End-to-End Optimization of LLM-Driven Multi-Agent Search Systems via Heterogeneous-Group-Based Reinforcement Learning

Anonymous ACL submission

Abstract

Large language models (LLMs) are versatile, yet their deployment in complex real-world settings is limited by static knowledge cutoffs and the difficulty of producing controllable behavior within a single inference. Multi-agent search systems (MASS), which coordinate specialized LLM agents equipped with search tools, mitigate these issues via task decomposition and retrieval-augmented problem solving. However, optimizing LLMs for agent-specific roles remains labor-intensive with prompt engineering or supervised fine-tuning, motivating automated end-to-end training. Existing MARL methods such as Multi-Agent Proximal Policy Optimization (MAPPO) typically depend on large critic networks to evaluate joint actions, leading to instability and high memory cost. We introduce Multi-Agent Heterogeneous Group Policy Optimization (MHGPO), which updates policies by estimating relative advantages across heterogeneous groups of multi-agent rollouts. We further study three group rollout sampling strategies to trade off sample efficiency and optimization quality. Experiments show that MHGPO captures implicit inter-agent dependencies and consistently outperforms strong baselines in both task performance and computational efficiency.

1 Introduction

Large language models (LLMs) have demonstrated strong performance across a wide range of domains. Despite these advances, deploying LLMs in real-world industrial settings remains challenging. On the one hand, the fixed knowledge cutoff of LLMs (Gao et al., 2024) limits their ability to reason over previously unseen information. On the other hand, although advanced reasoning models such as DeepSeek-R1 (Shao et al., 2024) have achieved substantial progress on complex tasks, LLMs still struggle to produce controllable and consistently high-accuracy outputs within a single inference.

As a result, the practical deployment of LLMs in specialized scenarios increasingly relies on combining *agent-based* frameworks with retrieval-augmented generation (RAG), where each LLM is assigned a well-defined role, equipped with *search tools*, and governed by predefined communication protocols, thereby serving as a reasoning-capable task component with access to task-specific external information. In contrast to methods that augment a single strong reasoning model by integrating retrieval into *single-context* reasoning via tool calls (Li et al., 2025; Song et al., 2025), a widely adopted alternative for complex problem solving is to organize a team of specialized LLM agents as an multi-agent system (MAS) that decomposes high-level objectives into fine-grained, modular, *multi-context* subtasks; each subtask is solved within its own context and coordinated across agents via structured communication and LLM-driven reasoning, thereby reducing both the cognitive burden and per-model capability requirements while improving overall system interpretability and controllability.

Numerous studies have explored multi-agent search systems (MASSs)—the integration of MAS with RAG—for complex functionalities (Chang et al., 2025; Chen et al., 2025; Kulkarni et al., 2024; Wu et al., 2025; Zhao et al., 2025), demonstrating strong performance and adaptability; however, real-world deployment remains challenging because LLMs often fail to follow instructions and use tools effectively—for example, they may issue retrieval queries misaligned with the search engine, yielding irrelevant or low-quality results and degrading task performance. While prompt engineering and per-agent Supervised Fine-tuning (SFT) can mitigate these issues, they impose substantial engineering overhead and adapt poorly to evolving requirements, motivating Reinforcement Learning (RL)-based methods for robust, *end-to-end* system-level optimization.

RL methods for LLMs enhance capabilities by

rewarding desirable behaviors, driving policy exploration toward higher expected return. In MASs where LLMs underpin individual agents, RL enables end-to-end optimization of both agent competence and system performance, typically cast as Multi-Agent Reinforcement Learning (MARL). By assigning rewards to system-level outcomes and fine-tuning the underlying LLMs, MARL supports task-adaptive refinement within the MAS. However, MARL faces two central challenges: *(i)* efficiently optimizing diverse agents towards global optima with minimal overhead, and *(ii)* capturing *indirect* inter-agent dependencies and *implicit* inter-trajectory connections.

Recent work applies Multi-Agent Proximal Policy Optimization (MAPPO) (Schulman et al., 2017; Yu et al., 2022) to optimize MASs (Liao et al., 2025a; Chen et al., 2025), treating the LLM backbone as the *actor* and using a large *critic* to estimate returns for policy-gradient updates. However, in multi-agent settings the critic must approximate joint action values over diverse agents, which is often unstable under task diversity and adds substantial memory and computation, limiting scalability. By contrast, recent Group-based Optimization Algorithms (GOAs) such as Group Relative Policy Optimization (GRPO) (Schulman et al., 2016; Yu et al., 2025) remove the critic and estimate advantages from relative rewards across group rollouts. While effective in *single-context* settings, extending them to *multi-context* MASSs with *implicit* inter-agent coupling remains largely unexplored.

To bridge this gap, we present the first systematic study of GOAs for MASS and introduce Multi-Agent Heterogeneous Group Policy Optimization (MHGPO), a GOA that optimizes MASSs via multi-agent rollout sampling and backward reward propagation, with advantages estimated from relative rewards within *heterogeneous groups*. Our main contributions are:

- Leveraging **parameter sharing** and **critic-free training**, MHGPO recasts MARL as *multi-task* optimization, achieving cheaper and more stable learning than PPO-based methods.
- With **backward reward propagation**, MHGPO captures *indirect* inter-agent dependencies; with **heterogeneous-group advantage estimation**, it models *implicit* cross-trajectory correlations, enabling end-to-end *global* optimization.

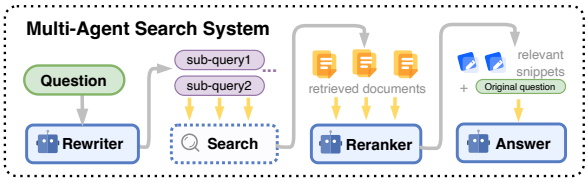

Figure 1: Illustration of the three-agent MASS: the *Rewriter* decomposes the question into sub-queries, the *Reranker* selects relevant snippets from retrieved documents, and the *Answerer* generates the final answer.

- We explore three rollout sampling strategies for MASSs—Independent Sampling (IS), Fork-on-first (FoF), and Round-robin (RR)—and their efficiency–effectiveness trade-offs.
- We formalize the first connection between parameter-shared MARL and single-context, RAG-oriented GOAs, showing that MHGPO yields **a near-aligned per-token advantage** yet empirically excels in stability and efficiency via multi-context optimization.
- On mainstream benchmarks, MHGPO consistently outperforms strong baselines in task performance and training efficiency.

2 Preliminary

2.1 Proximal Policy Optimization (PPO)

When adapted to LLM optimization, PPO retains the standard actor–critic structure with three models: an actor (policy) π_θ, a critic V_ϕ, and a frozen reference policy $\pi_{\theta_{\text{ref}}}$. Given a prompt q, the actor samples an output $o \sim \pi_\theta(\cdot \mid q)$. The critic estimates $V_\phi(q, o)$, a scalar baseline for advantage computation. The reference policy regularizes updates (typically via a KL penalty), stabilizing training. PPO then maximizes the following objective:

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_{q\sim\mathcal{D}, o_{\le t}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} [\min(r_t \hat{A}_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon)\hat{A}_t)], \tag{1}$$

where $r_t = \frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t|q,o_{<t})}$ is the importance-sampling ratio that constrains the policy update magnitude, and $\hat{A}_t$ denotes the advantage at time step t. In PPO, $\hat{A}_t$ is typically computed using Generalized Advantage Estimation (GAE). For the l-th token of the generated response o, the critic provides a value estimate V_l, while the reward model (or scoring function) assigns a reward R_l, with

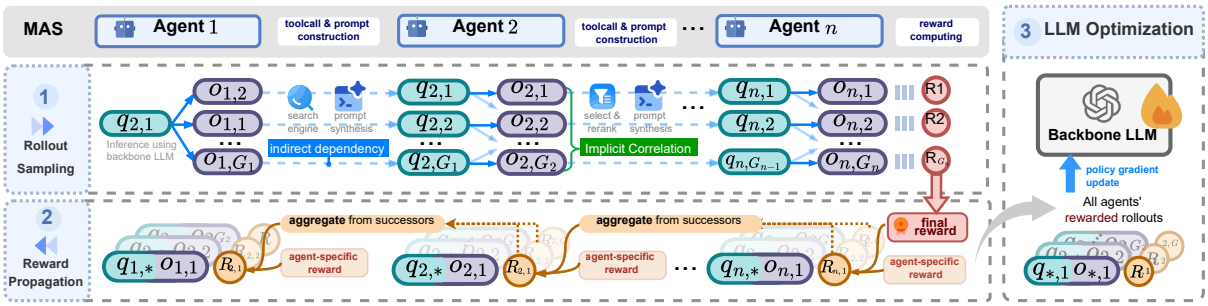

Figure 2: The proposed MHGPO framework proceeds as follows. For each MAS input (question), we first sample multi-agent rollouts, producing multiple trajectories. Each trajectory receives a terminal reward, which is propagated backward and aggregated to assign a reward to each rollout across trajectories. The resulting reward-labeled rollouts are then used to optimize the backbone LLM with Equation 6.

$l = 0, 1, 2, \ldots$. The advantage is estimated as

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \qquad \delta_l = R_l + \gamma V_{l+1} - V_l, \tag{2}$$

where $\gamma \in [0, 1]$ and $\lambda \in [0, 1]$ are the discount and GAE hyperparameters, respectively.

2.2 Group Relative Policy Optimization

Compared with PPO, instead of relying on a Critic model to conduct advantage evaluation, GRPO directly uses the within-group relative rewards as the basis for advantage evaluation. Specifically, for an input question q, GRPO will sample a group of G responses $\{o_i\}_{i=1}^{G}$. Thus, the advantage of the i-th response (shared across all time steps or only applied to the last time step) can be calculated as:

$$\hat{A}_i = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})} \tag{3}$$

2.3 Multi-Agent Search System

In this paper, we utilize a simple yet effective three-agent MASS as an example to investigate MARL algorithms, as illustrated in Figure 1. Following the design of previous work (Chen et al., 2025; Ma et al., 2023; Chang et al., 2025), this system comprises a *Rewriter*, responsible for generating retrieval queries tailored for search engines based on the original questions; a *Reranker*, which selects relevant items from a large pool of retrieval results to aid in answering the original questions; and an *Answerer*, which produces the final answers by integrating the original questions with the filtered and reranked retrieval results. This system can invoke external retrieval tools to address complex queries by leveraging supplementary information.

3 Method: MHGPO

In this section, we present Multi-Agent Heterogeneous Group Policy Optimization (MHGPO), the first GOA for MASs that estimates relative advantages over *heterogeneous* rollout groups. MHGPO comprises a general policy optimization framework and three group-based rollout sampling strategies. Although we focus on MASSs, the method is applicable to general MASs; thus, we use MAS and MASS interchangeably throughout the paper.

3.1 Multi-Agent Policy Optimization Framework

Following MAPPO (Yu et al., 2022), MHGPO uses *parameter sharing*: all n agents ($k \in 1, \ldots, n$) are instantiated with a single LLM backbone, which is jointly optimized to perform each agent role. This design simplifies training and reduces compute and memory overhead. Figure 2 summarizes the overall RL framework; details follow.

Multi-Agent Group Rollout Sampling. Given a single question $q \sim D$ sampled from the RL dataset, the MAS performs *group rollout sampling* by invoking its internal agents to collaboratively generate G final responses $\{o_i\}_{i=1}^{G}$. The sequence of intermediate steps from the input q to each final response constitutes a rollout *trajectory*. During sampling, the k-th agent produces a total of G_k input-output pairs, each corresponding to a distinct trajectory, denoted as $\{(q_{k,i}, o_{k,i}, m_{k,i})\}_{i=1}^{G_k}$. Here, $m_{k,i}$ denotes the *group identifier* associated with each rollout, as determined by a specific group rollout sampling algorithm. This identifier is subsequently used to aggregate rollouts into coherent groups. The resulting input-output pairs are collected as *agent rollouts* and serve as training data

for model optimization.

Backward Reward Propagation: Capturing Indirect Inter-Agent Dependencies After the MAS generates the final responses $\{o_i\}_{i=1}^{G}$ for a query q, a reward model or predefined reward rule assigns reward signals to these outputs, resulting in a shared reward set $\{R_i^{\text{shared}}\}_{i=1}^{G}$. Starting from the endpoints of the sampling trajectories, these shared rewards are then *propagated backward* through each trajectory to reach the preceding agents k, where they are *aggregated* for each agent's output. Specifically, for the i-th output $o_{k,i}$ produced by agent k, the corresponding shared reward is computed as:

$$R_{k,i}^{\text{shared}} = \text{Aggr}(\{R_{j,r}\}_{j>k}) \tag{4}$$

where j is the *direct successor* of agent k along the trajectory—i.e., j consumes k's output with no intermediate agents. We call (k, j) an *indirect-dependent* pair. Each $o_{j,r}$ is a response generated by agent j from inputs that include $o_{k,i}$. $\text{Aggr}(\cdot)$ denotes an aggregation operator, implemented as simple averaging by default. This reward-sharing scheme exposes indirect upstream–downstream dependencies even when agents do not share context.

After the shared rewards have been propagated, the final reward for each output $\{o_{k,i}\}_{i=1}^{G_k}$ is calculated by incorporating the agent-specific reward function $R_k^{\text{spe}}(\cdot)$, which typically imposes a penalty based on the output format of the agent. The resulting reward is given by $R_{k,i} = R_{k,i}^{\text{shared}} + R_k^{\text{spe}}(q_{k,i}, o_{k,i})$.

Heterogeneous Group Advantage Estimation: Implicit Cross-Trajectory Correlation Modeling Following GOAs, we estimate advantages using group-wise relative rewards, eliminating the need for a critic. For a single system-level input, the advantage of the i-th rollout from agent k is

$$\hat{A}_{k,i} = \frac{R_{k,i} - \text{mean}(\{R_{l,j} \mid m_{l,j} = m_{k,i}\})}{\text{std}(\{R_{l,j} \mid m_{l,j} = m_{k,i}\})}. \tag{5}$$

Here, advantages are normalized within the set of samples sharing the same *group identifier* m. The resulting scalar advantage is broadcast to all tokens, i.e., $A_{k,i}^t = A_{k,i}$. Unlike single-agent GRPO, a group may include rollouts from different prompts, yielding *heterogeneous groups*. This cross-trajectory normalization couples learning signals across diverse rollouts, exposing implicit trajectory relations that are critical for modeling and optimizing collaboration in the overall MAS.

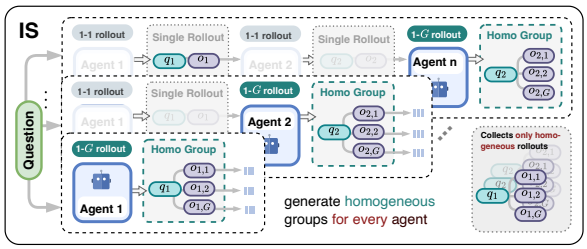

Figure 3: The all-homogeneous rollout strategy *Independent Sampling (IS)*, where each sample is propagated to all agents for one-to-G rollouts. Only the resulting homogeneous groups are used for model updates.

Multi-Agent Optimization Under parameter sharing, the RL objective for the MAS takes the same form as GRPO; however, training requires *aggregating* the losses from the n agents to enable collaborative optimization—casting multi-agent learning as a multi-task learning problem.

$$\mathcal{J}_{\text{MHGPO}}(\theta) = \mathbb{E}_{\substack{q\sim\mathcal{D} \\ \{o\}\sim\text{MAS}_{\theta_{\text{old}}}(\cdot|q)}} [\frac{1}{n}\sum_{k=1}^{n}\frac{1}{G_k}\sum_{i=1}^{G_k}\frac{1}{|o_{k,i}|} \sum_{t=1}^{|o_{k,i}|}\min(r_{k,i}^t\hat{A}_{k,i}^t, \text{clip}(r_{k,i}^t, 1-\epsilon, 1+\epsilon)\hat{A}_{k,i}^t)] - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}}), \tag{6}$$

$$r_{k,i}^t = \frac{\pi_\theta(o_{k,i}^t|q_{k,i}, o_{k,i}^{<t})}{\pi_{\theta_{\text{old}}}(o_{k,i}^t|q_{k,i}, o_{k,i}^{<t})}. \tag{7}$$

With the above objective, coordinated optimization of the multi-agent system can be performed based on all internal input-output pairs generated during the MAS rollout phase.

3.2 Rollout Sampling Strategies

A key challenge in the proposed multi-agent optimization framework is constructing the candidate group G_k' for each agent: executing *group rollouts* to obtain G_k, and then *forming completion groups* from the sampled trajectories. This section explores three strategies in MHGPO, each using a different mechanism for rollout sampling and group formation (Figures 3 and 4); full algorithms are given in the Appendix.

All Homogeneous: Independent Sampling (IS) We first introduce a baseline strategy for group rollout sampling, IS (Figure 3). Following the relative-advantage formulation in GRPO (Equation 3), each group is formed by sampling multiple rollouts from the *same* input, yielding a *homogeneous group*. Computing relative advantages within such groups

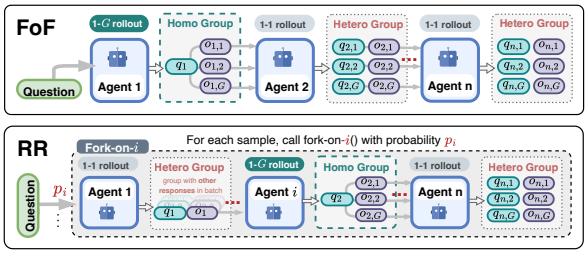

Figure 4: The proposed two *heterogeneous-group-based* rollout sampling strategies, namely *Fork-on-first (FoF)* and *Round-robin (RR)*.

provides a consistent, input-conditioned comparison of the LLM's responses.

Specifically, for each question $q \sim D$, we sequentially run the MAS to each agent and, upon reaching that agent, sample a 1-G group of rollouts. Rewards are computed via the backpropagation mechanism in MHGPO. Repeating this for all agents yields only 1-G homogeneous rollouts, totaling $n \times G$ samples organized into n independent homogeneous groups of size G. These are then used to fine-tune the backbone LLM, encouraging balanced performance across Agent-specific tasks.

Introduce Heterogeneity: Fork-on-first (FoF) The homogeneous-only IS baseline is inefficient and myopic: it incurs substantial redundant rollouts and optimizes each agent independently, ignoring inter-agent interactions. We therefore propose FoF, a simple alternative that removes redundant sampling while explicitly capturing interactions among agents (Figure 4, upper). Crucially, FoF supports advantage estimation over *heterogeneous groups*.

For each question, FoF samples G rollouts *only* at the entry agent of the MAS; all subsequent agents perform one-to-one rollouts. Consequently, the entry agent produces G rollouts from the same input (a homogeneous group), while each downstream agent receives G different inputs induced by upstream rollouts (forming *heterogeneous groups*). Relative advantages are computed within the corresponding groups, producing G reward signals at the system output and requiring only nG rollout operations over the entire MAS.

Less Utilization for More Homogeneity: Round-robin (RR) FoF improves per-agent sample utilization without discarding rollouts and performs well in our preliminary results (Figure 5). However, because it always forks at the first agent, only the entry agent forms homogeneous rollout groups; downstream agents rarely receive identical inputs when estimating advantages, which can slow convergence. We therefore propose RR, which randomizes the fork point across agents. For each input sample, we select agent i as the fork point with probability p_i. If the sample forks at i, agents $j \geq i$ follow the FoF procedure for advantage estimation, while agents $j < i$ execute only one rollout. For each $j < i$, we compute relative advantages by grouping that rollout with $\lfloor p_i \cdot \text{batch_size} \rfloor - 1$ other samples in the same batch that also fork at i. Consequently, each agent forms homogeneous rollout groups with probability p_i, while only $p_1 \cdot \text{batch_size}$ samples incur the full FoF rollout cost; the rest require fewer rollouts, reducing overall resource usage.

More Utilization: FoF with Over-sampling Complementary to RR, we also study whether higher sample utilization improves optimization under the FoF framework, despite increased training cost. Concretely, for each sampled trajectory we over-sample the *terminal agent* by generating G rollouts (instead of one), yielding a denser estimate of the final reward. We then propagate and aggregate this signal to earlier agents for reward estimation. We denote this variant as *FoF(os)*.

4 Experiments & Discussion

4.1 Experiment Setup

Implementation Details We instantiate each agent using prompts adapted from Chen et al. (2025). Following Su et al. (2024), we adopt a Wikipedia dump as the retrieval corpus and deploy `contriever` (Izacard et al., 2022) as the search backend for the *retriever* module. Consistent with standard multi-hop RL practice, we train for one epoch. Unless stated otherwise, MHGPO uses group rollouts with size $G = 4$, and MHGPO-RR uses round-robin probabilities $(0.7, 0.1, 0.2)$. Further details, including the effect of varying G, are provided in the Appendix.

Datasets and Models We adopt commonly used evaluation datasets in search systems, including the multi-hop datasets HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022), for training and evaluation. Llama3.1-Instruct-8B was selected as the backbone implementation for the MAS.

Evaluation Metrics Since the selected datasets all contain gold-standard answers, we use the Exact Match (EM) score, which evaluates the exact

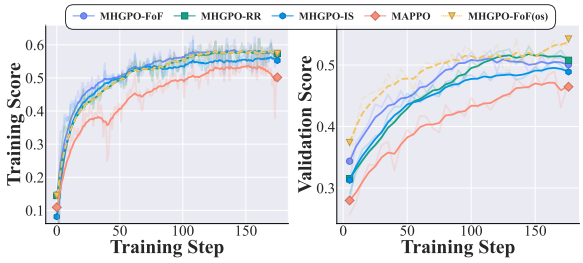

Figure 5: Training and validation scores (F1-score) during one epoch of training on the HotpotQA training set using different RL algorithms. Validation is performed every 5 steps.

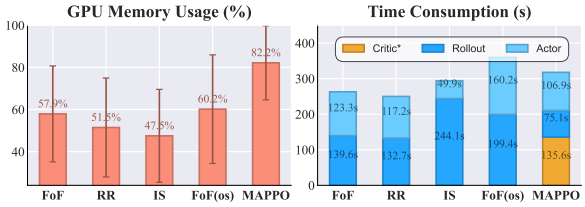

Figure 6: GPU memory usage (%) and average training-step time (s) for different MARL algorithms, averaged over the first 50 steps.

match between the system's final answer and the reference answer, as well as the F1-score and Accuracy (which measures the overlap between the normalized prediction and the gold answer) as the three primary evaluation metrics.

4.2 Main Results

We evaluate MHGPO on HotpotQA with a three-agent MASS trained *from scratch* (no SFT warm-up). We use a batch size of 512 and train for one epoch (176 steps). Baselines include MASs instantiated with different LLMs, an MAPPO-optimized MAS, and single-context RAG systems: Search-o1 (Li et al., 2025), which injects search into long-horizon LLM reasoning, and R1-Searcher (Song et al., 2025), which further optimizes this pipeline end-to-end with RL.

Question 1. *How does MHGPO improve training stability and efficiency compared to PPO-style methods?*

We evaluate learning dynamics by tracking training reward and validation accuracy on the first 500 HotpotQA test examples over one epoch (Figure 5), and quantify training cost separately (Figure 6). MAPPO exhibits noisier optimization and a lower performance ceiling, while incurring substantially higher memory and wall-clock overhead from maintaining and updating a full critic. In

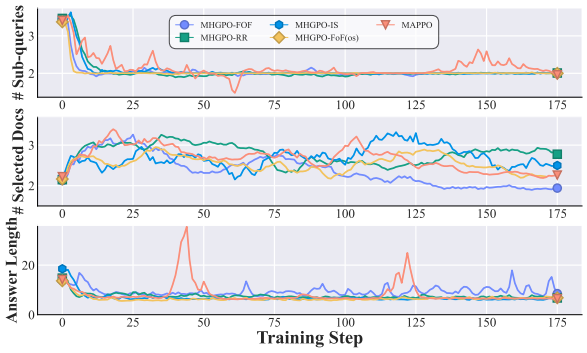

Figure 7: Training dynamics of agent outputs under different MARL algorithms. Rows correspond to the rewriter, reranker, and answerer (top to bottom).

contrast, MHGPO converges more smoothly under markedly smaller resource budgets. The IS variant underperforms because independent optimization within homogeneous groups fails to capture inter-agent dependencies, limiting system-level improvement. By estimating advantages over *heterogeneous* groups, FoF and RR enable end-to-end optimization with global rewards, yielding stronger and more stable convergence with minimal overhead. Notably, RR further reduces compute and memory by trading off sample efficiency, yet matches or surpasses the final performance of stronger baselines, making it attractive in large-sample regimes. Finally, FoF(os) achieves the best validation trajectory: although slightly weaker in training reward than FoF, it improves generalization by training longer without increasing the memory footprint. Overall, these results highlight the stability–efficiency gains and practical scalability of the MHGPO framework.

We also track each agent's outputs throughout training (Figure 7) to show how end-to-end optimization shapes individual behaviors. MAPPO exhibits pronounced instability, plausibly because a single critic must fit diverse objectives across three agents. In contrast, variants of MHGPO converge smoothly with higher cross-agent stability: the rewriter consistently produces two rewritten queries, the reranker settles on selecting 2–3 documents, and the final response becomes succinct, with answer length converging to < 10.

Question 2. *How does MHGPO perform on the test set compared to the baselines? Can MHGPO generalize to other datasets?*

After MARL training, we deploy the learned policy as the backbone of our MAS and evalu-

Table 1: Test performance of RL algorithms and training-free baselines on mainstream QA datasets, measured by Accuracy (%), Exact Match (%), and F1 (%). All trainable methods fine-tune the same Llama3.1-8B-Instruct checkpoint with one epoch of RL on HotpotQA; $*$ denotes out-of-domain (OOD) evaluation.

Methods/Models		**HotpotQA**			*2WikiMultihooQA			*MuSiQue			AVG	
		Acc	EM	F1	Acc	EM	F1	Acc	EM	F1	ID	OOD
LLM	Llama3.1-8B	21.269	14.018	22.782	16.963	7.069	20.817	5.492	1.365	2.813	19.356	9.087
	Qwen2.5-72B	30.304	25.564	36.183	29.373	27.234	32.038	7.737	5.337	15.411	30.684	19.522
LLM + RL	PPO	25.434	1.201	24.521	12.302	1.371	9.203	3.452	2.600	8.021	17.052	6.158
	GRPO	27.643	0.500	27.421	14.019	0.676	11.025	4.096	2.400	9.287	18.521	6.917
LLM + RAG (+RL)	Search-o1	23.023	17.920	25.702	24.231	17.802	25.680	8.012	5.673	15.982	22.215	16.230
	R1-Searcher(PPO)	37.493	35.982	47.232	34.544	29.982	36.403	9.082	7.230	16.541	40.236	22.297
	R1-Searcher(GRPO)	36.920	32.673	45.392	35.012	30.861	37.620	10.231	7.975	19.200	38.328	23.483
LLM + MAS	Llama3.1-8B	20.800	14.000	21.452	22.281	15.872	22.427	4.220	2.524	8.044	18.751	12.561
	Qwen2.5-72B	32.519	24.254	34.631	36.379	18.511	25.630	7.075	4.717	11.959	30.468	17.379
LLM +MAS +MARL	MAPPO	38.217	34.450	46.400	35.790	31.767	38.722	11.626	9.350	19.738	39.689	24.499
	MHGPO-IS	37.677	33.707	45.582	34.796	31.115	38.050	9.888	7.944	18.421	38.989	23.369
	MHGPO-FoF	40.459	36.570	49.429	36.299	31.750	39.089	**12.702**	**10.012**	**21.633**	42.153	**25.248**
	MHGPO-RR	**40.864**	**37.043**	**49.724**	**36.840**	**31.838**	**39.388**	11.378	9.350	21.144	**42.544**	24.990
	MHGPO-FoF(os)	41.512	37.623	50.858	37.222	32.801	40.368	12.660	10.923	22.677	43.331	26.109

Table 2: Ablation of role-wise training in MHGPO (FoF) on HotpotQA. We compare full collaborative optimization with training a single role only (*) and freezing one role (-). Results are averaged over three random seeds.

	ACC	EM	F1	AVG
MHGPO	40.421±0.230	36.120±0.273	49.534±0.199	**42.025**
*Rewriter	36.798±0.560	30.481±0.450	46.151±0.492	37.810
*Reranker	24.240±0.621	19.082±0.490	30.760±0.392	24.694
*Answer	21.180±0.390	18.667±0.383	29.027±0.410	22.958
-Rewriter	31.127±0.490	23.983±0.530	33.689±0.502	29.599
-Reranker	38.242±0.217	34.180±0.317	46.792±0.252	39.738
-Answer	38.608±0.473	35.983±0.359	48.982±0.563	41.190

ate retrieval-intensive QA benchmarks. Table 1 compares MASs optimized by different RL algorithms against strong baselines, including standalone LLMs, unoptimized MASs, and single-context RAG systems; we also report RL-tuned LLMs trained on HotpotQA for direct QA.

Table 1 shows that standalone LLMs are weak on these tasks, and na"ively wrapping them into an MAS without optimization can further hurt performance due to instruction-following failures, particularly for smaller models. In contrast, end-to-end MARL yields a substantial gain, improving the unoptimized MAS by more than $2\times$.

Within MARL, the FoF and RR variants of MHGPO markedly outperform MAPPO on the in-domain HotpotQA test set; RR is slightly better than FoF while also being the most compute-efficient. On out-of-domain 2WikiMultihopQA and MuSiQue, all MARL methods consistently surpass the unoptimized baseline, underscoring the robustness of multi-agent optimization, with MHGPO delivering the most consistent improvements across datasets. Moreover, MHGPO also beats single-context RAG and avoids the GRPO instability reported by R1-Searcher (Song et al., 2025), which we attribute to shorter per-context trajectories and implicit (rather than explicit) cross-stage coupling, leading to faster and more reliable group-based convergence.

4.3 Ablation Study & Discussion

Question 3. *How does heterogeneous group advantage, together with reward propagation, capture implicit inter-trajectory correlations?*

Single-Context Transcript Equivalence Before analyzing results, we first relate MHGPO to single-context GOAs: Under parameter sharing and reward broadcasting, MHGPO is approximately equivalent to applying GRPO to a single tool-augmented transcript over an augmented context space; the tightness of this equivalence depends on how explicitly each agent prompt embeds the full trajectory context. When the embedding is fully explicit, each token receives the same optimization signal as in a single-context LLM. More generally, **MHGPO can be viewed as a principled GRPO-style extension to MASs that relaxes strict single-context assumptions to model implicit cross-context dependencies**, and optimizes

Table 3: Ablation of the reward-aggregation operator $\mathrm{Aggr}(\cdot)$ in backward reward propagation (Eq. 4). We evaluate MHGPO-RR and MHGPO-FoF(os) on HotpotQA, reporting Exact Match (EM) and F1.

Aggr	MHGPO-RR		MHGPO-FoF(os)	
Function	EM	F1	EM	F1
AVG()	**37.043±0.251**	**49.724±0.408**	**37.623 ±0.190**	**50.858±0.360**
MAX()	34.129±0.377	46.343±0.532	36.704±0.368	49.020±0.489
MIN()	35.840±0.280	47.930±0.398	36.920±0.310	49.842±0.452
Rand()	35.643±0.730	46.824±0.808	36.602 ±0.520	49.013±0.674

in a more efficient, multi-context way. See Appendix D for details.

Therefore, heterogeneous groups together with reward propagation make per-token advantages in GOA-based MASs training better align with single-context GRPO. Empirically, they enable *system-level optimization*: rollouts within a heterogeneous group share the same system input, inducing trajectory-level couplings. In ablation, our baseline MHGPO-IS uses *all-homogeneous* groups and performs substantially worse than heterogeneous variants in Table 1. A complementary interpretation for why heterogeneous grouping remains viable is *training-time homogenization*: as the shared backbone LLM undergoes entropy reduction Cui et al. (2025), intra-group diversity decreases over training (Figure 11, Appendix).

We further ablate the role of individual agents to assess the importance of inter-agent interactions. As shown in Table 2, optimizing only a single agent or freezing any agent markedly degrades performance, as *global* correlations are insufficiently modeled. Notably, independently optimizing all three agents corresponds exactly to MHGPO-IS.

Another key component of MHGPO is reward *aggregation* along branched trajectories. To evaluate the aggregation operator $\mathrm{Aggr}(\cdot)$, we ablate its instantiations on HotpotQA using the two branched variants, RR and FoF(os). Table 3 shows that AVG consistently yields the best EM/F1, indicating the most stable and informative credit assignment. In contrast, MAX/MIN introduce biased signals (with MIN outperforming MAX here), while RAND increases variance and degrades performance.

5 Related Works

LLM-based Multi-agent Systems Beyond a standalone LLM, MASs—teams of interacting agents, potentially powered by different LLMs—can better decompose complex objectives and coordinate specialized reasoning. Emerging agent-development frameworks such as AutoGen (Wu et al., 2023) and OWL (Hu et al., 2025) further highlight the growing practicality of MASs in real-world deployments. In search-based QA, prior work consistently reports gains from collaboration: Chang et al. (2025) cast RAG as a sequential MAS, using structured inter-agent interaction to improve retrieval-augmented generation; Wu et al. (2025) introduce dynamic routing that assigns subtasks to expert agents; and Zhao et al. (2025) incorporate backtracking to support reversible multi-hop reasoning, strengthening RAG-based QA.

End-to-end RL for MASs Despite their promise, MASs remain brittle in practice: Cemri et al. (2025) attribute failures to fragile system design and inconsistent LLM formatting. This motivates end-to-end reinforcement learning for MASs (i.e., MARL) to learn robust, adaptive coordination instead of hand-crafted control logic. The dominant paradigm, centralized training with decentralized execution, is represented by MAPPO (Yu et al., 2022) and HAPPO (Kuba et al., 2022), which often adopt *parameter sharing* for scalable training. In LLM-based MASs, recent RL efforts include component-level optimization (Kulkarni et al., 2024; Ma et al., 2023), applying MAPPO to multi-agent RAG with a shared LLM backbone (Chen et al., 2025), and tailoring RL frameworks for end-to-end optimization across interacting agents (Liao et al., 2025b).

6 Conclusions

In this paper, we propose MHGPO, a novel reinforcement learning framework for end-to-end optimization of multi-agent LLM systems. By leveraging relative group advantages via a two-phase sampling-propagation scheme, MHGPO mitigates the instability and computational overhead typically associated with MAPPO in MASs. We explore three group-based rollout strategies within MHGPO, and show that both FoF and RR—which compute relative advantages over heterogeneous groups—deliver strong performance, with RR providing additional computational efficiency. Experiments on multi-hop QA tasks demonstrate that MHGPO consistently outperforms baselines in both accuracy and training efficiency. While our evaluation focuses on MASS, the framework is readily generalizable to broader MAS settings. Future work will extend our validation to more diverse scenarios. We hope these findings will encourage further exploration of reinforcement learning for LLM-based multi-agent systems.

7 Limitations

Task scope (QA-only). Our empirical evaluation is restricted to *question answering*, primarily multi-hop QA benchmarks. While QA offers a controlled and well-established testbed for studying multi-agent LLM coordination and end-to-end optimization, it does not cover other important agentic workloads (e.g., long-horizon planning, code generation with tool execution, interactive decision making, or open-ended web navigation). Consequently, the generality of MHGPO beyond QA-centric settings remains to be explored, and different task structures may introduce new challenges in reward design, credit assignment, and coordination dynamics.

Model scale (up to 8B). Due to computational constraints, all experiments are conducted with relatively small backbone models (up to 8B parameters). While this improves reproducibility and enables controlled ablations, it limits our ability to assess scaling behavior and emergent coordination effects that may arise with larger LLMs. Future work should evaluate MHGPO with stronger backbones and larger agent teams to better understand its scalability, robustness, and practical impact in higher-capacity regimes.

References

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya G. Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Why do multi-agent LLM systems fail? *CoRR*, abs/2503.13657.

Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, and Na Zou. 2025. MAIN-RAG: multi-agent filtering retrieval-augmented generation. *CoRR*, abs/2501.00332.

Yiqun Chen, Lingyong Yan, Weiwei Sun, Xinyu Ma, Yi Zhang, Shuaiqiang Wang, Dawei Yin, Yiming Yang, and Jiaxin Mao. 2025. Improving retrieval-augmented generation through multi-agent reinforcement learning. *CoRR*, abs/2501.15228.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. 2025. The entropy mechanism of reinforcement learning for reasoning language models. *CoRR*, abs/2505.22617.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Mengkang Hu, Yuhang Zhou, Yuzhou Nie Wendong Fan, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Ping Luo, and Guohao Li. 2025. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. 2022. Trust region policy optimisation in multi-agent reinforcement learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Mandar Kulkarni, Praveen Tangarajan, Kyung Kim, and Anusua Trivedi. 2024. Reinforcement learning for optimizing RAG for domain chatbots. *CoRR*, abs/2401.06800.

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

Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. 2025a. Marft: Multi-agent reinforcement fine-tuning. *Preprint*, arXiv:2504.16129.

Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. 2025b. MARFT: multi-agent reinforcement fine-tuning. *CoRR*, abs/2504.16129.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *CoRR*, abs/2305.14283.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *CoRR*, abs/2503.05592.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. DRAGIN: dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 12991–13013. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Feijie Wu, Zitao Li, Fei Wei, Yaliang Li, Bolin Ding, and Jing Gao. 2025. Talk to right specialists: Routing and planning in multi-agent system for question answering. *CoRR*, abs/2501.07813.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *CoRR*, abs/2308.08155.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre M. Bayen, and Yi Wu. 2022. The surprising effectiveness of PPO in cooperative multi-agent games. In *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*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476.

Xinjie Zhao, Fan Gao, Rui Yang, Yingjian Chen, Yuyang Wang, Ying Zhu, Jiacheng Tang, and Irene Li. 2025. Reagent: Reversible multi-agent reasoning for knowledge-enhanced multi-hop QA. *CoRR*, abs/2503.06951.

A Detailed Algorithm

A.1 Multi-Agent Policy Optimization

The complete optimization procedure of the proposed MHGPO framework is outlined in Algorithm 4. Central to this process is the sampling function $\mathcal{H}(\theta)$, which can be instantiated using one of the IS, FoF, or RR strategies. This function is pivotal for trajectory sampling within the MAS, as it determines both the sampling behavior and the group assignment for each rollout. In the prototypical MASS considered in this work, the reward function $\mathcal{R}$ is instantiated as the F1 score.

A.2 Rollout Sampling Strategies

To introduce the proposed rollout sampling algorithms, we begin by presenting a core utility function, *fork_on()*, defined in Algorithm 5. This function facilitates group rollout by conducting one-to-one trajectory sampling before and after the *first interaction with Agent* i, and performing a one-to-many branching exclusively at Agent i, designated as the fork point. Concretely, the function generates G parallel trajectories starting from the fork point and assigns group identifiers based on the index of the forking agent. The output is a complete set of group-specific forked trajectories.

A.2.1 Fork-on-first (FoF)

The FoF strategy leverages this utility by directly invoking *fork_on()* with the first agent in the trajectory designated as the fork point, as detailed in Algorithm 1.

A.2.2 Independent Sampling (IS)

In contrast, IS executes an exhaustive search across all agents for each input q, invoking *fork_on()* at every potential fork point. After sampling, it filters out all *heterogeneous* rollout groups and retains only the homogeneous ones for downstream training.

A.2.3 Round-robin (RR)

The RR method operates at the batch level. For each sample in the batch, it randomly selects an Agent i as the fork point according to a predefined probability distribution $\{p_i\}$. Upon processing the entire batch, rollout samples that belong to singleton groups (i.e., groups with no shared samples) are re-shuffled and regrouped prior to training, as specified in Algorithm 3.

Table 4: Key hyperparameters for the implementation of MHGPO and MAPPO.

Hyperparameter	Explanation	Value(s)
`lr`	learning rate	(5e-7, 1e-6)
`rollout.n`	group size for GOAs	4
`batch_size`	batch size	512
ϵ	clip range	0.2
`top_n`	param for top-n sampling	0.9
β	param for KL penalty	0.001

B Implementation Details

All experiments were run on a single machine with eight NVIDIA H100 GPUs (80 GB HBM3 each) and an Intel Xeon Platinum 8468 CPU. For the MAPPO baseline, we instantiate the critic with the same backbone LLM architecture as the actor to ensure comparable modeling capacity. For R1-Searcher and Search-o1, we use their official default implementations and modify only two components: the backbone LLM and the retrieval engine.

B.1 Training Dataset

We utilized the training set of HotpotQA (Yang et al., 2018), comprising 90,447 samples, as the reinforcement learning training data. Only the questions from the dataset were used as model inputs, with the corresponding answers serving as ground-truth labels for computing rewards based on the F1 score. Representative examples are presented in Figure 6. These questions require the model to invoke external tools for search and perform multi-hop reasoning based on the retrieved results, effectively evaluating the agent's tool-calling capabilities.

B.2 Hyper-parameters

The hyperparameter ranges and detailed configurations used in all experiments are presented in Table 4. Specifically, the Critic network for MAPPO employs a learning rate of 1e-6. For the Actor networks in both MAPPO and MHGPO, learning rates are selected from 1e-6, 5e-7, with 5e-7 empirically demonstrating superior performance. All other parameters are shared across different MARL algorithms.

Table 5: HotpotQA performance of MHGPO with varying group sizes, averaged over three random seeds.

Group Size	ACC	EM	F1	AVG
3	36.421±0.394	34.120±0.320	43.534±0.262	38.025
4	40.421±0.230	36.120±0.273	**49.534±0.199**	42.025
5	39.989±0.170	36.021±0.189	49.379±0.210	41.796
6	**41.030±0.190**	**36.400±0.202**	49.422±0.238	**42.284**

B.3 Agent Prompts

We follow the three-agent MAS design proposed by Chen et al. (2025), with only minor modifications to the agents' prompts and output formatting to simplify the system implementation.

Prompt for Rewriter

System: You are a professional assistant proficient in transforming complex or unclear questions into simpler, more searchable sub-questions.
Please assist me in rewriting or breaking down the provided questions into sub-questions to make it easier to search for answers using a search engine. The rewritten sub-questions must have logical connections and dependencies, and should not be overly repetitive in meaning.
Output the sub-questions in the form of a string list, with the format:

```
### <query1>; <query2>; <query3>; ... ###
```

User: Original question is `{raw_question...}`

Prompt for Reranker

System: You are a helpful, respectful, and honest assistant. Your task is to identify and output the IDs (such as 0, 1, 2, etc.) of the candidate Documents that are useful for answering the given Question.
Now, simply output the IDs of those candidate Documents that can assist in answering the Question: `{raw_question}`. Arrange them in the order where the more relevant documents are listed first.
User: Question is: `{raw_question}`
Document0: `{title:..., snippet:...}`
Document1: `{title:..., snippet:...}`
...
Do not output anything else.

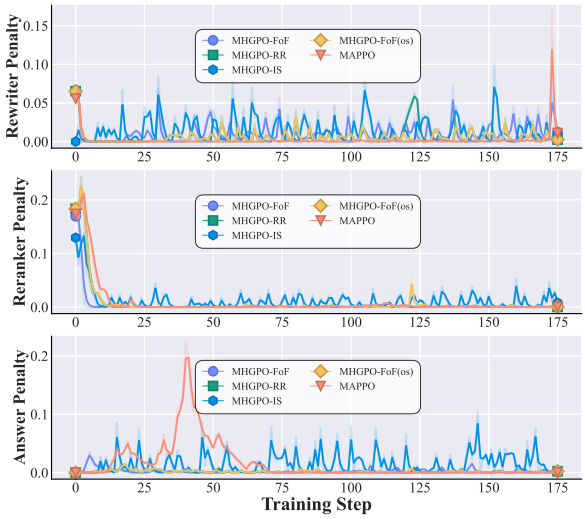

Figure 8: Agent-specific rewards (format penalty) for each agent in the three-agent MAS during one training epoch on HotpotQA using different MARL algorithms.

Prompt for Answerer

System: You are a helpful, respectful and honest assistant. Your task is to predict the answer to the question based on the given documents. If you don't know the answer to a question, please don't share false information. Answer the question as accurately as possible.
User: Question is: `{raw_question}`
Document 0: `{title: ..., snippet: ...}`
Document 1: `{title: ..., snippet: ...}`
...
Now, answer the Question: `{raw_question}`, based on the above Documents.

B.4 Agent-specific Rewards

For agent-specific reward design, we primarily employ a format reward to enforce output standardization for each agent, ensuring seamless functionality across intermediate system interfaces. Specifically, the Rewriter incurs a penalty proportional to the number of rewritten queries, receiving a -0.5 penalty if more than four queries are generated. The Reranker is penalized based on the format and redundancy of selected documents, with a -0.5 penalty for duplicated entries or index out-of-bounds errors. The Answerer faces a penalty tied to the length of the generated answer, incurring a

-1.0 penalty for excessively long responses. This reward structure facilitates precise control over individual agent behavior during optimization.

C Detailed Experiment Results

C.1 Training Dynamics

In order to meticulously observe and compare the characteristics of different MARL algorithms during the training process, we have recorded the detailed metrics during the MARL training process on HotpotQA for the example MAS under consideration. Figure 8 separately records the changes in the format penalties obtained by different agents during the training process within the considered example 3-agent MAS. It can be observed that all methods can quickly find the correct output format and thus avoid penalties (the decrease within 10 steps reaches within 0.1). During this process, MAPPO once again demonstrates instability, with the possibility of the output format of agents collapsing midway. In addition, FoF and RR are hardly troubled by the format and consistently obtain relatively low penalties.

C.2 Impact of Group Size

Table 6: Examples of multi-hop questions and answers in HotpotQA.

Question	Answer
What is the relationship of Yeshahework Yilma's mother to the man who was Ethiopia's emperor from 1930 to 1974?	niece
What role on "Switched at Birth" was played by the actor who is being replaced by Daniel Hall on "The Young and the Restless"?	Tyler "Ty" Mendoza
Are the Sleepers located north or south of the Kancamagus Highway?	south

Table 5 studies how the heterogeneous-group size in MHGPO-FoF affects final performance. Increasing the group size from 3 to 4 yields a clear jump across all metrics (ACC/EM/F1), indicating that a larger comparison set provides a stronger relative-reward signal for advantage estimation. Performance then largely saturates: group sizes 4–6 achieve very similar F1 (49.38–49.53) with small variance across seeds, suggesting that the learned policy is robust once the group is moderately sized. The best overall average is obtained at group size 6 (AVG 42.284), which also attains the highest ACC and EM, while group size 4 achieves the top F1 and a comparable AVG. These results imply a favorable trade-off: moderately larger groups improve stability and accuracy, but further scaling delivers diminishing returns on F1, making group sizes in the 4–6 range a reliable default in practice.

C.3 Progressive Homogenization of Rollout Groups

We hypothesize that the basic effectiveness of the Heterogeneous-group-based Advantage Estimation employed in MHGPO for MAS stems, in part, from the progressive convergence of heterogeneous groups into homogeneous ones during training. Specifically, sampled trajectories from heterogeneous groups gradually become more uniform. This observation is supported by empirical evidence, as illustrated in Figure11 and Figure 10. The similarity among group rollout samples, quantified by the F1 score, consistently increases for each agent throughout the training process. For the *Rewriter*, output similarity reaches approximately 0.7 between steps 25 and 30, indicating that the *Reranker* likely receives nearly identical sets of retrieved documents. Consequently, the *Reranker* effectively optimizes over a de facto homogeneous group, which likely contributes to the significant performance improvements observed from step 25 onward.

C.4 Case Study

To investigate the specific improvements brought by MARL to multi-hop question answering, we examine the performance of the proposed MAS system on a representative QA example, comparing its behavior in the early stage (step 5) and the later stage (step 160) of reinforcement learning, as illustrated in Figure 9.

At step 5, the system produces an incorrect answer (*unknown*) because the selected relevant snippets do not contain useful information, despite the seemingly reasonable sub-queries generated by the Rewriter. In contrast, by step 160, the MAS successfully answers the question. This improvement stems from the Reranker correctly identifying two relevant documents—biographical entries on Scott Derrickson and Ed Wood. Interestingly, the Rewriter now produces only two sub-queries, which, while less intuitive to humans, align better with the behavior of the `contriever` retriever engine, enabling the retrieval of the desired documents.

This example highlights how reinforcement learning can help agents like the Rewriter adapt to their environment (i.e., the retrieval engine), thereby overcoming performance bottlenecks.

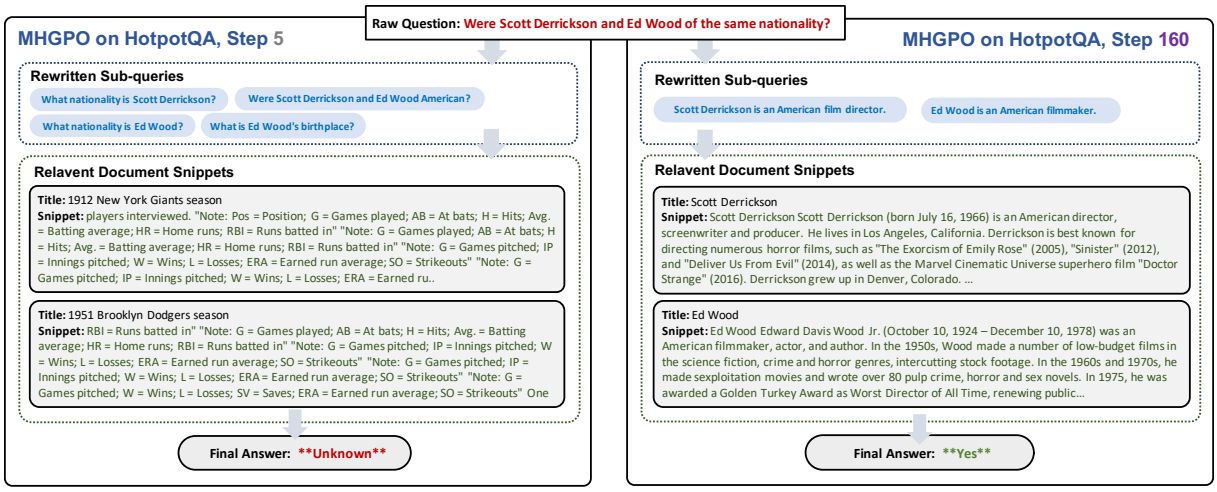

Figure 9: Case study: An example question is processed by the example MAS, with outputs from each stage shown for comparison between MHGPO-FoF after 5 steps and after 160 steps of training on HotpotQA.

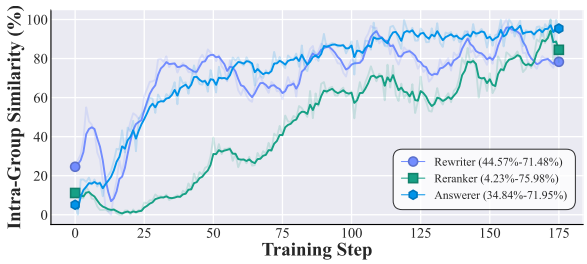

Figure 10: The intra-group similarity for each agent's rollout, measured by the average pairwise F1 score between generated responses. For consistency and ease of comparison across agents, similarity scores are normalized to the range [0, 1], with the corresponding minimum and maximum values indicated in the legend.

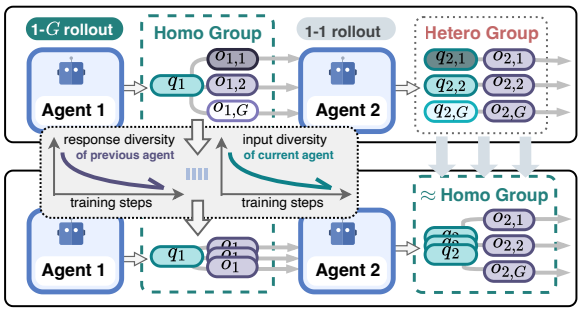

Figure 11: Illustration of how the heterogeneous groups produced by rollout sampling in MAS gradually transition into homogeneous groups during MHGPO.

D Single-Context Transcript Equivalence via Parameter Sharing and Reward Propagation

In this section, we show that MHGPO's **parameter sharing + reward propagation** yields an equivalent learning problem to training a single LLM that generates one *single-context transcript* (e.g., $\langle q, a, \texttt{<toolcall>}, q, a, \ldots \rangle$): the per-token reward-weighted score terms match between the MAS execution and this single-transcript view. We detail the analysis below.

D.1 MAS Trajectory and Single-Context Transcript

Fix an input question $q \sim \mathcal{D}$ and a sampled trajectory index i. The MAS executes agents and produces

$$(q_{1,i}, o_{1,i}) \;\rightarrow\; (q_{2,i}, o_{2,i}) \;\rightarrow\; \cdots \;\rightarrow\; (q_{n,i}, o_{n,i}),$$

where $q_{1,i} = q$ and $o_{k,i} = (o_{k,i,1}, \ldots, o_{k,i,T_{k,i}})$ is agent k's output sequence. Each agent input $q_{k,i}$ is constructed from system templates, tool outputs, and upstream realized texts.

By contrast, we construct a *single-context transcript* that alternates fixed workflow text with generated tokens:

$$x_{1,i} \triangleq q_{1,i},$$

$$x_{k+1,i} \triangleq \mathrm{Build}_{k+1}(x_{k,i}, o_{k,i}), \quad k = 1, \ldots, n,$$

where $x_{k,i}$ is the full prefix immediately before sampling $o_{k,i}$, and Build_{k+1} deterministically appends role templates, tool-call markers, tool outputs, separators, and required metadata. By construction, $q_{k,i}$ is a suffix of $x_{k,i}$.

Let the final system output o_i be a deterministic function of $\{o_{k,i}\}_{k=1}^{n}$, with terminal reward R_i. We show below that MHGPO over an MAS rollout is equivalent to RL over this single-context transcript.

D.2 Assumptions

(A1) Parameter sharing. All agents share the same backbone autoregressive policy π_θ.

(A2) Deterministic prompt construction. All tool/retrieval results that influence downstream generation are explicitly *materialized as tokens* in $x_{k+1,i}$. Consequently, in the single-context transcript, Build_{k+1} is deterministic given $(x_{k,i}, o_{k,i})$; equivalently, in the MAS formulation, $q_{k+1,i} = \Phi_k(o_{k,i})$ for a deterministic mapping Φ_k.

(A3) Context sufficiency (inter-agent Markov property). For each k, i, the constructed prompt $q_{k,i}$ is sufficient for agent k's generation:

$$\pi_\theta(o_{k,i} \mid x_{k,i}) \;=\; \pi_\theta(o_{k,i} \mid q_{k,i}).$$

This assumption is strong in general, but it is well aligned with our MAS setting. Concretely, $q_{k,i}$ is produced by a deterministic prompt-construction routine that *materializes* into text all upstream agent outputs and all tool-call returns that are available to the system (Assumption A2), and the backbone LLM is always queried on $q_{k,i}$ (a suffix of $x_{k,i}$) with fixed role templates. In our scenario, tool calls (retrieval, search, etc.) return discrete results (e.g., top-K documents, snippets, or structured tool outputs) that are explicitly inserted into $q_{k,i}$; therefore, conditioning on the full transcript prefix $x_{k,i}$ provides no additional actionable information beyond what is already contained in $q_{k,i}$. In other words, $q_{k,i}$ is an information-preserving summary of the preceding transcript with respect to agent k's decision, which is precisely the intended design of the workflow prompt synthesis.

D.3 Probability Equivalence to a Single-Context Transcript

Proposition 1 (Single-context transcript probability equivalence)**.** *Under Assumptions (A1)–(A3), the MAS rollout distribution equals the distribution induced by a single autoregressive LLM generating the same transcript:*

$$\begin{aligned}&\mathrm{MAS}_\theta(o_{1,i}, \ldots, o_{n,i} \mid q_{1,i})\\ &=\; \pi_\theta(o_{1,i}, \ldots, o_{n,i} \mid x_{1,i}),\end{aligned}$$

and both factorize as

$$\prod_{k=1}^{n} \prod_{t=1}^{T_{k,i}} \pi_\theta(o_{k,i,t} \mid x_{k,i}, o_{k,i,<t})\,.$$

Proof. We show that both processes assign the same probability to each generated token, hence the same joint probability.

Single-transcript process. Immediately before generating $o_{k,i}$, the transcript prefix is $x_{k,i}$. Autoregressive generation yields

$$\begin{aligned}&\pi_\theta(o_{1,i}, \ldots, o_{n,i} \mid x_{1,i})\\ &= \prod_{k=1}^{n} \prod_{t=1}^{T_{k,i}} \pi_\theta(o_{k,i,t} \mid x_{k,i}, o_{k,i,<t}).\end{aligned}$$

MASS process. At stage k, the system constructs the agent prompt $q_{k,i}$ and samples $o_{k,i}$ token-by-token from $\pi_\theta(\cdot \mid q_{k,i})$:

$$\mathrm{MAS}_\theta(o_{k,i} \mid \text{history}) = \prod_{t=1}^{T_{k,i}} \pi_\theta(o_{k,i,t} \mid q_{k,i}, o_{k,i,<t}).$$

By context sufficiency (A3), conditioning on $x_{k,i}$ is equivalent to conditioning on $q_{k,i}$ for generating $o_{k,i}$, hence

$$\begin{aligned}&\pi_\theta(o_{k,i,t} \mid x_{k,i}, o_{k,i,<t})\\ &= \pi_\theta(o_{k,i,t} \mid q_{k,i}, o_{k,i,<t}), \qquad \forall k, t.\end{aligned}$$

Thus the MAS joint probability over all generated tokens equals (D.3). Deterministic insertion of workflow/tool tokens (A2) does not affect the probability mass on generated tokens, so the two induced joint distributions coincide. □

D.4 Reward Propagation and Token-Level Learning-Signal Equivalence

We now formalize the role of **reward propagation** in MHGPO and show that, together with parameter sharing, it establishes *token-level learning-signal equivalence* between the MAS execution and a single-LLM transcript rollout—not only for **score-function gradients**, but also for **GRPO-style objectives** with group-based advantages.

Reward broadcasting in MHGPO. In MHGPO, the terminal reward R_i is first propagated backward to all agents in the trajectory (Eq. (4)), and is further *broadcast to all generated tokens* within each agent output, exactly as in GRPO. Concretely, for

each trajectory i, every generated token $o_{k,i,t}$ is optimized with respect to the same scalar learning signal (up to normalization by group statistics). This design ensures that all token decisions—regardless of which agent produced them—are treated as contributing to the final system outcome.

D.4.1 Token-Level Equivalence for Score-Function Policy Gradients

We begin by proving token-level learning-signal equivalence under the naive score-function (REINFORCE) policy-gradient estimator.

Proposition 2 (Token-level learning-signal equivalence (score-function))**.** *Under Proposition 1, and assuming the terminal reward R_i is broadcast to all generated tokens in trajectory i, the score-function gradient computed from MASS rollouts equals that computed from the single-LLM transcript rollout:*

$$\nabla_\theta J(\theta) = \mathbb{E}\left[R_i \sum_{k=1}^{n}\sum_{t=1}^{T_{k,i}} \nabla_\theta \log \pi_\theta(o_{k,i,t} \mid x_{k,i}, o_{k,i,<t})\right].$$

Each summand corresponds to the same token under the same conditional distribution in both views.

Proof. By Proposition 1, the joint likelihood of the MASS rollout equals the likelihood of the single-context transcript. Broadcasting R_i to all generated tokens yields a score-function gradient that is a sum over per-token log-probabilities weighted by the same terminal reward. Since each token has the same conditional distribution in both views, the per-token learning signal is identical. □

D.4.2 Token-Level Equivalence for GRPO-style Policy Gradients

We now formalize the token-level equivalence of MHGPO-FoF under GRPO-style policy gradients. In FoF, each agent's advantage is computed over *heterogeneous groups*: for a fixed raw input, a group collects multiple generations of the same agent across different MAS trajectories originated from the same system input. With this construction, **parameter sharing + reward propagation + token-level reward broadcasting** yields an exact token-wise correspondence to standard GRPO on a single LLM, executed over a single-context transcript that interleaves fixed workflow/tool tokens with generated tokens.

Importance sampling ratio equivalence. For any generated token $o_{k,i,t}$, the token-level importance ratio in GRPO is

$$r_{k,i,t}(\theta) \triangleq \frac{\pi_\theta\big(o_{k,i,t} \mid x_{k,i}, o_{k,i,<t}\big)}{\pi_{\theta_{\text{old}}}\big(o_{k,i,t} \mid x_{k,i}, o_{k,i,<t}\big)}.$$

By Proposition 1, the conditional distribution of each token given its single-context prefix $x_{k,i}$ is the same under both the MAS rollout and the single-transcript execution. Hence the ratio $r_{k,i,t}(\theta)$ is identical in the two views.

Advantage equivalence. In MHGPO-FoF, grouping is by the *system input*: for a fixed query q, $\mathcal{G}(q)$ contains all rollouts started from q. This is *heterogeneous* at the agent level because each agent prompt $q_{k,i}$ is a deterministic function of rollout-specific upstream text. Let $R_{k,i}$ be the propagated reward for agent k in rollout i (Eq. (4)). MHGPO-FoF normalizes $\{R_{k,i}\}_{(k,i)\in\mathcal{G}(q)}$ to obtain a scalar advantage $A_{k,i}$, and broadcasts $A_{k,i}$ to every token in $o_{k,i}$.

By Proposition 1, each (k, i) corresponds to a transcript segment within a trajectory sampled under π_θ for the same q, so standard GRPO over transcript trajectories in $\mathcal{G}(q)$ uses the *same* grouping set. If only terminal rewards are used, all segments from trajectory i share the same advantage A_i, hence $A_{k,i} = A_i$ and the token-wise advantages match exactly. If agent-specific rewards are included, they map directly to step-wise rewards on the same transcript MDP, and the same correspondence holds.

Proposition 3 (Token-level GRPO equivalence for MHGPO-FoF)**.** *Under Assumptions (A1)–(A3), with reward propagation and token-level broadcasting as in MHGPO-FoF, the GRPO surrogate yields identical token-wise learning signals in the MAS view and the single-transcript view. In particular, for every generated token $o_{k,i,t}$,*

$$r^{MAS}_{k,i,t}(\theta) = r^{tr}_{k,i,t}(\theta), \qquad A^{MAS}_{k,i,t} = A^{tr}_{k,i,t} = A_i, \tag{8}$$

where A_i is the (broadcast) FoF group-relative advantage for trajectory i. Hence the clipped GRPO term is identical token-by-token in both views.

Proof. By Proposition 1, each token has the same conditional distribution given the same transcript prefix $x_{k,i}$ in both executions, so $r^{\text{MAS}}_{k,i,t}(\theta) = r^{\text{tr}}_{k,i,t}(\theta)$. FoF advantages coincide because both views (i) normalize the same propagated rewards

over the same FoF group and (ii) broadcast the resulting scalar, giving (8). Since clipping is deterministic in $(r_{k,i,t}(\theta), A_i)$, the per-token GRPO surrogate matches. □

D.5 Summary and key takeaway.

Under Assumptions (A1)–(A3), the token-level learning signal of MHGPO can be interpreted as standard GRPO applied to a *constructed single-context transcript* sampled from a single policy π_θ, where workflow/tool-call tokens are explicitly interleaved with generated tokens (akin to agentic traces in systems such as R1-Searcher). Concretely, reward back-propagation followed by GRPO-style broadcasting assigns each generated token a *group-relative* advantage target that matches the GRPO advantage computed on the corresponding transcript trajectories.

This equivalence is primarily a *conceptual* bridge rather than an implementation claim: MHGPO still performs *multi-task, multi-context* optimization, which empirically improves convergence and stability over naively applying single-context GRPO to long transcripts. In real MAS execution, agents often condition on step-specific prompts derived from tool outputs and context compressions (instead of a monolithic transcript), so MHGPO captures inter-agent dependencies *implicitly* through the workflow and shared parameters; Assumption (A3) (*Context Sufficiency*) precisely formalizes this gap, **yielding token-level alignment with GRPO when $q_{k,i}$ is sufficient, while still allowing MHGPO to exploit implicit inter-agent correlations when it is not**.

Algorithm 1: Fork-on-First Rollout Sampling

Input: Question q, rollout group size G, shared actor parameters θ
Output: Set of rollout trajectories $\{\mathcal{T}_i\}_{i=1}^{G}$
`// Determine the first agent to handle the question`
$A_{c_0} \leftarrow MAS.next_agent(q)$
`// Call fork_on to perform Fork-on-First rollout`
$\{\mathcal{T}_i\}_{i=1}^{G} \leftarrow \texttt{fork_on}(q, G, A_{c_0}, \theta)$
return $\{\mathcal{T}_i\}_{i=1}^{G}$

Algorithm 2: Independent Sampling

Input: Single question q, rollout group size G, shared actor parameters θ
Output: Merged set of rollout pairs $\mathcal{P}$ collected from each individual agent
Initialize $\mathcal{P} \leftarrow \emptyset$
for *each agent* A_i *where* $i = 1$ *to* n **do**
  `// Fork on agent` A_i
  Extract rollout pairs $\{(q_{k,j}, o_{k,j}, m_{k,j})\}$ from $\texttt{fork_on}(q, G, A_i, \theta)$
  `// Filter only rollout pairs where` $k = i$
  for *each rollout pair* $(q_{k,j}, o_{k,j}, m_{k,j})$ **do**
    if $k = i$ **then**
      Append $(q_{k,j}, o_{k,j}, m_{k,j})$ to $\mathcal{P}$
return final rollout pairs $\mathcal{P}$

Algorithm 3: Round-Robin Rollout Sampling

Input: Batch of questions $\mathcal{B} = \{q_1, q_2, \ldots, q_B\}$, rollout group size G, shared actor parameters θ, agent sampling probabilities $\{p_1, p_2, \ldots, p_n\}$
Output: Set of rollout trajectories $\{\mathcal{T}_i\}_{i=1}^{G'}$ with consistent group identifiers
Initialize rollout pair set $\mathcal{P}_{\text{all}} \leftarrow \emptyset$
for *each question* $q \in \mathcal{B}$ **do**
  `// Sample fork agent from categorical distribution`
  $A_i \sim \text{Categorical}(p_1, p_2, \ldots, p_n)$
  `// Fork-on-`A_i` to obtain` G `trajectories`
  $\{\mathcal{T}_j^{(q)}\}_{j=1}^{G} \leftarrow \texttt{fork_on}(q, G, A_i, \theta)$
  for *each trajectory* $\mathcal{T}_j^{(q)}$ **do**
    Extract all rollout pairs $(q_{k,j}, o_{k,j}, m_{k,j})$ from $\mathcal{T}_j^{(q)}$ and append to $\mathcal{P}_{\text{all}}$
`// Group reassignment for rollout pairs with group size` $= 1$
Count the number of occurrences for each group identifier m in $\mathcal{P}_{\text{all}}$
Let $\mathcal{P}_{\text{single}} \leftarrow$ rollout pairs whose original group m appears only once
Let $\mathcal{P}_{\text{valid}} \leftarrow$ rollout pairs whose group m appears more than once
Shuffle $\mathcal{P}_{\text{single}}$ randomly
Partition $\mathcal{P}_{\text{single}}$ into $N = \lfloor |\mathcal{P}_{\text{single}}|/G \rfloor$ disjoint groups $\{g_i\}_{i=1}^{N}$, where each $g_i = \{(q_{i,j}, o_{i,j})\}_{j=1}^{G}$
for *each group* g_i *where* $i = 1, \ldots, N$ **do**
  for *each rollout pair* $(q_{i,j}, o_{i,j})$ *where* $j = 1, \ldots, G$ **do**
    Assign new group identifier: $m_{i,j} \leftarrow i + |\mathcal{P}_{\text{valid}}|$
Merge $\mathcal{P}_{\text{valid}}$ with all reassigned g_i groups to form final $\mathcal{P}_{\text{all}}$
return final rollout pairs $\mathcal{P}_{\text{all}}$

Algorithm 4: Multi-Agent Optimization in MHGPO

Input: Training dataset $\mathcal{D}$ with question-answer pairs (q, a_{golden}), reward function $\mathcal{R}$, `batch_size`, `total_epochs`, `ppo_epochs`, rollout sampling function $\mathcal{H}$
Output: Optimized actor parameters θ
Initialize actor parameters θ (shared among all agents), reference parameters θ_{ref}
for *epoch* $= 1$ **to** `total_epochs` **do**
  for *each batch* $\mathcal{B} \subset \mathcal{D}$ **do**
    for *each question* q *in batch* $\mathcal{B}$ **do**
      `// Multi-Agent rollout trajectory sampling`
      Generate G rollout trajectories $\{\mathcal{T}_i\}_{i=1}^{G} \leftarrow \mathcal{H}(\theta, q)$
      Collect rollout pairs $\{(q_{k,j}, o_{k,j}, m_{k,j})\}$ for all agents across all trajectories
      `// Final reward calculation`
      for *each trajectory* $\mathcal{T}_i$ **do**
        Let o_i be the final output of $\mathcal{T}_i$
        Compute final reward: $R_i^{\text{shared}} = \mathcal{R}(o_i, a_{\text{golden}})$
      `// Backward propagation of rewards`
      for *each rollout pair* $(q_{k,j}, o_{k,j})$ **do**
        Calculate aggregated reward $R_{k,j}^{\text{shared}}$ according to Equation 4
        Apply agent-specific reward $R_{k,j}^{\text{spe}}$ to obtain $R_{k,j}$
    `// Actor update`
    for *ppo_epoch* $= 1$ **to** `ppo_epochs` **do**
      for *each rollout pair* $(q_{k,j}, o_{k,j}, m_{k,j}, R_{k,j})$ **do**
        Form rollout groups based on $m_{k,j}$ and estimate the advantage using Equation 5
      Update policy parameters θ using the policy gradient method with Equation 6 as the target

Algorithm 5: Base Rollout Sampling Function *fork_on()*

Input: Question q, rollout group size G, fork agent A_i, shared actor parameters θ
Output: Set of rollout trajectories $\{\mathcal{T}_i\}_{i=1}^{G}$
Initialize $o \leftarrow q$, empty prefix trajectory $\mathcal{T}_{\text{prefix}} \leftarrow \emptyset$
`// One-to-one rollout until fork agent` A_i `is encountered`
while $MAS.next_agent(o) \neq A_i$ **do**
  $A_{c_j} \leftarrow MAS.next_agent(o)$
  $q_{c_j} \leftarrow MAS.process_prompt(o, A_{c_j})$
  $o_{c_j} \leftarrow A_{c_j}(q_{c_j}; \theta)$
  Append (q_{c_j}, o_{c_j}) to $\mathcal{T}_{\text{prefix}}$
  $o \leftarrow o_{c_j}$
`// Fork-on-`A_i`: one-to-many rollout from agent` A_i
$q_{c_i} \leftarrow MAS.process_prompt(o, A_i)$
for $i = 1$ **to** G **do**
  $o_{c_i,i} \leftarrow A_i(q_{c_i}; \theta)$
  Initialize $\mathcal{T}_i \leftarrow \mathcal{T}_{\text{prefix}} \cup \{(q_{c_i}, o_{c_i,i})\}$
  $o \leftarrow o_{c_i,i}$
  `// Continue one-to-one rollout for each fork`
  $j \leftarrow 1$
  while $MAS.has_next_agent(o)$ **do**
    $A_{c_{i+j}} \leftarrow MAS.next_agent(o)$
    $q_{c_{i+j}} \leftarrow MAS.process_prompt(o, A_{c_{i+j}})$
    $o_{c_{i+j}} \leftarrow A_{c_{i+j}}(q_{c_{i+j}}; \theta)$
    Append $(q_{c_{i+j}}, o_{c_{i+j}})$ to $\mathcal{T}_i$
    $o \leftarrow o_{c_{i+j}}$
    $j \leftarrow j + 1$
`// Group identifier assignment`
for $i = 1$ **to** G **do**
  for *each rollout pair* $(q_{i,*}, o_{i,*})$ *in trajectory* $\mathcal{T}_i$ **do**
    Assign group identifier $m_{i,*} \leftarrow i$
return $\{\mathcal{T}_i\}_{i=1}^{G}$