Subagent executions / rollout	5.963	6.493
Executions / realized round	3.248	3.993
Tool calls / rollout	6.366	6.943
Completion tokens / rollout	3,614	3,619
First-round stops (%)	42.6	57.2
Continue decisions (%)	54.22	45.96

MAMR-GRPO and GRPO during training The separate comparison over training steps 515–780 shows the same pattern (Tab. 3). MAMR-GRPO uses 3.987 subagents per realized round versus 3.215 for GRPO, with 25.7% fewer rounds

and 7.8% fewer total subagent executions. Its continue rate is 62.8%, compared with 89.3% for GRPO. Both comparisons show more parallel work per round and less frequent continuation.

Relation to training design The local advantages provide a possible explanation for this allocation. Task-assignment credit measures the current round's value gain (Eq. 7). The first round is measured against zero; later rounds receive credit for further gains. Wider assignments can receive more credit when their outputs improve answer quality. Subagent credit reinforces outputs whose deletion lowers round value (Eq. 9). Together, these signals can favor useful parallel work; they give no explicit bonus for adding agents. Assignments that repeat evidence from earlier rounds receive little local progress credit.

Table 3: Training rollout statistics averaged over steps 515–780 for MAMR-GRPO and GRPO.

Statistic	MAMR-GRPO	GRPO
Rounds / rollout	2.006	2.699
Subagent executions / rollout	7.997	8.677
Executions / realized round	3.987	3.215
Continue decisions (%)	62.8	89.3

Boundary supervision links continuation to the observed next-round value gain (Appendix D.6). For eligible sampled continuations, positive gains favor `continue` and negative gains favor `stop`; sampled stops provide no continuation target. This signal can reduce unproductive rounds while retaining useful follow-up search. GRPO shares this supervision and anti-redundancy shaping, so neither alone explains the cross-method difference. The statistics are consistent with the combined training design; component ablations are needed to identify which signals cause the observed changes.

Case study A paired HotpotQA example compares MAMR-GRPO at steps 50 and 600 on who created the show that aired the "Happy Fun Ball" parody advertisements. At step 50, three subagents identify Saturday Night Live and the ad's writer, Jack Handey. The CONTROLLER AGENT stops after one round and answers Handey, confusing the ad's writer with the show's creator. At step 600, four subagents retrieve both entities, but the CONTROLLER AGENT records the missing creator relation and continues. The next META AGENT assigns four subtasks, including finding the show's creator and checking official credits. The evidence supports the correct final answer, Lorne Michaels.

The later summary preserves an unresolved relation that guides the next assignment. This is the behavior that summary credit can reinforce: Eq. 10 adds subsequent value gains to the advantage of the preceding CONTROLLER AGENT prefix. Task-assignment credit separately rewards progress after the follow-up tasks execute. The case illustrates how these signals can support a useful extra round alongside earlier stopping on average; it does not measure their individual causal effects. Complete records are provided under `case_study/` in the supplementary materials.

6 LIMITATIONS AND FUTURE WORK

Our experiments focus on search-augmented QA and train on subsets of the available benchmark data. The local-credit design follows each execution role's contribution to answer quality. Future work will analyze how local and terminal advantages jointly affect policy-gradient updates and test MAMR-GRPO on other multi-agent tasks and larger training sets.

7 CONCLUSION

MAMR-GRPO assigns local credit to task assignment, subagent execution, and aggregation in a shared-policy execution graph. Round-value estimates and individual and joint output-deletion evaluations use task-level answer rewards; their credits combine with terminal group-relative advantages on generated segments. Across seven search-augmented QA benchmarks, MAMR-GRPO reaches 46.7 Average EM versus 43.6 for trajectory-level GRPO on the same graph. Checkpoint comparisons show more subagents per realized round and more first-round stopping. The budget ablations show higher Average EM at larger caps with higher generation cost. The gains concentrate on multi-hop questions that require combining intermediate findings. The case study illustrates how a summary can carry an unresolved relation into the next round's task assignment. The results show how execution-structured credit assignment improves learning in multi-round search QA.

AI USE STATEMENT

We used generative AI tools to revise manuscript text, discuss research ideas, assist with code implementation, and prepare visualizations. The authors retain responsibility for the research decisions, verification of claims and results, and all submitted text, code, and supplementary materials.

ETHICS STATEMENT

The study uses existing question-answering benchmarks to evaluate training methods for tool-using language-model agents; it does not collect human-subject data. Adaptive multi-agent systems can aggregate evidence incorrectly, overuse tools, or operate without adequate monitoring. Deployment in high-stakes domains requires task-specific safety evaluation and human oversight.

REPRODUCIBILITY STATEMENT

Sections 3–5 define the execution graph, credit estimators, routing rules, datasets, and metrics. Appendix D gives the role prompt templates and training details. Appendix E gives the experimental protocol, rollout-statistic definitions, and inference-budget results. Appendix F gives training pseudocode. The supplementary materials contain anonymized source code, training settings, and the complete paired case-study records, together with a script that verifies the reported behavior.

REFERENCES

Yanjun Chen, Yirong Sun, Hanlin Wang, Jinghan Wang, Xinming Zhang, Xiaoyu Shen, Wenjie Li, and Wei Zhang. Exact is easier: Credit assignment for cooperative LLM agents, 2026. URL https://arxiv.org/abs/2603.06859v2.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training, 2025. URL https://arxiv.org/abs/2505.10978.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *International Conference on Computational Linguistics*, 2020. URL https://aclanthology.org/2020.coling-main.580/.

Haoyang Hong, Jiajun Yin, Yuan Wang, Jingnan Liu, Zhe Chen, Ailing Yu, Ji Li, Zhiling Ye, Hansong Xiao, Yefei Chen, Hualei Zhou, Yun Yue, Minghui Yang, Chunxiao Guo, Junwei Liu, Peng Wei, and Jinjie Gu. Multi-agent deep research: Training multi-agent systems with M-GRPO, 2025. URL https://arxiv.org/abs/2511.13288.

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 *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2017. URL https://aclanthology.org/P17-1147/.

Kimi Team. Kimi K2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the*

Association for Computational Linguistics, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230-Abstract.html.

Junbo Li, Peng Zhou, Rui Meng, Meet P. Vadera, Lihong Li, and Yang Li. Turn-PPO: Turn-level advantage estimation with PPO for improved multi-turn RL in agentic LLMs, 2025a. URL https://arxiv.org/abs/2512.17008.

Shiyu Li, Yifan Wang, Peiming Li, Zheng Wei, and Yang Tang. ReSeek: A self-correcting framework for search agents with instructive rewards. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://tencentbac.github.io/ReSeek/.

Wenwu Li, Yuran Song, Mingze Zhao, Bo Jin, and Wenhao Li. Unifying temporal and structural credit assignment in LLM-based multi-agent prompt optimization, 2026b. URL https://arxiv.org/abs/2605.30227v1.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models, 2025b. URL https://arxiv.org/abs/2501.05366.

Yanming Li, Xuelin Zhang, Wenjie Lu, Ziye Tang, Maodong Wu, Haotian Luo, Tongtong Wu, Zijie Peng, Hongze Mi, Yibo Feng, Naiqiang Tan, Chao Huang, Lian Peng, and Li Shen. Who deserves the reward? SHARP: Shapley credit-based optimization for multi-agent system, 2026c. URL https://arxiv.org/abs/2602.08335.

Zhongyi Li, Wan Tian, Jinju Chen, Huiming Zhang, Yang Liu, Yikun Ban, and Fuzhen Zhuang. Counterfactual credit policy optimization for multi-agent collaboration, 2026d. URL https://arxiv.org/abs/2603.21563v5.

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. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://aclanthology.org/2023.acl-long.546/.

MiniMax. MiniMax-M1: Scaling test-time compute efficiently with lightning attention, 2025. URL https://arxiv.org/abs/2506.13585.

Zhanfeng Mo, Xingxuan Li, Yuntao Chen, and Lidong Bing. Multi-agent tool-integrated policy optimization, 2025. URL https://arxiv.org/abs/2510.04678.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023. URL https://aclanthology.org/2023.findings-emnlp.378/.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Leitian Tao, Baolin Peng, Wenlin Yao, Tao Ge, Hao Cheng, Mike Hang Wang, Jianfeng Gao, and Sharon Li. TRACE: Turn-level reward assignment via credit estimation for long-horizon agents, 2026. URL https://arxiv.org/abs/2607.13988v1.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 10014–10037, 2023. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Hongcheng Wang, Yinuo Huang, Sukai Wang, Guanghui Ren, and Hao Dong. Why tree-style branching matters for thought advantage estimation in GRPO. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://arxiv.org/abs/2509.24494.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. WideSeek-R1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning, 2026. URL https://arxiv.org/abs/2602.04634.

An Yang et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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. In *Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://aclanthology.org/D18-1259/.

Wenzhen Yuan, Wutao Xiong, Fanchen Yu, Shengji Tang, Ting Liu, Tao Chen, Peng Ye, Yuzhuo Fu, Wanli Ouyang, and Lei Bai. Small model as master orchestrator: Learning unified agent-tool orchestration with parallel subtask decomposition, 2026. URL https://arxiv.org/abs/2604.17009.

Zhiwei Zhang, Xiaomin Li, Yudi Lin, Hui Liu, Ramraj Chandradevan, Linlin Wu, Minhua Lin, Fali Wang, Xianfeng Tang, Qi He, and Suhang Wang. Unlocking the power of multi-agent LLM for reasoning: From lazy agents to deliberation, 2025. URL https://arxiv.org/abs/2511.02303v1.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

A Contents and Supplementary Materials

- **B: Extended Related Work.**
  - B.1: Coordination and Shared-Policy Optimization.
  - B.2: Temporal and Counterfactual Evaluations.
  - B.3: Credit Assignment for Prompt Optimization.
- **C: Illustrative Multi-Round Execution.** A rollout with round-level task names, condensed returns, and controller summaries.
- **D: Execution and Training Details.**
  - D.1: Training Hyperparameters.
  - D.2: Prompt Fields and Retrieval History.
  - D.3: Role Prompt Templates.
  - D.4: Deletion Sampling.
  - D.5: Advantages and Token Masks.
  - D.6: One-Round Retrospective Decision Supervision.
  - D.7: Optimization Details.
- **E: Experimental Details and Budget Ablations.**
  - E.1: Training and Evaluation Setup.
  - E.2: Rollout Analysis Protocol.
  - E.3: Comparator Implementations.
  - E.4: Inference Subagent-Limit Ablation.
  - E.5: Inference Round-Limit Ablation.
- **F: Execution and Training Pseudocode.**
- **Separate supplementary package.**
  - `case_study/`: Early and near-end checkpoint rollout records, guide, and verifier.
  - `code/`: Anonymized implementation, local dependencies, setup guide, and tests.
  - `configurations/`: Resolved training settings and recorded package versions.
  - Package root: Contents guide and validation record.

B Extended Related Work

B.1 Coordination and Shared-Policy Optimization

PARL learns dynamic parallel orchestration while keeping subagents frozen (Kimi Team, 2026). ParaManager learns state-aware parallel decomposition and delegation over a unified agent–tool action space (Yuan et al., 2026). These approaches study how to select and coordinate delegated work. MAMR-Framework likewise generates assignments during execution, while using a single trainable LLM across assignment, execution, and aggregation roles.

MATPO extends GRPO to jointly optimize planner and worker trajectories with a shared LLM (Mo et al., 2025). M-GRPO additionally addresses trajectory alignment under variable numbers of subagent executions and uses hierarchical group-relative training (Hong et al., 2025). These works provide related approaches to coordinating policy updates across agent roles. Our focus is on the local evaluation targets attached to generated segments: task-assignment progress, individual and joint output-removal effects, and subsequent progress conditioned on summaries.

B.2 Temporal and Counterfactual Evaluations

Turn-PPO formulates interaction as a turn-level MDP and uses a learned critic for turn-level advantage estimation (Li et al., 2025a). MAMR-GRPO estimates round values from forced-stop answer rewards. It evaluates complete rounds comprising assignment, parallel execution, and aggregation.

Counterfactual methods also differ in what they intervene on and which downstream quantities they evaluate. In CCPO's sequential Think–Solve setting, Thinker credit compares collaborative outcomes

with independently sampled Solver-only outcomes (Li et al., 2026d). Dr. MAMR groups semantically similar steps and combines masking-induced changes in next-step likelihood with outcome and restart feedback (Zhang et al., 2025). C3 fixes the pre-decision history, samples alternative actions under a frozen behavior policy, and compares their downstream terminal returns using a leave-one-out baseline (Chen et al., 2026).

MAMR-GRPO instead removes realized parallel outputs while holding the surviving outputs fixed, then regenerates the downstream summary prefix and forced-stop answer. Its removal effects therefore evaluate the dependence of the current round value on selected outputs, conditional on the surviving outputs. This differs from comparing alternative actions and their complete continuations, or measuring changes in the likelihood of an already sampled next step. Joint removals additionally evaluate output sets whose usefulness may not be apparent from individual removals alone.

B.3 Credit Assignment for Prompt Optimization

Temporal and structural credit have also been combined for multi-agent prompt optimization (Li et al., 2026b). This framework uses an explicit aggregated state between rounds and LLM critics to provide auxiliary feedback for alternating updates of role and aggregation prompts, while keeping the base model fixed. MAMR-Framework shares the use of an aggregated cross-round state. The distinction lies in the feedback and optimization variables: MAMR-GRPO derives local credits from generated-answer rewards and uses them to update the shared LLM policy parameters.

C Illustrative Multi-Round Execution

This HotpotQA rollout (`hotpotqa::dev_4512`) asks for the premiere channel of the show featuring Gerald Bald Z. It returns Cartoon Network (EM 100%) after three rounds and 12 Sub Agent executions. Tab. 4 and Tab. 5 retain all generated task names and condense model returns and Controller Agent summaries h_t; they omit prompts and retrieval records.

Table 4: Round 1 task names, condensed returns, and Controller Agent state.

ID	Generated task name	Returned answer (condensed)
Round 1: identify the show and candidate outlet		
1.1	Identify Show Featuring Gerald Bald Z	Perfect Hair Forever.
1.2	Find Premiere Channel of a Known Show	Adult Swim.
1.3	Verify if Gerald Bald Z is a Fictional or Satirical Name	Fictional.
1.4	Cross-Reference Media Outlets for Show Details	Perfect Hair Forever; Adult Swim on Cartoon Network.
Controller Agent after Round 1 (h_1). Candidate: *Perfect Hair Forever* (HIGH); channel unresolved. Decision: `continue` to check channel versus block.		

Table 5: Rounds 2 and 3 task names, condensed returns, and CONTROLLER AGENT states. Regional-premiere and production-partner claims are unverified model returns.

ID	Generated task name	Returned answer (condensed)
Round 2: distinguish channel from programming block		
2.1	Verify Premiere Channel	Cartoon Network.
2.2	Check Official Sources	Adult Swim.
2.3	Clarify Channel vs Block	Cartoon Network is the broadcast channel; Adult Swim is its late-night programming block.
2.4	Explore International Premieres	Not found.
CONTROLLER AGENT after Round 2 (h_2). Candidate: Cartoon Network (HIGH); exact premiere slot unconfirmed. The summary requests a source explicitly naming the channel or block. Decision: `continue`.		
Round 3: check the final channel interpretation		
3.1	Verify Official Premiere Source	Cartoon Network's Adult Swim block.
3.2	Clarify Channel vs. Block	Adult Swim is a programming block on Cartoon Network.
3.3	Check for Regional Variants	Claims premieres on other channels abroad; identifies Cartoon Network for the U.S. premiere.
3.4	Cross-Verify with Production Details	Names Cartoon Network (Adult Swim) as the original production and distribution partner.
CONTROLLER AGENT after Round 3 (h_3). Candidate: Cartoon Network (HIGH). The summary still asks for formal source wording, but the boundary decision is `stop`.		

In Round 1, the CONTROLLER AGENT candidate is the show title, although the question asks for a channel. Summary h_1 directs Round 2 toward the Adult Swim–Cartoon Network distinction; h_2 directs Round 3 toward source wording. After the Round 3 `stop` decision, a separate terminal-answer generation emits `<response>Cartoon Network</response>`. The trace shows summary-conditioned task allocation and stopping.

D EXECUTION AND TRAINING DETAILS

This appendix gives the implementation details for the local credits in Section 4.

D.1 TRAINING HYPERPARAMETERS

Tab. 6 lists the MAMR-GRPO training hyperparameters. We optimize all parameters of one shared policy, using batches of 64 questions and sampling eight rollouts per question.

Table 6: Training hyperparameters for MAMR-GRPO. The format-weight order is task assignment, subagent execution, information aggregation, and final answering.

Hyperparameter	Value
Model and optimization	
Shared policy	Qwen3-4B-Instruct-2507
Hardware / training precision	$8\times$ A100 80GB / bfloat16
Parameter updates / sharding	Full parameter / FSDP2, size 8
Training examples / epochs / steps	49,938 / 1 / 780
Optimizer / learning rate	AdamW / 10^{-6}
Adam momentum (β_1, β_2) / weight decay	$(0.9, 0.999)$ / 0.01
Learning-rate schedule / warmup	Constant after 200 warmup steps
Gradient norm clipping	1.0
Policy update / ratio clipping	CISPO (MiniMax, 2025) / $[0.8, 1.2]$
Reference regularizer / coefficient	Squared log-ratio / 0.001
Entropy coefficient / loss reduction	0 / token mean
Generation batch / rollouts per question K	64 questions / 8
Configured data batch / actor minibatch	32 / 96
Actor epochs per batch	1
Microbatch per GPU / accumulation setting	1 / 2
Dynamic batch token limit per GPU	50,000
Training seed / checkpoint interval	42 / 50 steps
Rollout and retrieval	
Temperature / top-p / top-k sampling	1.0 / 1.0 / disabled
Maximum rounds / subagents per round	3 / 4
Prompt / response token limit	16,384 / 8,192
Configured training context limit	24,576 tokens
Subagent assistant-turn limit	8
Retrieval corpus / encoder / top-k results	Wiki2018 / e5-base-v2 / 3
Prior document / query limit	8 / 6
Prior-document snippet limit	150 characters
Tool-response limit / truncation	4,096 characters / middle
Rollout workers / tensor parallel size	64 / 1
Credit assignment and auxiliary feedback	
Answer samples per value estimate M	4
Deletion evaluations per round F	8
Terminal weight λ_g	1.0
Role weights $(\lambda_{\text{meta}}, \lambda_{\text{sub}}, \lambda_{\text{ctrl}})$	$(1.0, 1.0, 1.0)$
Future-gain discount γ	1.0
Decision margin ϵ_{act} / weight cap $w_{\max}$	0 / 0.1
Shaping weights $(\lambda_{\text{nov}}, \lambda_{\text{rep}}, \lambda_{\text{empty}})$	$(0.05, 0.10, 0.05)$
Answer-advantage ratio filter	$[0.8, 1.2]$
Terminal format-feedback scale	0.5
Format weights	$(0.3, 0.1, 0.3, 0.3)$

D.2 Prompt Fields and Retrieval History

Each generated task specification contains free-form `name`, `goal`, `context`, and `must_find` fields. Appendix C shows generated task names, condensed returns, and CONTROLLER AGENT state changes from one rollout. The retrieval history supplies prior queries and document metadata for redundancy control. When building a subagent prompt for round t, the implementation includes only records from rounds strictly earlier than t. Updates from other subagents in round t do not enter that prompt. Earlier reasoning and evidence text reach subsequent rounds through the summary.

The CONTROLLER AGENT prefix p_t includes any generated reasoning before the summary, the summary itself, and the opening decision tag. The decision value and closing tag follow this prefix. The answer interface retains both the CONTROLLER AGENT input c_t and the generated prefix p_t. For a deletion evaluation, it uses the reduced context and a newly generated prefix.

D.3 ROLE PROMPT TEMPLATES

The following templates give the complete Search-QA role instructions used during training. The META AGENT and CONTROLLER AGENT receive separate system and user messages. The SUB AGENT receives a system and user message assembled from the META AGENT's task fields. Braces below denote runtime content, not text shown to the model. We show the training setting with $S = 4$, which renders the META AGENT's task range as "2–4", and with the prior-retrieval rule enabled. At inference, changing S changes that range. The previous summary is `EMPTY` in the first round; `round_idx` is one-based. In the supplementary code, `mamr_grpo/prompts/search_qa/` contains the templates, and `mamr_grpo/tasks/search_qa.py` handles field substitution and assembles the system and user messages for each subagent task.

META AGENT. The system message specifies the task and JSON fields. *System message*

```
You are a task planner for a multi-round search workflow.

Your job is to decompose the main question into 2-4 parallel subtasks that together best
     improve the final answer.

Rules:
1. Read the previous summary first. Reuse HIGH-confidence facts. Treat MEDIUM/LOW items as
     unverified hypotheses.
2. Create 2-4 independent subtasks, each targeting a different angle.
3. No overlap: do not send two agents after the same missing fact.
4. No repetition: do not repeat searches already covered in the summary.
5. Each subtask must be self-contained --- embed all needed context in <context> and
     <must_find>.
6. Assign only tasks solvable with the search tool and public web evidence.
7. Cross-round diversity: each subtask must explore an angle distinct from those covered in
     prior rounds --- different entity, attribute, time period, source type, or alternative
     hypothesis. Reformulating earlier queries with synonyms or near-duplicate phrasing is
     forbidden; the new angle must change WHAT is being asked, not just HOW it is phrased.

## Output format (STRICT)

<plan>
What is known, what is missing, and why these subtasks help.
</plan>

<subtasks>
JSON list of subtask objects (2 to 4).
Each object must contain:
- "name": short task identifier (2-5 words)
- "goal": one-sentence task objective
- "context": minimal background the sub-agent needs (include relevant known facts)
- "must_find": bullet list of 2-4 concrete items to search for
</subtasks>
```

User message

```
Task:
{question}

Previous summary:
{previous_summary}

Round:
{round_idx}

Propose 2-4 subtasks that best address what is still missing.
```

Sub Agent. For a structured task, the parser uses the generated `goal`, `context`, and `must_find` fields. The `Background` line is omitted if context is empty, and the `You must find` lines are omitted if `must_find` is empty. For legacy task fields, the generated system and user task text replaces the corresponding goal-based text; the fixed instructions below still apply. The system message begins as follows. *System message*

```
You are a search agent. Your task: {goal}
Background: {context}
When you need outside evidence, call the native 'search' tool with a 'query_list' argument
     containing one or more fully formed search queries. Return a concise evidence-backed
     note for the controller.
```

If prior retrieval records exist, the system message then appends a blank line, a line containing ---, the retrieval-history block, and another --- line. The block lists up to eight earlier document titles and six earlier queries, without current-round records. The form with retrieved documents is:

```
Documents already retrieved in previous rounds:
- [{title}]  [repeated for retained documents]

Previously attempted queries:
- {query}  [repeated for retained queries]

Search for evidence not already covered above. Avoid repeating the same documents or facts.
```

The query heading and lines are omitted when no earlier queries exist. When only earlier queries exist, the document lines are absent and the closing sentence is: "The above queries returned no useful documents. Try genuinely different queries or angles, not paraphrases." No retrieval-history block is appended in the first round or when earlier retrievals supplied neither queries nor documents.

The SUB AGENT's user message consists of the generated task request followed by the fixed answering and output instructions below. *User message*

```
{goal}

You must find:
{must_find}
## Answering policy (IMPORTANT)

Before writing "not found", check whether the answer can be **directly inferred in one step**
     from facts already retrieved. Examples of allowed one-step inferences:
- "X is the son of Y" + "Y is the son of Z" ⇒ paternal grandfather of X is Z
- "Film F was released in 1925" ⇒ F came out in the 1920s
- "Person P was born in city C, which is in country K" ⇒ P was born in K
- "Film F was directed by D" + "D is American" ⇒ F's director is American

If such an inference is available, give the inferred answer. Only answer "not found" when
     **no retrieved fact** supports any reasonable inference.

When multiple retrieved candidates are plausible but none is confirmed, pick the
     best-supported one and signal uncertainty in <key_evidence> --- do NOT default to "not
     found".

## Output requirement (MANDATORY)
After your last tool call, your final assistant message must be exactly one XML block --- no
     text before or after it:

<summary>
<answer>Direct answer to the assigned task (may be a one-step inference from retrieved
     facts), or "not found" only if no evidence supports any answer.</answer>
<key_evidence>
- fact or quote from search results
- fact or quote from search results
</key_evidence>
<open_questions>
- unresolved point or missing fact, or "none"
</open_questions>
</summary>
```

CONTROLLER AGENT. The CONTROLLER AGENT receives the original question, the prior summary, and the current round's SUB AGENT outputs. The latter are joined in order as `[sub_0]`, `[sub_1]`, and so on; `NONE` replaces an empty list. *System message*

```
You are the Controller in a multi-round search workflow.

Your responsibilities:
1. Read the task, the previous summary, and the sub-agent outputs from this round.
2. Write a <summary> that consolidates everything learned so far.
3. Decide whether to continue or stop, and if stopping, produce the final answer.

## Summary

The <summary> is the **sole memory** carried between rounds. Treat the previous summary as a
     reference --- preserve what the new evidence confirms, correct what it contradicts, and
     discard what turns out to be irrelevant. Structure it as follows:

<summary>
<best_answer>Current best answer.</best_answer>
<best_answer_confidence>HIGH|MEDIUM|LOW|NONE</best_answer_confidence>
<key_evidence>2-4 short bullets with the strongest supporting facts.</key_evidence>
<open_questions>Unresolved points or alternative hypotheses that could change the final
     answer, if any.</open_questions>
<dead_ends>Searches or leads that were tried and found unhelpful --- so the next round does
     not repeat them.</dead_ends>
<next_step>The single most useful missing fact to search next, or "none".</next_step>
</summary>

## Decision

- **stop**: the current answer is supported well enough, and remaining uncertainty is
     unlikely to change it. Also stop if no concrete new search direction exists.
- **continue**: one more round could realistically find a missing fact that may change the
     answer. Do not continue just to collect more confirming evidence.

## Output format (STRICT)

<summary>
...
</summary>

<decision>continue or stop</decision>

If and only if decision is "stop", append:

<response>
answer
</response>

Rules:
- <summary> and <decision> are mandatory in every output.
- Only include <response> when decision is "stop".
- <response> must contain ONLY the direct answer --- no explanation, no reasoning, no full
     sentences. Just the answer entity itself.

Examples of correct <response> format:
- <response>Beijing</response>
- <response>42</response>
- <response>William Shakespeare</response>

## Final answer rules (STRICT --- avoid refusal answers)

When producing <response> you MUST commit to a concrete entity / value. Forbidden final
     answers include:
"None", "Unknown", "N/A", "Not found", "Cannot determine", "Insufficient information",
"No valid answer", and any similar refusal phrase.

Rules for constructing the final answer:
1. **Aggregate across sub-agents and rounds.** If ANY sub-agent's <answer> is a concrete
     entity (not "not found"), treat that as a usable candidate --- do not discard it just
     because other sub-agents failed.
2. **Combine partial evidence with one-step inference.** Facts scattered across sub-agents
     count: if one sub gives "X is son of Y" and another gives "Y is son of Z", the
     grandfather question is answerable. If one sub confirms one side of a comparison and the
     other side is missing, still commit to the confirmed candidate.
3. **When no sub-agent answer is available at all**, pick the best-supported entity from
     <key_evidence> of the current or previous summary. Mark
     <best_answer_confidence>LOW</best_answer_confidence> but still output a concrete guess
     in <response>.
```

```
4. Only if <key_evidence> contains no candidate entity whatsoever may you output a short
     concrete guess based on the question form (e.g. a year for "when", a person name for
     "who") --- still never a refusal phrase.

<best_answer_confidence>LOW</best_answer_confidence> is acceptable and expected for hard
     questions. A low-confidence guess scores better than a refusal.
```

User message

```
Task:
{question}

Previous summary (use as reference --- confirm, correct, or discard based on new evidence
     this round):
{previous_summary}

Round:
{round_idx}

Sub-agent outputs (new evidence from this round):
{subagent_outputs}
```

D.4 Deletion Sampling

For a budget of F deletion evaluations, the Search-QA implementation shuffles subagent indices and first deletes each of the first $\min(F, n_t)$ subagents individually. Each remaining evaluation draws a deletion size uniformly from $1, \ldots, n_t - 1$, then a subset uniformly at that size. Sampling uses subagent indices and independent random draws, without inspecting output text or scores. With $F \geq n_t$, every subagent is selected at least once. Our configuration uses $F = 8$ and $M = 4$. Proper subsets retain at least one subagent record: $n_t = 1$ has no local deletion credit, $n_t = 2$ permits only individual deletion, and joint deletion requires at least three subagents in the same round.

D.5 Advantages and Token Masks

For role $r \in \{\mathrm{meta}, \mathrm{sub}, \mathrm{ctrl}\}$, the segment advantage is

$$A^{(r)} = \lambda_g A^{\mathrm{group}} + \lambda_r C^{(r)},$$

where λ_g and λ_r set the relative weights. The shared anti-redundancy shaping adds an adjustment $B_{t,j}$ to subagent credit before applying λ_{sub}; it does not enter R_i^{term} or the value estimates. For a subagent in round $t > 1$, let $\rho_{t,j}$ be the fraction of its returned document identities retrieved in earlier rounds. With nonnegative weights λ_{nov}, λ_{rep}, and λ_{empty}, the adjustment is $\lambda_{\mathrm{nov}}(1-\rho_{t,j}) - \lambda_{\mathrm{rep}}\rho_{t,j}$ when documents are returned, and $-\lambda_{\mathrm{empty}}$ when no search is made or no documents are returned. It is zero in the first round and for tool-service errors. Tab. 7 specifies which tokens receive these advantages and which use separate supervision.

Table 7: Training signals by generated segment. Prompt tokens, tool responses, and inserted formatting tokens have zero policy-loss mask.

Generated segment	Training signal
Task assignment	$\lambda_g A^{\mathrm{group}} + \lambda_{\mathrm{meta}} C_t^{\mathrm{meta}}$ on generated task-assignment tokens.
Sub Agent execution	$\lambda_g A^{\mathrm{group}} + \lambda_{\mathrm{sub}}(C_{t,j}^{\mathrm{sub}} + B_{t,j})$ on all model-generated execution tokens.
Controller Agent prefix p_t	$\lambda_g A^{\mathrm{group}} + \lambda_{\mathrm{ctrl}} C_t^{\mathrm{ctrl}}$ on the summary, preceding generated reasoning, and opening decision tag.
Decision value and close tag	Weighted target-token cross-entropy; no policy-gradient loss or reference-policy regularization.
Terminal answer samples	Answer-level leave-one-out advantages from individual answer scores for the same input; no local role credit.
Evaluation generations	Credit estimation only: forced-stop answers at nonterminal boundaries and all generations in deletion branches.

Only answer samples at the terminal boundary of an original rollout participate in optimization, whether the rollout ends by a sampled stop or the round limit. Their advantages compare each

sample's individual score with the mean of the other answer samples for the same input. The trajectory-level reference for role segments uses terminal boundary means. Neither reference divides by a group standard deviation. Answer tokens receive zero policy advantage when the old-actor-to-collection probability ratio is outside $[0.8, 1.2]$; their reference-policy regularization mask remains active. The GRPO-style baseline uses the same unnormalized trajectory advantage and policy-update settings, and its answer samples inherit their parent trajectory's advantage. Both methods retain anti-redundancy shaping and boundary-action supervision.

D.6 One-Round Retrospective Decision Supervision

For a sampled continuation with an observed next round, define $g_t = \widehat{V}_{t+1} - \widehat{V}_t - \epsilon_{\text{act}}$. The target is `continue` if $g_t > 0$ and `stop` otherwise, with weight $w_t = \min(|g_t|, w_{\max})$. The target tokens include the decision value and closing tag, conditioned on the retained prefix p_t. Sampled stops supply no observed continuation outcome and receive no target. Our configuration uses $\epsilon_{\text{act}} = 0$ and $w_{\max} = 0.1$; a continuation truncated at the round limit also has zero supervision weight. Invalid or aborted traces are excluded. Each target evaluates the observed gain from one additional round.

D.7 Optimization Details

The policy loss and reference-policy regularizer are averaged over generated tokens with active policy-loss masks. The reference term is a squared log-ratio penalty with coefficient 0.001. Boundary decisions use weighted target-token cross-entropy with the targets and weight cap defined above. The update settings are listed in Tab. 6.

E Experimental Details and Budget Ablations

E.1 Training and Evaluation Setup

We initialize MAMR-GRPO with Qwen3-4B-Instruct-2507 and train on 49,938 examples: 15,000 from 2WikiMultiHopQA, 10,000 from HotpotQA, 19,938 from MuSiQue, and 5,000 from Natural Questions. We sample eight trajectories per input at temperature 1.0 and set the reference-policy regularization coefficient to 0.001. Training permits up to three collaboration rounds and four subagents per round. Retrieval uses e5-base-v2 over Wiki2018 with top-k=3 results. MAMR-GRPO uses role-specific credits and terminal answer-level advantages (Section 4). Nonterminal forced-stop samples supply credit estimates only. GRPO and MAMR-GRPO share execution and answering protocols; MAMR-GRPO estimates local credits with counterfactual evaluations.

Evaluation uses one answer per example on 51,713 examples from 2WikiMultiHopQA, Bamboogle, HotpotQA, MuSiQue, Natural Questions, PopQA, and TriviaQA. We report normalized EM by benchmark; *Average* is always the unweighted mean of the seven benchmark-level scores. Candidate checkpoints were screened on a 20% subset of the evaluation collection before evaluation on the full collection. Section 5 describes the published baselines, local reproductions, and comparisons on the shared execution graph. The supplementary materials provide the implementation, training configuration, and complete paired case-study transcripts.

E.2 Rollout Analysis Protocol

For the controlled GRPO–MAMR-GRPO full-collection evaluation, a stop decision is followed by a final answer conditioned on the Controller Agent context and generated prefix, including its summary. We generate one final answer per question at temperature 0.7 for the full-collection evaluation and at temperature 1.0 for the paired checkpoint comparison.

For valid rollouts, rounds and total Sub Agent executions are averaged per rollout; executions per realized round equal total subagent executions divided by total realized rounds; the continue rate equals continue decisions divided by realized Controller Agent rounds; and final-answer reward is mean token F1. This rollout diagnostic differs from the reported benchmark EM. We apply these definitions to the early (step 50) and near-end (step 600) checkpoints from the same MAMR-GRPO training run, evaluated on the same 2,086 test questions under identical limits. The cross-method statistics aggregate MAMR-GRPO and GRPO rollouts collected during training steps 515–780. We

compare rounds, subagent executions, and continue rates between the two training runs. The paired checkpoint evaluation also measures completion tokens and tool calls per rollout.

The supplementary materials include complete transcripts for the `hotpotqa::dev_3799` case in Section 5.5: `case_study/early_checkpoint.json` (step 50) and `case_study/near_end_checkpoint.json` (step 600). Both records use MAMR-GRPO. Their transcripts contain the task assignments, subagent outputs, CONTROLLER AGENT summaries, and boundary decisions described in the main text. The supplied `case_study/verify_case.py` verifies the answers, execution counts, and follow-up task assignments in the paired example.

E.3 COMPARATOR IMPLEMENTATIONS

For C3 SubMessage Lite (Chen et al., 2026), we retain the shared MAMR-Framework execution and training pipeline while replacing the SUB AGENT random-subset removal credit with a C3-style SubMessage leave-one-out credit. Alternative SUB AGENT messages are evaluated through the downstream CONTROLLER AGENT-to-answer path rather than full multi-round continuation.

For SHARP-MC (Li et al., 2026c), we retain execution and replace SUB AGENT credit with a Monte Carlo approximation to Shapley-style marginal contributions over SUB AGENT permutations.

E.4 INFERENCE SUBAGENT-LIMIT ABLATION

We evaluate the near-end checkpoint with subagent caps $S = 2, \ldots, 6$ and a fixed round cap $R = 3$. Each setting uses the same 51,713 questions, temperature 0.7, top-p 1.0, and forced final-answer generation. Retrieval uses Wiki2018 and top-3 results. The budget evaluations use one sampling run per setting without a fixed generation seed. Each subagent has limits of eight assistant turns, four tool calls, and 4,096 generated tokens; the META AGENT and CONTROLLER AGENT each have a 2,048-token limit. Changing S changes both the task-count range in the META AGENT prompt and the enforced subagent limit.

Completion tokens sum the model-generated output from the META AGENT, SUB AGENT executions, CONTROLLER AGENT, and final-answer call, then average over questions. This metric measures output generation; prompt and retrieved-document tokens are excluded from the count.

Table 8: Subagent-limit comparison ($R = 3$). Average EM is the unweighted seven-benchmark mean. Subagent executions and completion tokens are averaged per question.

Subagent limit S	Average EM (%)	Subagent executions	Completion tokens
2	45.43	3.70	2,496
3	46.37	5.22	3,119
4	46.71	6.72	3,735
5	46.81	8.26	4,393
6	47.76	9.94	5,136

Table 9: Per-benchmark EM (%) by subagent cap at $R = 3$. Average weights the seven benchmarks equally. 2Wiki and NQ denote 2WikiMultiHopQA and Natural Questions.

Benchmark	$S = 2$	$S = 3$	$S = 4$	$S = 5$	$S = 6$
2Wiki	54.13	55.42	55.23	55.71	56.20
Bamboogle	48.80	50.40	49.60	52.00	56.80
HotpotQA	44.62	46.04	46.14	46.20	46.51
MuSiQue	21.72	22.18	22.42	21.47	22.13
NQ	39.50	40.36	40.86	40.91	41.16
PopQA	44.81	44.80	46.65	45.68	45.48
TriviaQA	64.46	65.37	66.08	65.69	66.07
Average	45.43	46.37	46.71	46.81	47.76

The per-benchmark results identify the contributions to the increase in Average EM. From $S = 4$ to $S = 6$, Bamboogle improves by 7.20 points, contributing 1.03 points to the 1.05-point increase in Average EM. The other six benchmarks make a small net contribution: gains on 2Wiki, HotpotQA, and NQ are offset by lower MuSiQue and PopQA scores and an almost unchanged TriviaQA score. Increasing the subagent cap beyond four thus mainly benefits Bamboogle. Total executions rise from 6.72 to 9.94 and completion tokens from 3,735 to 5,136 per question. The extra generation yields little net improvement across the six remaining benchmarks.

E.5 Inference Round-Limit Ablation

We evaluate the same near-end checkpoint with round caps $R = 1, \ldots, 5$ and a fixed subagent cap $S = 4$. The question collection, retrieval system, sampling settings, and per-agent limits are the same as in the subagent-limit ablation. Caps $R = 4, 5$ allow collaboration to continue beyond the three-round training limit. We report EM and average realized rounds, subagent executions, and completion tokens per question.

Table 10: Round-limit comparison ($S = 4$). Average EM is the unweighted seven-benchmark mean. Realized rounds, subagent executions, and completion tokens are averaged per question.

Round limit R	Average EM (%)	Realized rounds	Subagent executions	Completion tokens
1	44.07	1.000	3.995	2,058
2	46.21	1.449	5.788	3,143
3	46.71	1.681	6.716	3,735
4	47.32	1.839	7.349	4,141
5	47.61	1.961	7.836	4,457

Table 11: Per-benchmark EM (%) by round cap ($S = 4$). Average is the unweighted mean.

Benchmark	$R = 1$	$R = 2$	$R = 3$	$R = 4$	$R = 5$
2Wiki	49.89	55.30	55.23	55.72	55.97
Bamboogle	49.60	49.60	49.60	55.20	56.00
HotpotQA	42.27	45.24	46.14	45.77	46.10
MuSiQue	17.25	22.09	22.42	22.63	22.55
NQ	40.42	41.00	40.86	41.08	41.86
PopQA	44.45	44.91	46.65	45.45	45.20
TriviaQA	64.58	65.33	66.08	65.38	65.59
Average	44.07	46.21	46.71	47.32	47.61

Different benchmarks benefit at different round limits. The first extra round raises 2Wiki by 5.41 points, HotpotQA by 2.97, and MuSiQue by 4.84; these three gains contribute 1.89 points of the 2.14-point increase in Average EM. Their scores then change much less as R grows from 2 to 5. Bamboogle follows a different pattern: it remains at 49.60 through $R = 3$ and rises to 55.20 at $R = 4$. The 5.60-point increase contributes 0.80 points to the seven-benchmark average, exceeding the net 0.61-point gain because some other benchmarks decline. Additional interaction therefore has a benchmark-dependent useful depth. Most gains on 2Wiki, HotpotQA, and MuSiQue occur after one additional round; Bamboogle benefits from search beyond the training limit.

F EXECUTION AND TRAINING PSEUDOCODE

The implementation processes each training input x as follows:

1. Sample MAMR-Framework rollouts at fixed π_θ; compute terminal group advantages.
2. At each round, fix its history and sample M forced-stop answers for $\widehat{V}_t$.
3. For `continue` decisions, sum discounted adjacent round-value gains.
4. Evaluate deletions first for individual subagents, then for nonempty proper subsets. Keep surviving records fixed, regenerate CONTROLLER AGENT prefixes and forced-stop answers, and compute Eq. 9.
5. Assign advantages to generated role segments, separate targets to eligible boundary decisions, and leave-one-out advantages to original terminal answers. Update π_θ with the policy loss, reference-policy regularization, and boundary-decision supervision.