Orch-RM: Reward Modeling for Multi-Agent Orchestration

King Yeung Tsang[*1] **Zihao Zhao**[*1,3] **Vishal Venkataramani**[1] **Haizhou Shi**[†1,2]

Zixuan Ke[†2] **Semih Yavuz**[2] **Shafiq Joty**[‡2,4] **Hao Wang**[‡1,3]

Code Dataset

Abstract

Multi-Agent Systems (MAS) built on Large Language Models (LLMs) require effective orchestration to coordinate specialized agents, yet training such orchestrators is hindered by limited supervision and high computational cost. We propose **Orch**estration **R**eward **M**odeling (Orch-RM), a framework for evaluating orchestration quality without new human annotations. Orch-RM leverages intermediate artifacts from multi-agent executions to construct win-lose pairs for Bradley-Terry reward model training. Once trained, Orch-RM operates directly at the orchestration level, enabling efficient and high-performing reward-guided orchestrator training and MAS test-time scaling. As shown in Fig. 1, Orch-RM improves training efficiency by up to 10× in token usage while improving MAS test-time scaling performance by up to 8% in accuracy. Improvements vary across benchmarks, with larger gains on AIME and BrowseComp+, while remaining competitive on HotpotQA and GPQA. The smaller gains on the latter two benchmarks may partly reflect their limited structural diversity in orchestration plans. These results highlight orchestration-level reward modeling as a promising direction for scalable and robust multi-agent orchestration.

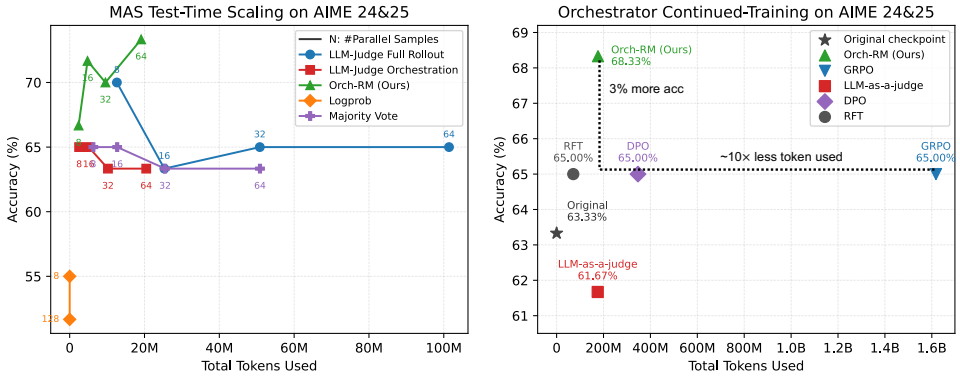

Figure 1: **Orch-RM for MAS Test-Time Scaling (Left) and Continued Orchestrator Training (Right). Left:** By operating directly at the orchestration level of MAS, Orch-RM achieves a stronger accuracy-efficiency trade-off for test-time scaling, reducing token usage while improving performance over baselines. **Right:** Using Orch-RM as the reward signal for continued MAS orchestrator training yields consistent performance gains with substantially lower token usage.

[*]Equal contribution. [†]Project co-lead. [‡]Equal advising. [1]Rutgers University. [2]Salesforce AI Research. [3]University of Illinois at Urbana-Champaign. [4]Nanyang Technological University. Correspondence to: King Yeung Tsang <kingyeung.tsang@gmail.com>, Haizhou Shi <haizhou.shi@rutgers.edu>, Zixuan Ke <zixuan.ke@salesforce.com>, Shafiq Joty <sjoty@salesforce.com>, Hao Wang <hwangml@illinois.edu>.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Multi-Agent Systems (MAS) built on Large Language Models (LLMs) have demonstrated strong potential for tackling complex tasks through the coordination of specialized agents [8]. By leveraging mechanisms such as task decomposition, evidence gathering, critique, and aggregation, these systems can produce outputs that are often more robust than those of a single agent. More recent advances have shifted from manually designed multi-agent workflows to automated *orchestration*, where an LLM-based controller dynamically plans the workflow structure and agent interactions required for a given task [9, 16, 31, 33]. This transition enhances both the flexibility and generalizability of MAS, enabling them to adapt their planning strategies across diverse domains, including previously unseen scenarios.

As naive off-the-shelf orchestrators often underperform in specialized domains [9, 16, 31], developing a capable LLM-based orchestrator has become increasingly important, yet remains highly non-trivial. This difficulty stems from several factors. First, the strong interdependencies among orchestration strategies, task types, and sub-agent capabilities make it challenging to collect high-quality human annotations for training. Second, the complexity of multi-agent workflows, coupled with the accumulated variance across full sub-agent rollouts, leads to substantial computational and time costs during training; for example, the recent MAS-Orchestra framework employs GRPO training, consuming over 1B tokens for only 100 update steps of the orchestrator [9].

To address the scarcity of high-quality orchestration annotations and the inefficiency of training such models, we propose **Orch**estration **R**eward **M**odeling (`Orch-RM`) for MAS. Rather than introducing a new orchestrator architecture or reinforcement learning objective, our goal is to investigate *the feasibility of leveraging reward models* to improve both training and test-time performance in *orchestration-based MAS systems*. In contrast to Reinforcement Learning via Human Feedback (RLHF) [18], which relies heavily on large-scale human preference annotations, `Orch-RM` learns from automatically constructed preference pairs to evaluate orchestration quality. Specifically, we collect intermediate artifacts, i.e., model checkpoints and generated trajectories, from MAS-Orchestra training and, through a multi-faceted grouping strategy, construct diverse win-lose pairs of orchestration trajectories to train a Bradley-Terry reward model [35]. The resulting reward model's ability to assess multi-agent orchestrations complements the known limitations of existing RMs and LLM-as-a-Judge methods when applied to MAS, both at the trajectory level [25] and at the orchestration level (Sec. 4.2 and 4.3).

We evaluate our approach on mathematical reasoning, web-based question answering, multi-hop reasoning, and out-of-distribution scientific reasoning. We integrate our learned `Orch-RM` into both *(i) inference through best-of-N (BoN) selection for efficient test-time scaling* and *(ii) reward-guided orchestrator training*. On the inference side, `Orch-RM` selects orchestration plans through best-of-N (BoN) scoring before sub-agent execution. Compared with majority voting, `Orch-RM` achieves clear gains on most evaluated benchmarks. On the training side, reward-guided continued training generally improves MAS-Orchestra [9] over the original checkpoint across the four evaluated benchmarks. Even training an orchestrator from scratch with reward-model feedback can approach the performance of the original teacher model. Together, these results suggest that our proposed orchestration reward modeling is a promising direction for both improving test-time selection and providing reusable supervision for automated multi-agent orchestration. Our main contributions are:

- We developed `Orch-RM`, a reward modeling framework for training LLM-based orchestrators.
- We integrated `Orch-RM` into orchestrator training, enabling reward-guided continued optimization that improves existing models and supports training from scratch to approach the performance of RL-trained counterparts.
- We applied `Orch-RM` at inference time, where it selects higher-quality rollouts than majority voting and enables more effective test-time scaling.

2 Related Work

LLM-based Multi-Agent Systems. While agentic capabilities exist on a spectrum, traditional Single-Agent Systems (SAS) operate via a monolithic control loop governed by a single LLM, even when utilizing tools or self-reflection [29, 15]. In contrast, Multi-Agent Systems (MAS)

distribute cognitive load across specialized sub-agents with distinct contexts and objectives [27]. By interacting through structured communication channels, MAS synthesize individual outputs into cohesive reasoning, inspiring diverse architectures from dynamic routing [24, 32] to skill-centric topologies [3].

The performance of MAS is fundamentally constrained by orchestration. Earlier methods relied on manual heuristics or inference-time routing [10, 33, 14], but recent literature has shifted toward *training-time orchestration*. MAS-Orchestra [9] pioneers training-time orchestration by formulating agent coordination as a holistic, function-calling reinforcement learning problem. As the first framework to generate complete multi-agent topologies in a single step, it achieves state-of-the-art, Pareto-optimal performance across diverse reasoning domains. Furthermore, by abstracting sub-agents as modular functions, it provides a universally representative and highly scalable baseline, making it the ideal foundational framework for our experiments. However, training this orchestrator relies on sparse, final-answer rewards. Consequently, the system must execute full multi-agent rollouts to receive any feedback, resulting in an immense computational overhead. `Orch-RM` addresses this by utilizing intermediate training artifacts to learn a reward model from automatically constructed preference pairs, enabling direct evaluation of orchestration plans without the prohibitive overhead of exhaustive sub-agent execution.

Reward Modeling for LLMs. Historically, Reward Modeling (RM) was primarily developed for Single-Agent Systems (SAS) to align LLMs through Reinforcement Learning from Human Feedback (RLHF) [18]. In this context, reward models are typically trained as discriminative scoring functions over human-annotated preference pairs, often utilizing a Bradley-Terry optimization objective as in PILAF [5]. However, directly applying these single-agent reward models to Multi-Agent Systems introduces severe credit assignment challenges, as interaction histories and prompts vary dynamically across distinct roles and conversation turns [36]. Crucially, as established in MAS-ProVe [25], traditional scalar and process reward models (PRMs) trained on single-agent paths struggle to reliably evaluate the out-of-distribution, conversational dynamics inherent to multi-agent orchestration.

To address this, recent literature has begun tailoring RM and reinforcement learning explicitly for collaborative MAS. For instance, MARTI [34] combines rule-based verifiable rewards with LLM-based generative rewards, and OrchMAS [6] trains its orchestration policy using a layered critique paradigm where the rewards are deterministically calculated based on structural completeness and reference-based precision. These frameworks are not applicable as baselines for our setting because they are policy optimization architectures designed to *consume* rewards, rather than methodologies for *training* a reward model. They rely on static external oracles, lacking the structural mechanisms to automatically construct preference pairs from distributed rollouts. In contrast, `Orch-RM` bridges this gap by natively extracting win-lose pairs from intermediate model checkpoints, optimizing a standalone reward model to provide the scalable evaluative signal required for training-time orchestration.

3 Methodology

This section presents our proposed orchestration reward modeling framework, `Orch-RM`, for improved test-time scaling and training efficiency for MAS orchestrators. Sec. 3.1 introduces the necessary preliminaries, discussing reward modeling for LLMs and its integration into test-time scaling and training of policy models. Sec. 3.2 provides an overview of `Orch-RM` as a complete pipeline including data construction, reward model training, and Sec. 3.3 discusses its application to test-time scaling of MAS and training of orchestrator models.

3.1 Preliminaries

Notation. Throughout this paper, scalars are denoted by lowercase letters (e.g., x), vectors or token/word sequences by bold lowercase letters (e.g., $\boldsymbol{x}$), random vectors by bold lowercase letters with a random-variable notation (e.g., $\mathbf{x}$), and matrices or collections of tokens, words, or phrases by bold uppercase letters (e.g., $\boldsymbol{X}$). We use $[m] = \{1, 2, \ldots, m\}$ to denote the set of consecutive integers from 1 to m. We follow the convention of using $\pi_0(\cdot)$, $\pi_{\boldsymbol{\theta}}(\cdot)$, and $\pi_{\boldsymbol{\Theta}}$ as the base model, being-trained policy model, and the herd of models in MAS, respectively.

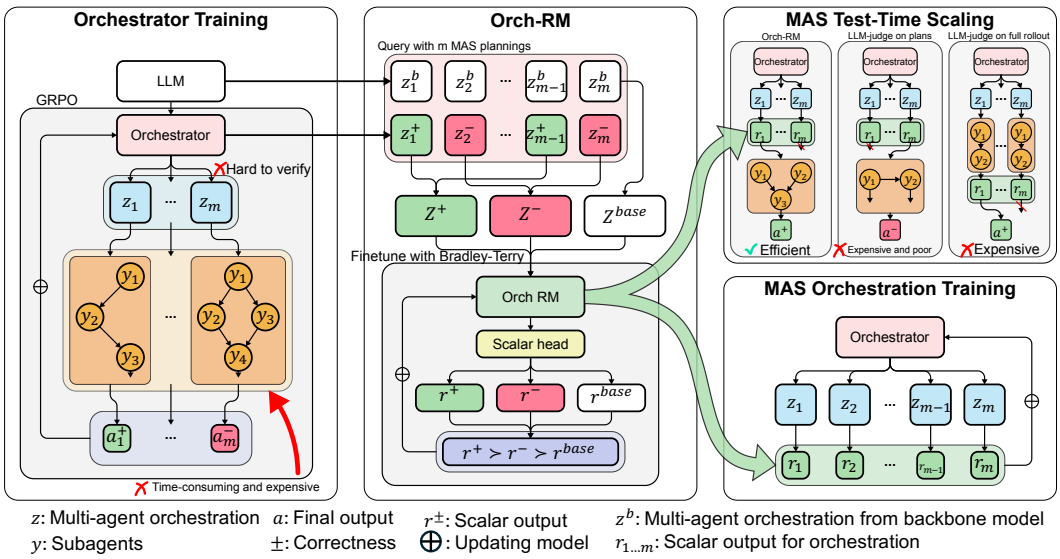

Figure 2: **Overview of Orchestration Reward Modeling (`Orch-RM`).** We construct preference pairs (*"specialized-over-base"* and *"correct-over-incorrect"*) by collecting intermediate artifacts from trained MAS orchestrator models [9], and optimize the reward model with a Bradley-Terry objective. Once trained, `Orch-RM` can be used for both test-time trajectory selection and continued orchestrator training, where it provides scalar rewards for candidate *orchestrations* without requiring *full sub-agent inference.*

Training RMs from Human Preferences. Given a prompt $\boldsymbol{x}$, candidate responses $(\boldsymbol{y}_w, \boldsymbol{y}_l)$ are generated and labeled by the base model π_0 and human annotators, where $(\boldsymbol{y}_w, \boldsymbol{y}_l)$ indicates the (preferred, dispreferred) pair. These preferences are assumed to arise from an underlying latent reward function $r^*(\boldsymbol{x}, \boldsymbol{y})$. The probability of $\boldsymbol{y}_1$ being better than $\boldsymbol{y}_2$ is typically modeled by the Bradley-Terry (BT) formulation [35]:

$$p^*(\boldsymbol{y}_1 \succ \boldsymbol{y}_2 \mid \boldsymbol{x}) \triangleq \sigma[r^*(\boldsymbol{x}, \boldsymbol{y}_1) - r^*(\boldsymbol{x}, \boldsymbol{y}_2)]. \tag{1}$$

To learn the transferrable capabilities of quality judge of LLM responses, a parametric reward model $r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y})$ is trained on a dataset of comparisons $\mathcal{D} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}_w^{(i)}, \boldsymbol{y}_l^{(i)}\}$ by maximizing likelihood, yielding the standard logistic loss:

$$\mathcal{L}_{\text{RM}}(\boldsymbol{\phi}) \triangleq -\mathbb{E}_{\mathcal{D}}\left[\log \sigma\left(r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}_w) - r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}_l)\right)\right]. \tag{2}$$

RMs for Improving Policy Models. The learned reward model $r_{\boldsymbol{\phi}}$ then provides scalar feedback for optimizing a policy $\pi_{\boldsymbol{\theta}}$. The typical objective balances reward maximization while staying close to a reference policy π_0 through KL regularization, optimized using reinforcement learning algorithms (such as PPO [22] and GRPO [23]):

$$\mathcal{L}_{\text{RLHF}}(\boldsymbol{\theta}) \triangleq -\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}, \boldsymbol{y}\sim\pi_\theta(\cdot|\boldsymbol{x})}\left[r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}) - \beta D_{\text{KL}}\left[\pi_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \,\|\, \pi_0(\boldsymbol{y} \mid \boldsymbol{x})\right]\right]. \tag{3}$$

RMs for Test-time Scaling. For tasks where the final answers can be effectively aggregated, e.g., math and short-form question answering, scaling up test-time compute by sampling multiple answers in parallel and then aggregating them has shown promise in significantly improving the model performance, e.g., majority voting [26]. An effective reward model can further improve this process by providing a learned scoring function over sampled responses.

- *Best-of-N (BoN).* BoN selects the single response with the highest reward:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}_n} r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}_n), \quad \boldsymbol{y}_n \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}), \tag{4}$$

  which replaces uniform aggregation with reward-based selection, allowing the system to identify higher-quality trajectories even when they are not the most frequent.

- *Weighted BoN (w-BoN).* Beyond selecting a single trajectory, w-BoN aggregates candidates at the answer level using reward-weighted voting. Let $\mathcal{Y}(\boldsymbol{a})$ denote the set of responses that map to answer $\boldsymbol{a}$. The induced answer distribution becomes

$$\pi_{\text{w-BoN}}(\boldsymbol{a} \mid \boldsymbol{x}) \propto \sum\nolimits_{\boldsymbol{y}_n \in \mathcal{Y}(\boldsymbol{a})} \exp\big(r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{y}_n)\big), \tag{5}$$

  which re-weights each sample according to its reward score. This formulation combines the diversity of multiple samples with the discriminative power of the reward model, often yielding more robust performance than both majority voting and standard BoN.

The following section will discuss the complications and challenges of integrating the idea of reward modeling for multi-agent orchestration.

3.2 `Orch-RM`: Reward Modeling for Multi-Agent Orchestration

Orchestration-Based LLM Multi-Agent Systems. Given an input query $\boldsymbol{x}$, an orchestration-based MAS first uses a policy model $\pi_{\boldsymbol{\theta}}$ to generate a multi-agent orchestration $\boldsymbol{z}$. The orchestration specifies the sub-agent roles, role descriptions, collaboration structure, and interaction graph among agents. It is then executed by the sub-agents, producing a sequence of intermediate responses $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_k\}$ following the workflow order. After all sub-agents complete inference, the system aggregates their outputs to produce a final answer $\boldsymbol{a}$. We denote a complete execution as a trajectory $\boldsymbol{\tau} = (\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{Y}, \boldsymbol{a})$. Each trajectory is assigned an outcome label based on the correctness of $\boldsymbol{a}$, which is later used to construct reward model training data.

Motivation and Overview. From the formulation of a MAS trajectory $\boldsymbol{\tau}$, we identify two key bottlenecks for effective MAS:

- *Performance bottleneck*, determined by the quality of the orchestration $\boldsymbol{z}$. Prior work shows that well-designed orchestration can substantially improve MAS effectiveness [9].
- *Efficiency bottleneck*, driven by the cost of executing sub-agent rollouts $\boldsymbol{Y}$, compounded by the fact that even an intermediate response $\boldsymbol{y}_1$ requires the execution of multiple agents. These executions are time-consuming, limiting the scalability of both inference and training. In particular, supervision for orchestration quality is costly, as it typically requires executing $\boldsymbol{z}$ to completion.

To address these challenges, we aim to learn a reward model $r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z})$ that assigns a scalar score to an orchestration $\boldsymbol{z}$, directly estimating its quality. Crucially, this model operates at the orchestration level rather than over full trajectories. This design offers two advantages: (i) at inference time, it enables early-stage intervention in MAS execution, leading to more token-efficient test-time scaling; (ii) during training, it provides dense feedback for the orchestrator without requiring expensive full trajectory rollouts. Fig. 2 illustrates the training pipeline of `Orch-RM` and its downstream applications, which we describe in the following sections.

Data Construction. To the best of our knowledge, there is no existing labeled dataset exclusively for MAS orchestration. As discussed earlier, collecting such data faces two main challenges: (1) Human annotations may be misaligned with machine objectives, as humans often emphasize readability or interpretability rather than execution effectiveness, introducing noise. (2) Labeling orchestration quality is expensive, as it typically requires executing full MAS trajectories.

To address these challenges, we leverage an off-the-shelf trained MAS orchestrator and the intermediate artifacts generated during its training [9]. Specifically, we construct two sources of orchestration comparisons for training `Orch-RM`:

- *Specialized over base* ($\pi_{\boldsymbol{\theta}} \succ \pi_0$): we compare trajectories generated by the trained orchestrator $\pi_{\boldsymbol{\theta}}$ against those from the backbone model π_0, where the former generally produces higher-quality orchestrations due to task-specific optimization [2].
- *Correct over incorrect* ($\boldsymbol{z}^+ \succ \boldsymbol{z}^-$): for each input, we use sampled trajectories from $\pi_{\boldsymbol{\theta}}$ and label them based on the correctness of their final outputs, treating correct trajectories ($\boldsymbol{\tau}^+$) as preferred over incorrect ones ($\boldsymbol{\tau}^-$). Notably, this source is constructed entirely from accessible rollout artifacts *without additional annotation* [3].

[2] Repeated-execution win rates for SoB range from 94.0% to 96.5% across three datasets (see Appendix C.1)

[3] Repeated-execution win rates for CoI range from 67.5% to 72.5% across three datasets (see Appendix C.1).

We combine these two sources of pairwise data using a fixed SoB:CoI mixing ratio of 1:3 to train the reward model. For an ablation study on the effect of data mixing ratios in `Orch-RM` training, please refer to Sec. 4.2.

Training Objective. We train $r_{\boldsymbol{\phi}}$ to assign higher scores to higher-quality orchestration than to lower-quality orchestration, using the standard Bradley-Terry modeling as detailed in Sec. 3.1:

$$\mathcal{L}_{\texttt{Orch-RM}}(\boldsymbol{\phi}) \triangleq -\mathbb{E}_{\mathcal{D}}\left[\log \sigma\big(r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z}_w) - r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z}_l)\big)\right]. \tag{6}$$

3.3 Applying `Orch-RM` to MAS: Efficient Test-Time Scaling and Orchestrator Training

MAS Test-Time Scaling. We perform test-time scaling by sampling multiple orchestrations from the orchestrator policy $\pi_{\boldsymbol{\theta}}$ and using the reward model $r_{\boldsymbol{\phi}}$ to score them. Specifically, given an input $\boldsymbol{x}$, we sample a set of orchestrations $\boldsymbol{Z} \triangleq \{\boldsymbol{z}_n\}_{n\in[N]}$ of size N in parallel and select the highest-scoring orchestration $\widehat{\boldsymbol{z}}$ with our trained orchestration reward model $r_{\boldsymbol{\phi}}$:

$$\widehat{\boldsymbol{z}} = \arg\max_{\boldsymbol{z}\in\boldsymbol{Z}} \{r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z})\}. \tag{7}$$

The selected best orchestration is then used for the full MAS trajectory generation $\boldsymbol{Y} \sim \pi_{\boldsymbol{\Theta}}(\cdot \mid \boldsymbol{x}, \widehat{\boldsymbol{z}})$.

Unlike typical MAS test-time scaling [25] that repeatedly executes diverse sub-agent behaviors in parallel, our `Orch-RM` operates at the *orchestration level*, directly *evaluating high-level plans before any sub-agent execution*. This avoids expensive, redundant sub-agent rollouts at inference time and enables much more efficient scaling, as the reward model captures the quality of both the orchestration strategy and the induced sub-agent behaviors.

Orchestrator Training. Beyond inference, we integrate $r_{\boldsymbol{\phi}}$ into the training of the orchestration policy. We consider both continued training from a pretrained $\pi_{\boldsymbol{\theta}}$ and training from scratch initialized from the backbone π_0. In both settings, the reward model replaces the original task-level reward by assigning a scalar score to each orchestration. For a group of sampled orchestrations $\{\boldsymbol{z}_n\}_{n\in[N]}$, we compute normalized advantages with

$$A_n = r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z}_n) - \tfrac{1}{N}\sum\nolimits_{j=1}^{N} r_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{z}_j), \tag{8}$$

and optimize the policy using the GRPO objective. Importantly, this formulation provides supervision at the orchestration level, allowing the model to implicitly learn better coordination of sub-agent behaviors; it *avoids expensive sub-agent execution (rollouts)* and does not require explicit annotations for each step. As a result, training becomes more efficient and scalable, while still improving overall system performance.

4 Experiments

We evaluate `Orch-RM` through extensive experiments. Sec. 4.1 presents the experimental setup. In Sec. 4.2, we first evaluate the application of `Orch-RM` to orchestration-level MAS test-time scaling, followed by an ablation study on the data mixing ratios used for reward model training. Finally, Sec. 4.3 studies the effectiveness of using `Orch-RM` as the reward signal for MAS orchestrator training, under both training-from-scratch and continued-training settings.

4.1 Experimental Setup

Models. We follow the setup of MAS-Orchestra [9], using `Qwen2.5-7B-Instruct` [19] as the orchestration policy $\pi_{\boldsymbol{\theta}}$ and `GPT-OSS-120B` [1] as the backbone model for all sub-agent calls during MAS trajectory generation. For orchestration-level reward modeling, we initialize from `Skywork-Reward-Llama-3.1-8B` [12] and fine-tune it on the constructed preference data.

Benchmarks. We base our evaluation on three main benchmarks widely adopted in prior works [9]: **AIME 24&25** [2] for challenging mathematical reasoning with moderate MAS requirements, **BrowseComp+** [4] for web-based QA requiring high-level planning and precise tool use, and **HotpotQA** [28] for multi-hop reasoning. To assess cross-domain robustness, we additionally report out-of-distribution results on **GPQA** [21] for scientific reasoning. Further dataset details are provided in Appendix B.2

Table 1: **Performance of Orchestration-Level Automatic Verification Methods for MAS Test-Time Scaling.** We compare `Orch-RM` primarily against orchestration-level verification methods (bottom two blocks). Trajectory-level baselines are included for reference, including MAS-Orchestra [9] without verification as a lower bound (LB), and an oracle reward model that always selects the best answer as an upper bound (UB). **Acc.:** Best-of-N accuracy (N=8 across all benchmarks), reported in percentages. **#Tok (M):** Total number of tokens (input + output) used for verification. **Boldface** and underline indicate the best and the second best results among orchestration-level methods.

Method	Verification Level	AIME 24&25		BrowseComp+		HotpotQA		GPQA	
		Acc.	#Tok (M)	Acc.	#Tok (M)	Acc.	#Tok (M)	Acc.	#Tok (M)
Baseline									
MAS-Orchestra [9] (LB)	-	63.33	-	9.50	-	42.50	-	63.63	-
Maj. Vote [26]	Trajectory	63.33	-	9.50	-	42.50	-	63.63	-
`GPT-5-mini` [7]	Trajectory	70.00	12.62	12.50	142.80	44.00	14.86	68.89	29.76
Oracle (UB)	Trajectory	76.67	-	24.50	-	57.00	-	84.85	-
LLM-as-a-Judge									
`GPT-4.1-mini` [17]	Orchestration	58.33	2.52	9.00	<u>8.63</u>	42.00	<u>7.66</u>	67.68	7.85
`GPT-5-mini` [7]	Orchestration	65.00	2.53	10.50	8.66	40.50	7.69	**71.72**	7.88
`GPT-5.2` [7]	Orchestration	65.00	<u>2.40</u>	10.00	8.75	42.00	7.77	68.02	7.89
`GPT-5.4-mini` [7]	Orchestration	<u>66.67</u>	2.53	11.00	8.72	**46.00**	7.75	68.13	<u>7.83</u>
Reward Model									
$\log P$ [11]	Orchestration	55.00	-	10.50	-	39.50	-	<u>71.21</u>	-
`Skywork-Llama-3.1-8B` [12]	Orchestration	61.67	**2.38**	11.50	**8.26**	40.50	**7.32**	64.67	7.49
`Skywork-V2-Llama-3.2-3B` [13]	Orchestration	58.33	**2.38**	<u>12.00</u>	**8.26**	41.50	**7.32**	64.65	7.49
`Skywork-V2-Qwen3-4B` [13]	Orchestration	61.67	**2.38**	8.00	**8.26**	42.00	**7.32**	68.69	7.49
`Skywork-V2-Qwen3-8B` [13]	Orchestration	60.00	**2.38**	9.00	**8.26**	41.00	**7.32**	68.18	7.49
`Orch-RM` (Ours)	Orchestration	**68.33**	**2.38**	**14.00**	**8.26**	<u>42.50</u>	**7.32**	66.16	**7.49**

Baselines. We compare `Orch-RM` against strong baselines under the same evaluation protocol (see Appendix B.1 for implementation details).

- *For test-time scaling:* we include state-of-the-art automatic verification methods for LLMs, applied at both the orchestration level and the full MAS trajectory level when applicable. These include (i) **general-purpose LLM-as-a-Judge** [37] methods, with base models `GPT-4.1-mini`, `GPT-5-mini`, `GPT-5.2`, and `GPT-5.4-mini` [7]; (ii) **off-the-shelf reward signals**, including log-probability as internal confidence ($\log P$) [11] and Skywork-Reward models (`Llama-3.1-8B` [12]; `V2-Llama-3.2-3B`, `V2-Qwen3-4B`, and `V2-Qwen3-8B` [13]); (iii) **test-time scaling baselines**, including no reward model, oracle selection (Pass@N), and Majority Vote [26].
- *For orchestrator training:* we include standard post-training methods for LLMs, including Rejection Sampling Fine-Tuning (**RFT**) [30] and continued reinforcement learning with Group Relative Policy Optimization (**GRPO**) [23] applied at the full MAS trajectory level using final correctness as reward [9]. We also include orchestration-level training with **LLM-as-a-Judge** [37] as the reward signal (with `GPT-5-mini` as the judge model), and Direct Preference Optimization (**DPO**) [20] trained on the same static dataset as our `Orch-RM`.

Implementation Details. We strictly follow the setup of MAS-Orchestra [9] and train domain-specific reward models using corresponding data and off-the-shelf orchestrator models. For detailed dataset description, please refer to Appendix B.2.

- For *"specialized-over-base"* pairs, we sample 10 *orchestrations* per query from both the orchestration policy $\pi_{\boldsymbol{\theta}}$ and the base policy π_0;
- For *"correct-over-incorrect"* pairs, we collect *full trajectories* from MAS-Orchestra training logs (up to the final step)[4] and pair orchestrations that lead to correct versus incorrect outcomes.

We combine these two types of pairwise data at a Specialized-over-Base (SoB) to Correct-over-Incorrect (CoI) ratio of 1:3 as it provides the best generalization ability for the model, yielding 4 *"specialized-over-base"* pairs and 12 *"correct-over-incorrect"* pairs per query. Ablation results on the effect of this mixing ratio are provided in Sec. 4.2.

[4]We follow the setting of MAS-Orchestra [9] as it provides the model checkpoints and full training log which are available at `https://github.com/SalesforceAIResearch/MAS-Orchestra`.

Table 2: **Ablation Study on Data Mixing Ratios for `Orch-RM` Training.** The evaluation setup follows Table 1. We consider three sources of pairwise data for orchestration reward model training: (i) Specialized-over-Base (*SoB*), (ii) Specialized-over-Other-domains (*SoO*), and (iii) Correct-over-Incorrect (*CoI*). **Avg. Adv.:** average normalized advantage between orchestrations leading to correct and incorrect answers. **(a):** untrained `Skywork-Reward-Llama-3.1-8B` baseline; **(b)-(d):** alternative data mixing ratios for `Orch-RM`; **(e):** default setting of `Orch-RM`; **(f):** oracle reward model upper bound. **Boldface** denotes the best performance.

	Method	Verification Level	Data Mix Ratio			AIME 24&25		BrowseComp+		HotpotQA	
			SoB	*SoO*	*CoI*	*Acc.*	*Avg. Adv.*	*Acc.*	*Avg. Adv.*	*Acc.*	*Avg. Adv.*
(a)	`Orch-RM`	Orchestration	-	-	-	61.67	**0.32**	11.50	0.12	40.50	**0.03**
(b)			3	2	0	63.33	0.18	**14.00**	0.18	41.00	-0.05
(c)			0	0	1	63.33	0.20	10.50	0.22	42.00	-0.04
(d)			1	0	1	63.33	0.30	11.00	**0.32**	**42.50**	-0.03
(e)			1	0	3	**68.33**	0.25	**14.00**	0.31	**42.50**	-0.03
(f)	Oracle (UB)	Trajectory	-	-	-	76.67	-	24.50	-	57.00	-

4.2 `Orch-RM` for MAS Test-Time Scaling

Table 1 reports the performance of `Orch-RM` for MAS test-time scaling against other verification baselines across four datasets. Table 2 further analyzes the effect of different data mixing ratios used in `Orch-RM` training. Table 3 shows the correlation between `Orch-RM` scores and empirical orchestration success rates estimated from repeated executions of fixed plans.

`Orch-RM` improves the accuracy-efficiency trade-off for MAS test-time scaling. As shown in Table 1, `Orch-RM` achieves the strongest overall accuracy-efficiency trade-off among orchestration-level verification methods. Compared to the self-consistency baseline, `Orch-RM` improves Best-of-N accuracy from 63.33% to 68.33% on AIME 24&25 and from 9.50% to 14.00% on BrowseComp+, while using only 2.38M and 8.26M verification tokens, respectively. `Orch-RM` also outperforms all orchestration-level LLM-as-a-Judge and off-the-shelf reward model baselines on these two benchmarks. In particular, on BrowseComp+, `Orch-RM` surpasses the strongest trajectory-level LLM judge (12.50% by GPT-5-mini) while reducing verification cost from 142.80M to 8.26M tokens. Additionally, as shown in Fig. 1, `Orch-RM` continues to benefit as the parallel sampling size scales up to 64 on AIME 24&25, demonstrating the scalability of its reward signals for large-scale test-time sampling.

`Orch-RM` benefits from combining multiple sources of orchestration preference data. Table 2 shows that the data mixing ratio substantially affects reward-guided test-time scaling performance. Using only Correct-over-Incorrect (*CoI*) pairs or only domain-level comparisons leads to weaker results than the default mixed setting. In contrast, the default configuration in row (e), which combines Specialized-over-Base (*SoB*) and Correct-over-Incorrect (*CoI*) pairs with a 1:3 ratio, achieves the best overall performance, reaching 68.33% accuracy on AIME 24&25 and 14.00% on BrowseComp+. This setting also produces strong average normalized advantages (0.25 on AIME and 0.31 on BrowseComp+), indicating improved separation between correct and incorrect orchestrations.

`Orch-RM` scores correlate with orchestration reliability. We operationalize orchestration quality as the probability that executing a plan produces a correct final answer. To estimate this quantity beyond a single execution outcome, we fix each orchestration plan and repeat its sub-agent execution five times. We use the proportion of correct executions as an empirical estimate of the plan's expected accuracy, and compute Spearman's rank correlation between these estimates and the scores assigned by `Orch-RM`. As shown in Table 3, the correlations are positive across all three benchmarks: 0.724 on AIME 24&25, 0.597 on BrowseComp+, and 0.546 on HotpotQA. These results suggest that `Orch-RM` assigns higher scores to plans that succeed more reliably across stochastic executions, rather than merely reproducing the outcome of a single rollout. This analysis supports alignment with execution reliability, although it does not disentangle orchestration quality from answer correctness, which remains our operational measure.

Table 3: Spearman correlation between Orch-RM scores and empirical plan success rates, estimated from five executions of each fixed orchestration plan.

Benchmark	Spearman correlation
AIME 24&25	0.724
BrowseComp+	0.597
HotpotQA	0.546

Table 5: **Performance of `Orch-RM` for MAS Orchestrator Training.** Values are mean ± standard deviation over 10 bootstrap resamples of evaluation questions. **(a):** base orchestrator without additional training; **(b)–(c):** training-from-scratch methods initialized from the base model; **(d)–(h):** continued-training methods initialized from the off-the-shelf MAS-Orchestra model. **Boldface** and underline denote the best and second-best mean performance in each column, respectively. Tied means receive the same marking.

	Method	Sampling Level	AIME 24&25		BrowseComp+		HotpotQA		GPQA	
			Acc.	Maj. Vote	Acc.	Maj. Vote	Acc.	Maj. Vote	Acc.	Maj. Vote
(a)	`Qwen2.5-7B-Instruct` [19]	-	30.08 ± 1.09	23.83 ± 0.81	2.96 ± 0.45	0.35 ± 0.47	31.48 ± 1.01	34.65 ± 1.25	24.92 ± 0.32	19.39 ± 0.54
(b)	+ `Orch-RM` (Ours)	Orchestration	64.15 ± 0.80	62.67 ± 0.86	6.04 ± 0.48	4.40 ± 0.52	40.44 ± 1.04	41.85 ± 1.38	65.20 ± 0.47	63.38 ± 1.02
(c)	+ MAS-Orchestra [9]	Trajectory	59.92 ± 0.63	62.50 ± 1.18	$\underline{8.93 \pm 0.51}$	$\underline{8.90 \pm 0.70}$	40.43 ± 0.95	41.80 ± 1.23	$\mathbf{66.16 \pm 0.70}$	64.34 ± 0.26
(d)	+ DPO [20]	-	63.98 ± 1.28	$\underline{64.83 \pm 1.46}$	8.64 ± 0.31	7.35 ± 0.34	40.10 ± 1.14	40.10 ± 1.10	64.54 ± 0.39	64.49 ± 0.24
(e)	+ RFT [30]	Trajectory	$\underline{64.33 \pm 0.84}$	63.83 ± 0.81	6.91 ± 0.59	5.50 ± 0.53	39.07 ± 0.73	41.05 ± 0.76	64.75 ± 0.36	64.14 ± 0.71
(f)	+ GRPO [23]	Trajectory	63.78 ± 1.06	$\underline{64.83 \pm 0.53}$	6.48 ± 0.38	5.70 ± 0.54	$\underline{40.95 \pm 1.01}$	$\mathbf{43.28 \pm 1.22}$	65.58 ± 0.66	$\underline{65.25 \pm 0.40}$
(g)	+ LLM-as-a-Judge [37]	Orchestration	59.13 ± 1.39	61.17 ± 0.81	8.53 ± 0.39	8.13 ± 0.28	39.69 ± 0.81	42.00 ± 0.91	61.48 ± 0.44	61.26 ± 0.54
(h)	+ `Orch-RM` (Ours)	Orchestration	$\mathbf{67.72 \pm 1.44}$	$\mathbf{69.83 \pm 0.53}$	$\mathbf{9.75 \pm 0.51}$	$\mathbf{10.90 \pm 0.70}$	$\mathbf{42.08 \pm 0.85}$	$\underline{42.10 \pm 0.99}$	$\underline{66.07 \pm 0.90}$	$\mathbf{66.11 \pm 0.73}$

In both Table 1 and Table 2, the gains are smaller on HotpotQA and GPQA, where our diversity analysis (Appendix C.2) reveals highly homogeneous orchestration structures on these benchmarks, which may partly explain the limited benefits of orchestration-level verification (see Appendix D for qualitative examples). In particular, on HotpotQA, all settings achieve similar accuracy (40.50-42.50%) and slightly negative normalized advantages, suggesting that many trajectories share similar orchestration structures and are therefore difficult to distinguish at the orchestration level. Nevertheless, `Orch-RM` remains competitive while maintaining substantially lower verification cost than trajectory-level methods.[5] Overall, these results suggest that combining correctness-based and domain-level orchestration comparisons provides the most effective supervision signal for `Orch-RM`.

4.3 `Orch-RM` for MAS Orchestrator Training

Table 5 reports the performance of using `Orch-RM` as the orchestration-level reward signal for MAS orchestrator training under two settings: *(i) training from scratch* and *(ii) continued training*. Fig. 3 further illustrates the performance-efficiency trade-off of different post-training methods during continued training of the MAS orchestrator. We additionally compare the estimated computational costs of continued training in Table 4, accounting for upfront reward-model training and subsequent orchestrator optimization.

`Orch-RM` enables efficient MAS orchestrator training from scratch. As shown in Table 5, training an orchestrator from scratch with `Orch-RM` substantially improves over the untrained base model across all benchmarks, increasing majority-vote accuracy from 23.83% to 62.67% on AIME 24&25 and from 0.35% to 4.40% on BrowseComp+. Despite relying only on orchestration-level supervision, `Orch-RM` achieves performance competitive with trajectory-level MAS-Orchestra training.

`Orch-RM` significantly improves the performance-efficiency trade-off for continued orchestrator training. Under continued training, `Orch-RM` achieves the best overall performance on AIME 24&25 and BrowseComp+, reaching 69.83% and 10.90% majority-vote accuracy, respectively. Crucially, *`Orch-RM` is the only method that consistently improves performance under the continued training setting,* demonstrating the reliability of its reward signals. As shown in Fig. 3, `Orch-RM` consistently provides a stronger performance-efficiency trade-off than trajectory-level post-training methods such as RFT and GRPO. In particular, on BrowseComp+, `Orch-RM` improves majority-vote accuracy from 8.90% to 10.90% while using over $10\times$ fewer training tokens than GRPO. These results highlight the efficiency advantage of orchestration-level reward supervision, which avoids expensive full-trajectory optimization over sub-agent rollouts.

Table 4: Estimated computational cost of continued orchestrator training at $N = 8$.Ratios are total GRPO FLOPs over total `Orch-RM` FLOPs, including upfront reward-model training.

Training dataset	FLOPs ratio (GRPO / `Orch-RM`)
DeepScaleR	$9.3\times$
BrowseComp+	$64.7\times$
HotpotQA	$3.2\times$

`Orch-RM` reduces computational cost even after accounting for reward-model training. To complement the token-based comparison in Fig. 3, we estimate the computational cost of continued

[5] For plans generated by GPT-5-mini rather than MAS-Orchestra, `Orch-RM` matches or exceeds the LLM-as-a-Judge baseline across three benchmarks when scoring their XML representations. See Appendix C.5 for details.

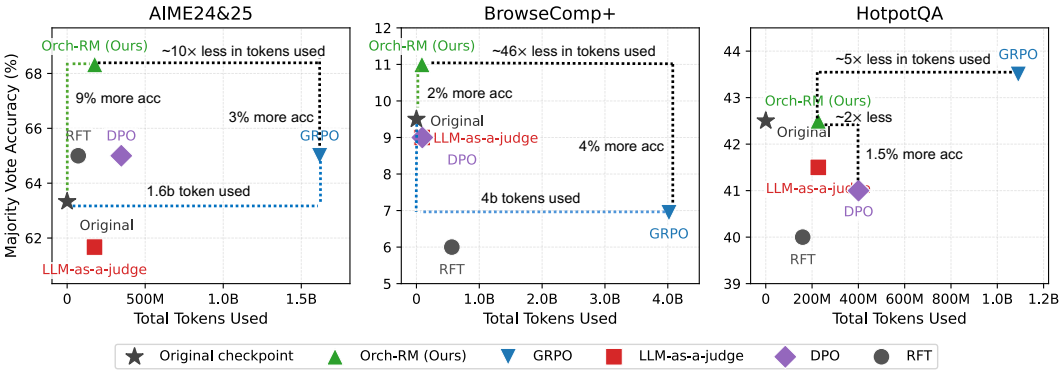

Figure 3: **Continued Training of MAS Orchestrators with Different Reward Signals.** We compare the performance-efficiency trade-off of different post-training methods by reporting downstream majority-vote accuracy and the total number of training tokens consumed. Across AIME 24&25 and BrowseComp+, `Orch-RM` achieves the strongest overall trade-off, consistently improving over the original MAS-Orchestra [9] checkpoint while using substantially fewer training tokens than trajectory-level reinforcement learning methods such as GRPO. In particular, `Orch-RM` attains comparable or stronger performance using up to $\sim$10$\times$ fewer tokens on AIME 24&25 and $\sim$46$\times$ fewer tokens on BrowseComp+.

orchestrator training, including upfront reward-model training, orchestrator sampling and optimization, and reward computation. Under the matched $N = 8$ setting, Table 4 shows that GRPO requires approximately 9.3, 64.7, and 3.2 times the FLOPs of `Orch-RM` on DeepScaleR, BrowseComp+, and HotpotQA, respectively. The largest reduction occurs on BrowseComp+, where full sub-agent execution is particularly costly. These estimates indicate that the computational benefits persist after accounting for reward-model training. Detailed cost breakdowns and estimation assumptions are provided in Appendix C.3.

5 Conclusion

In this work, we propose `Orch-RM`, an orchestration reward modeling framework for multi-agent LLM systems. By learning orchestration-level reward signals from accessible training artifacts and pairwise orchestration comparisons, `Orch-RM` avoids costly human annotations and expensive full-trajectory supervision over sub-agent rollouts during orchestrator inference and training. We show that `Orch-RM` can be effectively applied to both MAS test-time scaling and orchestrator training, consistently improving the performance-efficiency trade-off while operating entirely at the orchestration level. Experiments across mathematical reasoning, web-based QA, multi-hop reasoning, and scientific reasoning suggest that orchestration-level reward modeling is a promising future direction for improving automated MAS orchestration.

Limitations. `Orch-RM` has several limitations. First, although it avoids expensive human annotations by leveraging existing MAS orchestration artifacts, its performance is still constrained by the diversity and quality of available orchestrator models and sampled trajectories. Constructing larger-scale and more diverse orchestration datasets remains an important challenge. Second, `Orch-RM` is currently trained in a domain-specific manner, limiting its generalizability. Whether orchestration reward models can benefit from multi-domain and multi-task training remains underexplored.

Acknowledgments and Disclosure of Funding

We thank all reviewers, the area chair, and the senior area chair for their valuable comments.

Funding. HW is supported by the Amazon Faculty Research Award, Microsoft AI & Society Fellowship, Nokia Faculty Research Award, NSF CAREER Award IIS-2340125, NIH grant R01CA297832, and NSF grant IIS-2127918.

Competing interests. The authors declare no competing interests.

References

[1] S. Agarwal, L. Ahmad, J. Ai, S. Altman, A. Applebaum, E. Arbus, R. K. Arora, Y. Bai, B. Baker, H. Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

[2] AIME. American Invitational Mathematics Examination. Mathematical Association of America, 2025. `https://maa.org/maa-invitational-competitions/`.

[3] Anthropic. Equipping agents for the real world with agent skills. `https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills`, 2024.

[4] Z. Chen, X. Ma, S. Zhuang, P. Nie, K. Zou, A. Liu, J. Green, K. Patel, R. Meng, M. Su, S. Sharifymoghaddam, Y. Li, H. Hong, X. Shi, X. Liu, N. Thakur, C. Zhang, L. Gao, W. Chen, and J. Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

[5] Y. Feng, A. Kwiatkowski, K. Zheng, J. Kempe, and Y. Duan. PILAF: Optimal human preference sampling for reward modeling. In *Forty-second International Conference on Machine Learning*, 2025.

[6] Y. Feng, H. Luo, Z. Lin, Y. Sun, P. Wei, L. B. Hsieh, and A. T. Luu. Orchmas: Orchestrated reasoning with multi collaborative heterogeneous scientific expert structured agents, 2026.

[7] GPT-5Team. Openai gpt-5 system card, 2026.

[8] Z. Ke, F. Jiao, Y. Ming, X.-P. Nguyen, A. Xu, D. X. Long, M. Li, C. Qin, P. Wang, S. Savarese, C. Xiong, and S. Joty. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *TMLR*, 2025.

[9] Z. Ke, Y. Ming, A. Xu, R. Chin, X.-P. Nguyen, P. Jwalapuram, J. Wang, S. Yavuz, C. Xiong, and S. Joty. Mas-orchestra: Understanding and improving multi-agent reasoning through holistic orchestration and controlled benchmarks. *Forty-third International Conference on Machine Learning*, 2026.

[10] Z. Ke, A. Xu, Y. Ming, X.-P. Nguyen, C. Xiong, and S. Joty. MAS-ZERO: Designing multi-agent systems with zero supervision. *SEA@NeurIPS*, 2025.

[11] A. Kumar, R. Morabito, S. Umbet, J. Kabbara, and A. Emami. Confidence under the hood: An investigation into the confidence-probability alignment in large language models. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 315–334, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics.

[12] C. Y. Liu, L. Zeng, J. Liu, R. Yan, J. He, C. Wang, S. Yan, Y. Liu, and Y. Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.

[13] C. Y. Liu, L. Zeng, Y. Xiao, J. He, J. Liu, C. Wang, R. Yan, W. Shen, F. Zhang, J. Xu, et al. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. *arXiv preprint arXiv:2507.01352*, 2025.

[14] Z. Liu, Y. Zhang, P. Li, Y. Liu, and D. Yang. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024.

[15] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[16] B. Niu, Y. Song, K. Lian, Y. Shen, Y. Yao, K. Zhang, and T. Liu. Flow: Modularized agentic workflow automation, 2025.

[17] OpenAI. Openai gpt-4.1 nano model, 2025. Released April 14, 2025 via API; 1 million token context window; fastest & cheapest version in GPT-4.1 family.

[18] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Gray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[19] T. Qwen. Qwen2.5: A party of foundation models, September 2024.

[20] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[21] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. Gpqa: A graduate-level google-proof qa benchmark, 2023.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[23] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[24] H. Su, S. Diao, X. Lu, M. Liu, J. Xu, X. Dong, Y. Fu, P. Belcak, H. Ye, H. Yin, Y. Dong, E. Bakhturina, T. Yu, Y. Choi, J. Kautz, and P. Molchanov. Toolorchestra: Elevating intelligence via efficient model and tool orchestration, 2025.

[25] V. Venkataramani, H. Shi, Z. Ke, A. Xu, X. He, Y. Zhou, S. Yavuz, H. Wang, and S. Joty. Masprove: Understanding the process verification of multi-agent systems. *Forty-third International Conference on Machine Learning*, 2026.

[26] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[27] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Cheng, Q. Zhang, W. Qin, Y. Zheng, X. Qiu, X. Huang, and T. Gui. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[28] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[29] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[30] Z. Yuan, H. Yuan, C. Li, G. Dong, C. Tan, and C. Zhou. Scaling relationship on learning mathematical reasoning with large language models. *CoRR*, abs/2308.01825, 2023.

[31] L. Yue, K. R. Bhandari, C.-Y. Ko, D. Patel, S. Lin, N. Zhou, J. Gao, P.-Y. Chen, and S. Pan. From static templates to dynamic runtime graphs: A survey of workflow optimization for llm agents, 2026.

[32] H. Zhang, T. Feng, and J. You. Router-r1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[33] J. Zhang, J. Xiang, Z. Yu, F. Teng, X.-H. Chen, J. Chen, M. Zhuge, X. Cheng, S. Hong, J. Wang, B. Zheng, B. Liu, Y. Luo, and C. Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[34] K. Zhang, K. Tian, R. Liu, S. Zeng, X. Zhu, G. Jia, Y. Fan, X. Lv, Y. Zuo, C. Jiang, Y. wang, J. Wang, E. Hua, X. Long, J. Gao, Y. Sun, Z. Ma, G. Cui, N. Ding, B. Qi, and B. Zhou. MARTI: A framework for multi-agent LLM systems reinforced training and inference. In *The Fourteenth International Conference on Learning Representations*, 2026.

[35] Z. Zhang, H. Liu, X. Li, Z. Dai, J. Zeng, F. Wang, M. Lin, R. Chandradevan, L. Wu, Z. Li, C. Luo, Z. Wu, X. Tang, Q. He, and S. Wang. Bradley-terry and multi-objective reward modeling are complementary. In *The Fourteenth International Conference on Learning Representations*, 2026.

[36] Y. Zhao, L. Hu, Y. Wang, M. Hou, H. Zhang, K. Ding, and J. Zhao. Stronger-MAS: Multi-agent reinforcement learning for collaborative LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

[37] L. Zheng, W. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Appendix

In Appendix A, we discuss the limitations of our current framework in detail. Next, in Appendix B, we provide implementation details, including baseline descriptions, dataset specifications, and prompt templates. In Appendix C, we present additional analyses, covering the reliability of the constructed preferences, orchestration diversity across benchmarks, the computational cost of continued orchestrator training, and the sensitivity of Orch-RM to teacher quality and orchestration format. Finally, in Appendix D, we present qualitative examples of the orchestration rollouts of `Orch-RM` and baselines across multiple domains of data.

A Limitations and Broader Impact

Limited scope of data and models. Although `Orch-RM` avoids expensive human annotations by leveraging existing MAS orchestration artifacts, its performance is still constrained by the diversity and quality of available orchestrator models and sampled trajectories. Due to the high computational cost of executing full MAS rollouts, we can only construct reward-model training data from subsets of the available domains and sampled trajectories, particularly for BrowseComp+, where trajectory diversity remains limited. Nevertheless, `Orch-RM` still consistently outperforms competing selection and training methods under this restricted setting (Table 1 and Table 5). Developing larger-scale and more diverse orchestration datasets and benchmarks remains an important direction for future work.

Separate reward models across domains. In this work, we train domain-specific reward models instead of a single unified reward model across all domains. While a unified reward model would improve scalability, different domains involve substantially different MAS orchestration patterns [9]. For instance, DeepScaleR and AIME 24&25 primarily require minimal orchestration, whereas BrowseComp+ and HotpotQA involve medium-level orchestration. As shown in Table 2, directly combining these heterogeneous planning trajectories can introduce noise into the reward-model training data, making reward learning less stable and potentially reducing domain-specific accuracy. Developing a unified reward model that generalizes across diverse domains remains an important direction for future work.

Dependence on teacher quality. Our preference-construction pipeline relies on artifacts from existing orchestrators. As shown in Appendix C.4, reward models trained using artifacts from earlier MAS-Orchestra training steps (25–40) achieve lower Best@N accuracy than those trained using the default teacher artifacts across all three evaluated benchmarks. These results indicate sensitivity to the quality of the available teacher artifacts and suggest that effectiveness may be limited when only weaker orchestrators are available. Reducing this dependence and extending the approach to settings without suitable pretrained orchestrators or execution logs remain important directions for future work.

Broader Impact. This research advances methods for making multi-agent LLMs more capable and efficient through orchestration-level reward modeling, enabling earlier and more efficient intervention during MAS execution. Beyond improving performance, such orchestration-level supervision may also contribute to making MAS more controllable, trustworthy, and safe, as it allows system behaviors to be evaluated and guided before costly or potentially harmful sub-agent actions are fully executed. We hope this direction encourages future research on reliable oversight and governance mechanisms for increasingly autonomous multi-agent systems.

B Implementation Details

We first present the implementation details of the post-training baselines in Appendix B.1. Next, we describe the data specifications used for training `Orch-RM` in Appendix B.2. Finally, we include all prompt templates used throughout this paper in Appendix B.3.

B.1 Baselines

We compare `Orch-RM` against state-of-the-art post-training algorithms for MAS orchestrator training.

- **DPO** [20]: We use the same paired dataset constructed for `Orch-RM` and fine-tune the orchestration policy π_θ with the DPO objective for one epoch.

Table 6: Summary of training configuration.

Setting	Parameters
GPU	2 × RTX PRO 6000 96GB
Batch size	16
Actor rollout	8
Clip ratio	0.2
KL loss	0.001
Learning rate	$1e{-}5$
Actor temperature	0.7
Actor Top p	0.9
LoRA Setting	
rank r	32
alpha α	32
target modules	All linear

- **RFT** [30]: Using the same *"correct-over-incorrect"* data collected from MAS-Orchestra training logs, we perform rejection sampling by retaining only orchestrations that lead to correct final answers, followed by supervised fine-tuning of $\pi_{\boldsymbol{\theta}}$.
- **GRPO** [23]: This baseline corresponds to continued training of MAS-Orchestra [9], strictly following its original training configuration.
- **LLM-as-a-Judge** [37]: We replace the original trajectory-level reward with scores generated by an LLM judge (`GPT-5-mini`), while keeping the remaining GRPO training setup unchanged.

Shared Training Configuration. Please refer to Table 6 for the shared training configuration.

B.2 Datasets

Table 7: Summary of dataset splits used for training, full inference, and testing. GPQA is evaluated strictly zero-shot as an out-of-distribution benchmark.

Dataset	Domain	Training	Full Inference	Testing
DeepScaleR / AIME	Mathematical Reasoning	5,000	500	60
BrowseComp+	Web-based QA	1,066	500	200
HotpotQA	Multi-hop Reasoning	5,000	2,000	200
GPQA	Scientific Reasoning (OOD)	–	–	199

In this section, we provide a detailed breakdown of the datasets used across the different phases of our methodology: policy training, full inference (e.g., generating trajectories/rollouts for reward modeling), and final evaluation. Table 7 summarizes the sample counts for each benchmark.

For mathematical reasoning, we utilize a subset of the DeepScaleR dataset for training and inference, while reserving the AIME 2024 and AIME 2025 problems strictly for testing. Similarly, GPQA is completely held out during the training and inference phases, serving exclusively as an out-of-distribution (OOD) evaluation to assess the cross-domain robustness of our trained orchestrator.

B.3 Prompt Templates

Prompt Template for DeepScaleR

```
<|start_header_id|>system<|end_header_id|>
Cutting Knowledge Date: December 2023
Today Date: 26 Jul 2024
<|eot_id|><|start_header_id|>user<|end_header_id|>
```

<|im_start|>system
You are a helpful assistant.
MASness (How much Multi-Agent System-ness): Minimal
Valid Channels: thinking, agent, answer
Model: gpt-oss-120b
An agent is a pre-configured AI personality that you can delegate tasks to. Each subagent:

1. Has a specific purpose and expertise area
2. Uses its own context window separate from the main conversation
3. (Optional) Can be configured with specific tools it's allowed to use
4. Includes a custom system prompt that guides its behavior

An agent should be defined in channel <agent>. Each agent must contain <agent_name>, <agent_description>, <required_arguments>, and <agent_output_id>.
DO NOT MISS ANY REQUEST FIELDS and ensure that your response is a well-formed XML object! <|im_end|>

<|im_start|>user
Please solve the question step by step and create agents to delegate the task when necessary. First decide whether to solve directly or to delegate to exactly one agent. If you delegate, you must define that agent by outputting <agent> with agent_name (select one of the agents: CoTAgent, SCAgent, DebateAgent, ReflexionAgent), agent_description, required_arguments, and agent_output_id.
The final output of the delegated agent (identified by its agent_output_id) must represent the complete and final answer to the original question, not an intermediate result.
Always put either the final value or the agent_output_id in the <answer> tag, and use EXACTLY the same field names defined for the agent. If the selected agent uses roles (e.g., DebateAgent), also output debate_roles.

For example:
If you decide to solve the entire task yourself, you will output the following:

> **Question:** What is (20+9)*(30+7)?
> <thinking>
> The problem only requires basic arithmetic.
> No specialized reasoning agent or multi-agent discussion is needed.
> (20+9)*(30+7) = 600 + 140 + 270 + 63 = 1073.
> </thinking>
> <answer>1073</answer>

If you decide to solve the entire task via delegation, you will output the following. In this case, since the agent_input is the same as the original task, you must set the agent_input as empty (""), and the parser will replace it with the original question.

> **Question:** Compute the definite integral of (2x + 5) dx from 0 to 3.
> <thinking>
> This problem involves symbolic integration and applying the Fundamental Theorem of Calculus.
> It requires structured reasoning rather than simple numeric computation.
> I will use a calculus agent that can perform step-by-step Chain-of-Thought reasoning.
> The final answer to the original question will be the output of the CoTA-gent.
> </thinking>
> <agent>
> <agent_name>CoTAgent</agent_name>
> <agent_description>Definite integrals with one Chain-of-Thought call.</agent_description>

```
<required_arguments>
<agent_input></agent_input>
</required_arguments>
<agent_output_id>calc_agent_output</agent_output_id>
</agent>
<answer>calc_agent_output</answer>
```

[Additional examples formatted identically...]

Below is the question to solve:
N students are seated at desks in an $m \times n$ array, where $m, n \geq 3$. Each student shakes hands with the students who are adjacent horizontally, vertically or diagonally. If there are 1020 handshakes, what is N?
<|im_end|>

<|im_start|>assistant
Channels:

- <thinking>: internal reasoning and planning
- <agent>: definition of agents
- <answer>: final user-facing answer

Models (the Large Language Model used in agent):

- **gpt-4.1-nano:** Fastest, most cost-efficient version. Excels at instruction following and tool calling. Features a 1M token context window, low latency without a reasoning step. Jun 01, 2024 knowledge cutoff.
- **gpt-oss-120b:** State-of-the-art language model delivering strong real-world performance at low cost. Outperforms similarly sized open models on reasoning tasks, demonstrates strong tool use, and is optimized for efficient deployment. 131,072 context window, Jun 01, 2024 knowledge cutoff, Reasoning token support.

MASness Levels:

- **Minimal:** direct solve or at most one agent
- **Medium:** one or more agents delegation
- **High:** complex multi-agent delegation

Sub-agent Schema (all fields required):

```
<agent>
    <agent_name>...</agent_name> (CoTAgent, SCAgent,

    DebateAgent, ReflexionAgent)
    <agent_description>...</agent_description>
    <required_arguments>
        <...>...</...>
    </required_arguments>
    <agent_output_id>...</agent_output_id>
</agent>
```

Available Agents & Implementations:

1. Chain-of-Thought Agent (CoTAgent)

Description: Encourages the LLM to think step by step rather than directly outputting an answer, enabling complex problem-solving.

Required Arguments: `agent_input` (If left empty, defaults to original question).

```
async def CoTAgent(self, agent_input, model:  str):
   from mas_r1_reasoner.agents.agent_system import LLMAgentBase, Info
   assert isinstance(agent_input, Info), f"agent_input must be an Info
object"
```

```
    temperature = 0.5
    cot_instruction = "Please think step by step and then solve the task."
    cot_agent = LLMAgentBase(['thinking', 'answer'], 'Chain-of-Thought LLM',
model, temperature)
    thinking, answer = await cot_agent([agent_input], cot_instruction)
    return self.make_final_answer(thinking, answer)
```

2. Self-Consistency with Chain-of-Thought (SCAgent)

Description: Repeatedly asks the same question with high temperature to generate different reasoning paths, combining them through majority voting ensembling.

Required Arguments: `agent_input`.

```
async def SCAgent(self, agent_input, model:  str):
    from mas_r1_reasoner.agents.agent_system import LLMAgentBase, Info
    temperature = 0.5; num_repeated_samples = 5
    cot_instruction = "Please think step by step and then solve the task."
    cot_agents = [LLMAgentBase(['thinking', 'answer'], 'Chain-of-Thought LLM',
model, temperature) for _ in range(num_repeated_samples)]
    possible_answers = []
    for i in range(num_repeated_samples):
        thinking, answer = await cot_agents[i]([agent_input], cot_instruction)
        possible_answers.append(answer.content)
    answer = self.majority_voting(possible_answers)
    return self.make_final_answer(thinking, answer)
```

3. LLM Debate (DebateAgent)

Description: Lets different LLMs debate with each other, leveraging diverse perspectives to find better solutions.

Required Arguments: `agent_input`, `debate_roles` (List of roles, e.g., ["Math Professor", "Statistician"]).

```
async def DebateAgent(self, agent_input, model:  str, debate_roles:
List[str]):
    temperature = 0.5; max_debate_round = 5
    debate_agents = [LLMAgentBase(..., role=role) for role in debate_roles]
    for r in range(max_debate_round):
        for i in range(len(debate_agents)):
            # Agents review each other's previous thoughts and update
            thinking, answer = await debate_agents[i](input_infos,
debate_instruction)
    thinking, answer = await final_decision_agent(...,
final_decision_instruction)
    return self.make_final_answer(thinking, answer)
```

4. Self-Refine (ReflexionAgent)

Description: Iteratively improves answers based on feedback from a Critic agent.

Required Arguments: `agent_input`.

```
async def ReflexionAgent(self, agent_input, model:  str):
    temperature = 0.5; max_reflection_round = 5
    cot_agent = LLMAgentBase(['thinking', 'answer'], 'Chain-of-Thought LLM',
...)
    critic_agent = LLMAgentBase(['feedback', 'correct'], 'Critic LLM', ...)
    thinking, answer = await cot_agent(cot_inputs, initial_instruction, 0)
    for i in range(max_reflection_round):
        feedback, correct = await critic_agent([agent_input, thinking, answer],
...)
        if correct.content == 'True':  break
        thinking, answer = await cot_agent(..., reflect_instruction, i + 1)
    return self.make_final_answer(thinking, answer)
```

<|im_end|><|eot_id|><|start_header_id|>assistant<|end_header_id|>
<thinking>
To solve this problem, we need to determine the number of handshakes that occur in an $m \times n$ grid where each student shakes hands with their adjacent neighbors (horizontally, vertically, and diagonally). The total number of handshakes is given as 1020. We need to find the number of students N in the grid.
</thinking>

<agent>
<agent_name>DebateAgent</agent_name>
<agent_description>Near-tie numeric choice using one Debate call.</agent_description>
<required_arguments>
<agent_input></agent_input>
<debate_roles>["Graph Theory Expert", "Combinatorics Specialist"]</debate_roles>
</required_arguments>
<agent_output_id>debate_agent_output</agent_output_id>
</agent>
<answer>debate_agent_output</answer>
In this problem, the number of handshakes can be complex to calculate directly without a structured approach. The DebateAgent will help us consider different combinatorial and graph theory perspectives to ensure we reach the correct solution. <|eot_id|>

C Additional Analyses

C.1 Reliability of Constructed Preferences

We construct SoB preferences by treating orchestration plans generated by the domain-specialized orchestrator as preferred over those generated by the base model for the same input. For CoI preferences, we compare plans for the same input and treat those whose executions produce correct final answers as preferred over those whose executions produce incorrect answers. Neither guarantees that the preferred plan will outperform the other plan in every execution, since sub-agent execution is stochastic. We therefore evaluate the reliability of both preference sources through repeated executions of fixed orchestration plans.

Experimental setup. We sample SoB and CoI preference pairs from the DeepScaleR, BrowseComp+, and HotpotQA training datasets. For each pair, we fix both orchestration plans and execute each plan five times using different sampling seeds. We estimate each plan's success probability by the proportion of executions producing a correct final answer, and compare these estimates to assess agreement with the original preference label.

Table 8: Pairwise win rates of the originally preferred orchestration plans under repeated execution. Each fixed plan is executed five times. Values are percentages.

Dataset	CoI	SoB
DeepScaleR	68.00	94.00
BrowseComp+	67.50	96.50
HotpotQA	72.50	96.00

Results. As shown in Table 8, SoB preferences achieve pairwise win rates of 94.0–96.5%, whereas CoI preferences achieve 67.5–72.5%. Both sources therefore retain useful preference information under repeated execution, although agreement is lower for CoI. These results support their use as supervision while highlighting the potential benefit of constructing preferences from aggregated execution outcomes.

C.2 Orchestration Diversity Across Benchmarks

To investigate the uneven benefits of orchestration-level reward modeling across benchmarks, we analyze generated orchestration plans along two dimensions: (i) semantic diversity, based on cosine similarity between plan embeddings, and (ii) structural diversity, characterized by the number and types of selected sub-agents. Table 9 summarizes the results.

Table 9: Orchestration diversity across datasets. Sub-agent-count distributions and agent-type shares are separately normalized to 100% before rounding. Values are reported to two decimal places; displayed totals may differ slightly from 100%.

Semantic diversity and sub-agent counts

Dataset	Semantic diversity	1	2	3	4	5	6	> 6
DeepScaleR	0.12	86.65	9.04	1.63	1.20	0.47	0.33	0.67
BrowseComp+	0.15	0.02	0.04	1.71	37.48	45.21	9.87	5.67
HotpotQA	0.04	0.00	99.89	0.11	0.00	0.00	0.00	0.00
GPQA	0.19	99.75	0.25	0.00	0.00	0.00	0.00	0.00

Agent-type composition

Dataset	CoT	Debate	Reflexion	Web Search	SC
DeepScaleR	0.00	55.45	44.52	0.00	0.02
BrowseComp+	15.48	0.00	17.69	66.81	0.02
HotpotQA	50.00	0.00	0.00	50.00	0.00
GPQA	0.00	83.17	16.83	0.00	0.00

Results. HotpotQA has the lowest reported semantic-diversity score, and 99.89% of its plans use exactly two sub-agents. Its agent-type composition is evenly divided between CoT and Web Search. GPQA is also structurally homogeneous: 99.75% of its plans use a single sub-agent, with Debate and Reflexion accounting for 83.17% and 16.83% of the agent-type composition, respectively. Importantly, GPQA does not exhibit low semantic diversity under the reported metric and its homogeneity is primarily structural. These patterns suggest that limited structural variation may leave fewer opportunities for reward-guided plan selection, potentially contributing to the smaller gains on these benchmarks.

C.3 Computational Cost of Continued Orchestrator Training

We compare the estimated computational costs of GRPO and `Orch-RM`-guided continued orchestrator training. The comparison includes upfront reward-model training, orchestrator sampling and optimization, and reward computation.

Cost estimation. Let C denote computational cost in FLOPs, and let P_o, P_r, and P_s denote the parameter counts of the orchestrator, reward model, and sub-agent model, respectively. In our experiments, we use a 7B orchestrator ($P_o = 7 \times 10^9$), an 8B reward model ($P_r = 8 \times 10^9$), and a 120B sub-agent model ($P_s = 120 \times 10^9$). We approximate training and inference costs by $6PT$ and $2PT$, respectively, where T denotes the number of processed tokens.

Our continued-training experiments use $N = 8$ parallel candidate plans per query for both GRPO and `Orch-RM`. The matched computational-cost comparison uses this same configuration. We additionally report estimated GRPO costs at $N = 32$ as a separate reference corresponding to its default configuration.

The cost components are defined as follows:

1. **Reward-model training.** Let C_r^{train} denote the total cost of `Orch-RM` training, and T_r^{train} denote the total training tokens processed by `Orch-RM`. The upfront training cost is $C_r^{\text{train}} \approx 6P_rT_r^{\text{train}}$. This component applies only to `Orch-RM`.

2. **Orchestrator inference.** Let C_o^{infer} denote the total cost of orchestrator inference, and T_o^{infer} denote the total inference tokens processed when generating candidate orchestration plans. The inference cost is $C_o^{\text{infer}} \approx 2P_oT_o^{\text{infer}}$. This component applies to both GRPO and `Orch-RM`.

3. **Orchestrator update.** Let C_o^{train} denote the total cost of the orchestrator update, and T_o^{train} denote the total training tokens processed during orchestrator optimization. The update cost is $C_o^{\text{train}} \approx 6P_oT_o^{\text{train}}$. This component applies to both GRPO and `Orch-RM`.

4. **Sub-agent rollouts.** Let C_s^{infer} denote the total cost of sub-agent execution, and T_s^{infer} denote the total inference tokens processed by sub-agents when executing candidate plans. For GRPO, the reward-computation cost is $C_s^{\text{infer}} \approx 2P_sT_s^{\text{infer}}$. This component applies only to GRPO.

5. **Reward-model scoring.** Let C_r^{infer} denote the total cost of `Orch-RM` inference on the orchestration plans, and T_r^{infer} denote the total inference tokens processed by the reward model when scoring candidate plans. For `Orch-RM`, the reward-computation cost is $C_r^{\text{infer}} \approx 2P_rT_r^{\text{infer}}$. This component applies only to `Orch-RM`.

Token counts are accumulated separately for each method and component over the full sampling and training process. The total GRPO cost is $C_o^{\text{infer}} + C_o^{\text{train}} + C_s^{\text{infer}}$, whereas the total `Orch-RM` cost is $C_r^{\text{train}} + C_o^{\text{infer}} + C_o^{\text{train}} + C_r^{\text{infer}}$.

All costs are reported in exaFLOPs (EF), where $1\,\text{EF} = 10^{18}$ FLOPs. These are analytical estimates of computational cost, rather than measured wall-clock runtimes.

Table 10: Estimated computational costs of continued orchestrator training, in EF. N denotes the number of parallel candidate samples. Total costs include upfront reward-model training for `Orch-RM`.

Method	N	`Orch-RM` training	Orch. inference	Orch. update	Sub-agent rollouts	`Orch-RM` scoring	Total
DeepScaleR							
GRPO (default)	32	0.00	18.84	2.68	321.44	0.00	342.96
GRPO	8	0.00	4.71	0.67	80.36	0.00	85.74
`Orch-RM`	8	1.06	4.71	0.67	0.00	2.82	**9.26**
BrowseComp+							
GRPO (default)	32	0.00	7.24	1.32	1028.44	0.00	1037.00
GRPO	8	0.00	1.81	0.33	257.11	0.00	259.25
`Orch-RM`	8	1.09	1.81	0.33	0.00	0.78	**4.01**
HotpotQA							
GRPO (default)	32	0.00	28.80	5.80	130.88	0.00	165.48
GRPO	8	0.00	7.20	1.45	32.72	0.00	41.37
`Orch-RM`	8	1.34	7.20	1.45	0.00	3.11	**13.10**

Results. Under the matched $N = 8$ setting, Table 10 shows that GRPO requires approximately 9.3, 64.7, and 3.2 times the estimated FLOPs of `Orch-RM` on DeepScaleR, BrowseComp+, and HotpotQA, respectively. These reductions persist after including the upfront cost of reward-model training. The largest reduction occurs on BrowseComp+, where sub-agent rollouts dominate the GRPO cost. These results highlight the computational benefit of replacing full sub-agent execution with orchestration-level reward scoring during continued training.

C.4 Sensitivity to Teacher Quality

We investigate how the quality of the source orchestrator affects the resulting orchestration reward model. We train a weaker-teacher variant, denoted as `Orch-RM` (WT), using artifacts from early MAS-Orchestra training steps (25–40), following the same preference-construction procedure as in our main setting. The default model instead uses artifacts from the domain-specialized teachers. Table 11 summarizes the teacher checkpoints and their average accuracy.

Table 11: Comparison of weaker and default teachers. WT denotes the weaker teacher. Steps refer to MAS-Orchestra training steps. Accuracy values are percentages.

Benchmark	WT steps	WT avg. acc.	Default step	Default avg. acc.
AIME	25–40	48.75	180	58.96
BrowseComp+	25–40	6.25	140	9.44
HotpotQA	25–40	40.25	250	40.63

We then compare the Best-of-N selection performance of `Orch-RM` (WT) and the default `Orch-RM`. Oracle selection is included as a reference.

Table 12: Best-of-N accuracy of reward models trained using weaker and default teacher artifacts. Values are percentages.

Benchmark	Oracle	`Orch-RM` (WT)	`Orch-RM`
AIME	76.67	40.00	**68.33**
BrowseComp+	24.50	9.00	**14.00**
HotpotQA	57.00	36.50	**42.50**

Results. As shown in Table 12, the default `Orch-RM` outperforms the weaker-teacher variant across all three evaluated benchmarks. These results indicate that the choice of teacher artifacts affects reward-guided orchestration selection, with artifacts from the default teachers yielding stronger performance than those from earlier training stages.

C.5 Sensitivity to Orchestration Format

We investigate whether `Orch-RM` can score orchestration plans expressed in a format different from its training representation.

Experiment setup. We use GPT-5-mini to generate natural-language (NL) orchestration plans and convert the same plans into XML. We then score both representations using `Orch-RM` and an LLM-as-a-Judge baseline. Sub-agent execution uses the XML representation in both conditions, allowing us to compare selection performance when the scoring input is expressed in NL or XML.

Table 13: Orchestration selection performance with natural-language (NL) and XML scoring inputs. Both representations correspond to the same generated plans, and sub-agent execution uses XML. Accuracy values are percentages.

Benchmark	Average acc.	Oracle	`Orch-RM` Best@N		LLM Judge	
			NL	XML	NL	XML
AIME	49.33	76.67	50.00	56.66	50.00	50.00
BrowseComp+	3.50	13.50	3.50	7.00	3.50	5.50
HotpotQA	35.44	59.50	34.00	38.50	35.00	38.50

Results. As shown in Table 13, `Orch-RM` achieves higher selection accuracy with XML inputs than with NL inputs across all three benchmarks. Direct NL scoring provides little or no improvement over average candidate accuracy, whereas XML scoring improves selection performance. Even when the plans are generated by GPT-5-mini rather than MAS-Orchestra, `Orch-RM` matches or exceeds the LLM-as-a-Judge baseline across all three benchmarks when scoring their XML representations.

These results suggest that the transferability of `Orch-RM` is currently format-sensitive, but its learned preference signal remains effective for plan selection when new orchestration plans are mapped into the XML representation used during training.

D Orchestration Rollout Examples: Base vs. Trained Model

To empirically demonstrate the impact of our Orchestration Reward Modeling, we provide raw rollout samples from the AIME, BrowseComp+ and HotpotQA benchmarks. In the following sections, we define our two comparative stages as follows:

- **Base Model:** The baseline checkpoint trained entirely through the original MAS-Orchestra pipeline.
- **Trained Model:** The resulting policy after continual training from the baseline checkpoint using our proposed reward modeling framework.

These samples highlight a massive distributional shift in policy behavior, demonstrating how our continual training resolves the structural and planning bottlenecks inherent in the baseline framework. Poor orchestration plans directly result in subagent failure, while strict, mathematically sound plans yield correct ground truth answers.

D.1 Base Model Rollouts

Task: AIME 2025 (ID: aime25_11)

The set of points in 3-dimensional coordinate space that lie in the plane $x + y + z = 75$ whose coordinates satisfy the inequalities $x - yz < y - zx < z - xy$ forms three disjoint convex regions. Exactly one of those regions has finite area. The area of this finite region can be expressed in the form $a\sqrt{b}$, where a and b are positive integers and b is not divisible by the square of any prime. Find $a + b$.

Base Model: Sample 1

```
]
<thinking>
    This problem involves geometric and algebraic concepts. The set of points in 3-dimensional
     space that lie in the plane \(x + y + z = 75\) and satisfy the inequalities \(x - yz
     \textless y - zx \textless z - xy\) can be analyzed by breaking it down into smaller,
     manageable parts. The key is to understand the geometric regions defined by these
     inequalities and their intersection with the plane.
</thinking>
<agent>
    <agent_name>DebateAgent</agent_name>
    <agent_description>Near-tie numeric choice using one Debate call.</agent_description>
    <required_arguments>
        <agent_input></agent_input>
        <debate_roles>["Geometric Analyst", "Algebraic Solver"]</debate_roles>
    </required_arguments>
    <agent_output_id>
    debate_agent_output
    </agent_output_id>
</agent>
<answer>debate_agent_output</answer>

Here, we have delegated the problem to the DebateAgent, which will handle the detailed geometric
     and algebraic analysis. The two roles, "Geometric Analyst" and "Algebraic Solver," will
     debate and to find the final the final the inequalities and the intersection with the plane,
     ultimately providing the correct region.

--------------------------------------------------------------------------
> Executing DebateAgent [Geometric Analyst vs. Algebraic Solver]...
> Analyst: Projection onto xy-plane yields a triangle of area 507/4.
> Solver: Derived side lengths based on plane x+y+z=75, area is 150*sqrt(3).
> Resulting Rationale: Conflict in coordinate derivation led to summation error.
> Final Answer Produced: 1017
--------------------------------------------------------------------------
Result: [Incorrect] (Ground Truth: 510)
```

Base Model: Sample 4

```
<thinking>
    This problem involves geometric and algebraic reasoning, specifically dealing with the
     intersection of a plane and inequalities. The key steps involve understanding the plane
     equation and the inequalities, which can be complex. I will delegate this to the
     ReflexionAgent to ensure accuracy and thoroughness.
</thinking>
<agent>
    <agent_name>ReflexionAgent</agent_name>
    <agent_description>Solves reasoning tasks with iterative self-refine loop (single agent
     call).</agent_description>
    <required_arguments>
        <agent_input></agent_input>
    </required_arguments>
    <agent_output_id>
    reflexion_agent_output
    </agent_output_id>
</agent>
<answer>reflexion_agent_output</answer>

The ReflexionAgent will handle the detailed reasoning and ensure that the solution is accurate
     and complete. The ReflexionAgent will provide the final answer, which will be in the form
     \(a\sqrt{b}\).

--------------------------------------------------------------------------
> Executing ReflexionAgent...
> Subagent thinking: Triangle vertices at (0,0,75), (0,75,0), (75,0,0)...
> Side length is 75, area is 2812.5*sqrt(3).
> Final Answer Produced: 903
--------------------------------------------------------------------------
Result: [Incorrect] (Ground Truth: 510)
```

Base Model: Sample 7

```
<thinking>
    To solve this problem, we need to identify the regions in the plane \(x + y + z = 75\) that
     satisfy the inequalities \(x - yz\textless y - zx\textless z - xy\). Lets analyze the
     inequalities step by step.
</thinking>
<agent>
    <agent_name>DebateAgent</agent_name>
    <agent_description>Near-tie numeric choice using one Debate call.</agent_description>
    <required_arguments>
        <agent_input></agent_input>
        <debate_roles>["Mathematics Professor", "Geometry Specialist"]</debate_roles>
    </required_arguments>
    <agent_output_id>
    debate_agent_output
    </agent_output_id>
</agent>
<answer>debate_agent_output</answer>

### Detailed Explanation of the Agent Delegation

1. Initial Analysis:
   - The plane equation \(x + y + z = 75\) represents a plane in 3D space.
   - The inequalities \(x - yz\textless y - zx\textless z - xy\) need to be analyzed to determine
     the regions where they hold true.

2. Agent into the De De of the inequalities:
   - \(x - yz\textless y - zx\)
   - \(y - zx\textless z - xy\)

3. Delegation to DebateAgent:
   - We need to debate the regions where these inequalities hold true.
   - The DebateAgent will involve perspectives from a "Mathematics Professor" and a "Geometry
     Specialist" to ensure a comprehensive and accurate solution.

### Agent Delegation to DebateAgent

The DebateAgent will handle the detailed analysis and provide a final answer based on the
     perspectives of the two roles.
```

```
<thinking>
    To solve this problem, we need to identify the regions in the plane \(x + y + z = 75\) that
     satisfy the inequalities \(x - yz\textless y - zx\textless z - xy\). Lets analyze the
     inequalities step by step.
</thinking>
<agent>
    <agent_name>DebateAgent</agent_name>
    <agent_description>Near-tie numeric choice using one Debate call.</agent_description>
    <required_arguments>
        <agent_input></agent_input>
        <debate_roles>["Mathematics Professor", "Geometry Specialist"]</debate_roles>
    </required_arguments>
    <agent_output_id>
    debate_agent_output
    </agent_output_id>
</agent>
<answer>debate_agent_output</answer>

--------------------------------------------------------------------------
> Executing DebateAgent...
> System Error: Redundant XML blocks detected.
> Subagent thinking: Intersection yields region bounded by x=-1, y=-1, z=-1.
> Area calculation resulted in triangle with area 300*sqrt(3).
> Final Answer Produced: 303
--------------------------------------------------------------------------
Result: [Incorrect] (Ground Truth: 510)
```

D.2 Our Trained Orchestrator Rollouts

Task: AIME 2025 (ID: aime25_11)

The set of points in 3-dimensional coordinate space that lie in the plane $x + y + z = 75$ whose coordinates satisfy the inequalities $x - yz < y - zx < z - xy$ forms three disjoint convex regions. Exactly one of those regions has finite area. The area of this finite region can be expressed in the form $a\sqrt{b}$, where a and b are positive integers and b is not divisible by the square of any prime. Find $a + b$.

Trained Model: Planning Response (Sample 1)

```
<thinking>
    Compute the area of the finite convex region defined by $x+y+z=75$ and $x-yz\textless
     y-zx\textless z-xy$.
</thinking>
<agent>
    <agent_name>DebateAgent</agent_name>
    <agent_description>Near-tie numeric choice and geometric/inequality constraints resolution
     using one Debate call.</agent_description>
    <required_arguments>
        <agent_input></agent_input>
        <debate_roles>["Geometry Specialist", "Inequality Expert"]</debate_roles>
    </required_arguments>
    <agent_output_id>
    debate_agent_output
    </agent_output_id>
</agent>
<answer>debate_agent_output</answer>

Explanation:

1. Thinking Phase:
   - We comprehend the given problem: Finding the area of a finite convex region in 3D space
     defined by the plane equation \(x + y + z = 75\) and the inequalities \(x - yz\textless y -
     zx\textless z - xy\).

2. Delegation Decision:
```

- This problem combines geometry (plane and convex regions), algebra (inequalities), and area computation. It benefits from diverse perspectives, making it ideal for the DebateAgent. The DebateAgent allows different specialized LLMs (e.g., a "Geometry Specialist" and an "Inequality Expert") to debate and converge on the correct solution.
- DebateAgent is chosen over others (like CoTAgent, SCAgent, or ReflexionAgent) because:
  - CoTAgent focuses on step-by-step reasoning but doesn't leverage multiple viewpoints.
  - SCAgent aims for consensus through repeated sampling, which is overkill here.
  - ReflexionAgent is better suited for self-correction after initial attempts, not initial complex problem-solving.

3. Agent Definition:
   - Agent Name: DebateAgent
   - Agent Description: Fits the problem's need for diverse and justified reasoning.
   - Required Arguments:
     - agent_input: Set to "" (empty) as instructed, ensuring the original question is inserted by the parser.
     - debate_roles: A list with two distinct expert roles ("Geometry Specialist" and "Inequality Expert"), promoting a rich debate.
   - Agent Output ID: debate_agent_output to reference the final answer from the delegated agent.

4. Final Answer Tag:
   - The <answer> tag will contain the output identifier debate_agent_output, representing the complete and verified solution generated by the DebateAgent.

This approach ensures both correctness and confidence through collaborative reasoning.

> Executing DebateAgent [Geometry Specialist vs. Inequality Expert]...
> Specialist: Identified boundary lines x=y, y=z, and z=-1 on the plane.
> Expert: Verified bounded region occurs when x > -1, z > -1, forcing x < y < z.
> Consensus: Triangle area calculated via cross product yields 507*sqrt(3).
> Final Answer Produced: 510

Result: [Correct] (Ground Truth: 510)

Trained Model: Planning Response (Sample 4)

<thinking>
Compute the area of the finite region defined by $x+y+z=75$ and $x-yz \textless y-zx \textless z-xy$.
</thinking>
<agent>
<agent_name>DebateAgent</agent_name>
<agent_description>Near-tie numeric choice and geometric/inequality constraint satisfaction using one Debate call.</agent_description>
<required_arguments>
<agent_input></agent_input>
<debate_roles>["Geometry Specialist", "Inequality Expert"]</debate_roles>
</required_arguments>
<agent_output_id>
debate_agent_output
</agent_output_id>
</agent>
<answer>debate_agent_output</answer>

Explanation:

1. Thinking Phase:
   - We understand the given constraints: the plane equation \(x + y + z = 75\) and the inequalities \(x - yz\textless y - zx\textless z - xy\). The goal is to find the finite-area convex region formed by these conditions and express its area in the form \(a\sqrt{b}\), then compute \(a + b\).

2. Delegation Decision:
   - This problem combines geometry (plane equations and region definitions), algebra (inequalities), and area computation. It benefits from diverse perspectives, making it ideal for the DebateAgent. The DebateAgent allows different specialized LLMs (e.g., a "Geometry Specialist" and an "Inequality Expert") to debate and converge on the correct solution. This agent is chosen over others (like CoTAgent, SCAgent, or ReflexionAgent) because:
   - CoTAgent focuses on step-by-step reasoning but doesn't leverage multiple viewpoints.
   - SCAgent aims for consensus through repeated sampling, which is overkill here.
   - ReflexionAgent is better suited for self-correction after initial attempts, not initial complex problem-solving.

3. Agent Definition:
   - Agent Name: DebateAgent
   - Agent Description: Fits the problem's need for diverse and justified reasoning.
   - Required Arguments:
     - agent_input: Set to "" (empty) as instructed, ensuring the original question is inserted by the parser.
     - debate_roles: A list containing two distinct expert roles ("Geometry Specialist" and "Inequality Expert"), promoting a rich debate.
   - Agent Output ID: debate_agent_output to reference the final answer from the delegated agent.

4. Final Answer Tag:
   - The <answer> tag will contain the output identifier debate_agent_output, representing the complete and justified solution generated by the DebateAgent.

This approach ensures both correctness and confidence through collaborative reasoning.

> Executing DebateAgent [Geometry Specialist vs. Inequality Expert]...
> Specialist: Cross product of vectors AB and AC is (507, 507, 507).
> Expert: Magnitude is 507*sqrt(3), area is half that. Wait, full parallelogram area matches the problem constraints better.
> Consensus: a=507, b=3. Therefore, a+b=510.
> Final Answer Produced: 510

Result: [Correct] (Ground Truth: 510)

Trained Model: Planning Response (Sample 8)

<thinking>
Compute the area of the finite region defined by the plane \(x + y + z = 75\) and the strict inequalities \(x - yz\textless y - zx\textless z - xy\).
</thinking>
<agent>
<agent_name>DebateAgent</agent_name>
<agent_description>Near-tie numeric choice and geometric constraint satisfaction using one Debate call.</agent_description>
<required_arguments>
<agent_input></agent_input>
<debate_roles>["Geometry Specialist", "Inequality Expert"]</debate_roles>
</required_arguments>
<agent_output_id>
debate_agent_output
</agent_output_id>
</agent>
<answer>debate_agent_output</answer>

Explanation:

1. Thinking Phase:
   - We comprehend the given problem: Find the area of the finite region in the plane \(x + y + z = 75\) constrained by \(x - yz\textless y - zx\textless z - xy\).

2. Delegation Decision:
   - This problem combines geometric constraints (plane equation and region bounds) and algebraic inequalities. It benefits from diverse perspectives, making it ideal for the DebateAgent. The DebateAgent allows different specialized LLMs (e.g., a "Geometry Specialist" and an "Inequality Expert") to debate and converge on the correct solution.
   - DebateAgent is chosen over others (like CoTAgent, SCAgent, or ReflexionAgent) because:
     - CoTAgent focuses on step-by-step reasoning but doesn't leverage multiple viewpoints.
     - SCAgent aims for consensus through repeated sampling, which is overkill here.
     - ReflexionAgent is better suited for self-correction after initial attempts, not initial complex problem-solving.

3. Agent Definition:
   - Agent Name: DebateAgent
   - Agent Description: Fits the problem's need for diverse and justified reasoning.
   - Required Arguments:
     - agent_input: Set to "" (empty) as instructed, ensuring the original question is inserted by the parser.
     - debate_roles: A list with two distinct expert roles ("Geometry Specialist" and "Inequality Expert"), promoting a rich debate.
   - Agent Output ID: debate_agent_output to reference the final answer from the delegated agent.

```
4. Final Answer Tag:
   - The <answer> tag will contain the output identifier debate_agent_output, representing the
     complete and justified solution generated by the DebateAgent.

This approach ensures both correctness and confidence through collaborative reasoning.

--------------------------------------------------------------------
> Executing DebateAgent [Geometry Specialist vs. Inequality Expert]...
> Specialist: Intersecting plane with the "z=-1" equality gives an unbounded wedge.
> Expert: Incorrectly anchored on boundary lines x=-1, y=-1, z=-1.
> Consensus: Triangle area calculated as 3042*sqrt(3).
> Final Answer Produced: 3045
--------------------------------------------------------------------
Result: [Incorrect] (Ground Truth: 510)
```

D.3 Base Model Rollouts: BrowseComp+

Task: BrowseComp+ (qid: 371) – Ground Truth: Reclus

A person born between 1820 and 1830, both years inclusive, had a street in a European capital named after them, at least until 2020. This street was called with their name two years after their death. This individual and their brother left their house to move to Germany when the person was less than 15 but more than 10 years old. In Germany, the brothers met an English writer appointed to the Order of Merit between 1902 and 1907, both years inclusive. The person born between 1820 and 1830, both years inclusive, served as a private tutor in the United States. Also, This person was a friend of someone who photographed celebrities. What is the last name of the person mentioned who was born between 1820 and 1830, both years inclusive?

Sample 1

```
<thinking>
  The task involves multiple historical and biographical details that require precise and
     up-to-date information. A single agent is insufficient due to the complexity and the need
     for accurate, sourced information. We will use three WebSearchAgents to gather specific
     facts, a CoTAgent to synthesize the information, and a ReflexionAgent to ensure the final
     answer is accurate and well-supported.

  We will define three WebSearchAgents:
  - WS1 to find the name and birth year of the person.
  - WS2 to find the details of the street named after the person and when it was named.
  - WS3 to find the details of the persons move to Germany, meeting with the English writer, and
     their role as a private tutor.

  The CoTAgent will then consolidate the findings from the WebSearchAgents and provide a coherent
     answer. Finally, the ReflexionAgent will ensure the answer is accurate and complete.
</thinking>

<agent>
  <agent_id>WS1</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Find the name and birth year of the person.</agent_description>
  <required_arguments>
    <agent_input>Search for the name and birth year of a person born between 1820 and 1830 who
     had a street named after them in a European capital and served as a private tutor in the
     US.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>WS2</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Find the details of the street named after the person and when it was
     named.</agent_description>
```

```
  <required_arguments>
    <agent_input>Search for the details of the street named after the person born between 1820
      and 1830 and when it was named.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>WS3</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Find the details of the persons move to Germany, meeting with the English
     writer, and their role as a private tutor.</agent_description>
  <required_arguments>
    <agent_input>Search for the details of the person born between 1820 and 1830 who moved to
      Germany, met an English writer appointed to the Order of Merit between 1902 and 1907, and
      served as a private tutor in the US.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>CoT</agent_id>
  <agent_name>CoTAgent</agent_name>
  <agent_description>Consolidate the findings from the WebSearchAgents and provide a coherent
     answer.</agent_description>
  <required_arguments>
    <agent_input>Using the results below:
Street details: \${WS2}
Move details: \${WS3}
Name and birth year: \${WS1}

Determine the last name of the person born between 1820 and 1830 who had a street named after
     them and served as a private tutor in the US.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>REFLECT</agent_id>
  <agent_name>ReflexionAgent</agent_name>
  <agent_description>Ensure the final answer is accurate and well-supported.</agent_description>
  <required_arguments>
    <agent_input>Using the findings from the WebSearchAgents and the CoTAgent, verify the last
      name of the person born between 1820 and 1830.

WebSearchAgent results:
\${WS1}
\${WS2}
\${WS3}

CoTAgent result:
\${CoT}</agent_input>
  </required_arguments>
</agent>

<edge>
  <from>WS1</from>
  <to>CoT</to>
  <from>WS2</from>
  <to>CoT</to>
  <from>WS3</from>
  <to>CoT</to>
  <from>CoT</from>
  <to>REFLECT</to>
</edge>

--------------------------------------------------------------------------
> Executing WebSearchAgents...
> WS1: Retrieved candidate Jules Verne (born 1828).
> REFLECT: Synthesized findings and anchored on candidate.
> Final Answer Produced: Verne
--------------------------------------------------------------------------
Result: [Incorrect] (Ground Truth: Reclus)
```

D.4 Our Trained Orchestrator Rollouts: BrowseComp+

Task: BrowseComp+ (qid: 371) – Ground Truth: Reclus

A person born between 1820 and 1830, both years inclusive, had a street in a European capital named after them, at least until 2020. This street was called with their name two years after their death. This individual and their brother left their house to move to Germany when the person was less than 15 but more than 10 years old. In Germany, the brothers met an English writer appointed to the Order of Merit between 1902 and 1907, both years inclusive. The person born between 1820 and 1830, both years inclusive, served as a private tutor in the United States. Also, This person was a friend of someone who photographed celebrities. What is the last name of the person mentioned who was born between 1820 and 1830, both years inclusive?

Trained Model: Sample 2

```
<thinking>
  This problem requires verifying historical facts, dates, and relationships involving multiple
     individuals over decades. A single agent cannot reliably:
  - Confirm precise dates and names across different time periods.
  - Ensure the consistency of historical facts from various sources.
  - Handle the intricate timeline and relationships logically.

  We will use three WebSearchAgents to independently verify key historical facts (dates, places,
     and relationships):
  - One for the birth and early life of the individual.
  - One for their time in Germany and meeting the English writer.
  - One for their activities in the U.S. and friendship with the photographer.

  These agents will provide disjointed but detailed factual evidence. We then use a
     ReflexionAgent to integrate these findings, cross-verify, and refine the final answer. The
     ReflexionAgent ensures the narrative is coherent and factually accurate.

  The ReflexionAgent will produce the final, verifiable answer, making it the unique sink node of
     the graph.
</thinking>

<agent>
  <agent_id>SEARCH1</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Verify birth and early life details.</agent_description>
  <required_arguments>
    <agent_input>Find reliable sources confirming the birth year (1820-1830), early life, and
     moving to Germany before age 15.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>SEARCH2</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Confirm Germany and English writer interaction.</agent_description>
  <required_arguments>
    <agent_input>Search for evidence of the individual meeting the English writer appointed to
     the Order of Merit between 1902-1907 in Germany.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>SEARCH3</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Validate U.S. tutoring and photographer friendship.</agent_description>
  <required_arguments>
    <agent_input>Find credible sources confirming the persons U.S. tutoring role and friendship
     with a photographer who photographed celebrities.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>REFLECT</agent_id>
  <agent_name>ReflexionAgent</agent_name>
```

```
  <agent_description>Integrate findings and provide a coherent, verified
     answer.</agent_description>
  <required_arguments>
    <agent_input>Using the findings below:
    Birth and early life: \${SEARCH1}
    Germany and writer: \${SEARCH2}
    U.S. and photographer: \${SEARCH3}

    Determine the last name of the person born between 1820 and 1830, verifying each claim with
     citations.</agent_input>
  </required_arguments>
</agent>

<edge>
  <from>SEARCH1</from>
  <to>REFLECT</to>
  <from>SEARCH2</from>
  <to>REFLECT</to>
  <from>SEARCH3</from>
  <to>REFLECT</to>
</edge>

--------------------------------------------------------------------------
> Executing WebSearchAgents...
> SEARCH1: Identified Elisee Reclus (born 1830), Avenue Elisee-Reclus in Paris.
> SEARCH2: Confirmed move to Neuwied, Germany (1842) and meeting George Meredith.
> SEARCH3: Verified tutoring in Louisiana, US and friendship with Nadar.
> REFLECT: Synthesized constraints: Reclus matches all disjoint historical markers.
> Final Answer Produced: Reclus
--------------------------------------------------------------------------
Result: [Correct] (Ground Truth: Reclus)
```

Trained Model: Sample 4

```
<thinking>
  This problem requires verifying historical and biographical facts spanning multiple time
     periods and locations. A single agent cannot reliably confirm all details due to the need
     for precise, verifiable sources from different eras and regions.

  We will use three WebSearchAgents to independently verify:
  1. The birth year (1820-1830) and the named street in a European capital.
  2. The move to Germany with the brother and the encounter with the English writer.
  3. The role as a tutor in the United States and the friendship with the photographer.

  Each agent focuses on a distinct aspect, ensuring no detail is overlooked. The findings will be
     cross-verified, and the ReflexionAgent will consolidate the information, cross-checking
     dates, names, and events for accuracy.

  The ReflexionAgent is the final sink, producing a coherent, verified summary that directly
     answers the question about the persons last name.
</thinking>

<agent>
  <agent_id>SEARCH1</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Verify the birth year, street name, and European capital.</agent_description>
  <required_arguments>
    <agent_input>Find facts about the person born between 1820 and 1830 with a street named after
     them in a European capital, at least until 2020. Return a detailed citation.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>SEARCH2</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Verify the move to Germany and encounter with the English
     writer.</agent_description>
  <required_arguments>
    <agent_input>Research the persons move to Germany with their brother, and meeting with an
     English writer appointed to the Order of Merit between 1902 and 1907. Return detailed
     citations.</agent_input>
  </required_arguments>
</agent>
```

```
<agent>
  <agent_id>SEARCH3</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Verify the US tutor role and friendship with the
      photographer.</agent_description>
  <required_arguments>
    <agent_input>Find evidence of the person serving as a private tutor in the United States and
      being a friend of someone who photographed celebrities. Return detailed
      citations.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>REFLECT</agent_id>
  <agent_name>ReflexionAgent</agent_name>
  <agent_description>Consolidate findings and verify the persons identity.</agent_description>
  <required_arguments>
    <agent_input>Using the findings below:
    Birth and Street: \${SEARCH1}
    Germany and Writer: \${SEARCH2}
    Tutor and Photographer: \${SEARCH3}

    Cross-check all details: birth year, street name, move dates, writer details, US role, and
      photographer. Confirm the persons identity and last name.</agent_input>
  </required_arguments>
</agent>

<edge>
  <from>SEARCH1</from>
  <to>REFLECT</to>
  <from>SEARCH2</from>
  <to>REFLECT</to>
  <from>SEARCH3</from>
  <to>REFLECT</to>
</edge>

--------------------------------------------------------------------------
> Executing WebSearchAgents...
> SEARCH1: Retrieved Jules Verne (born 1828), noting Rue Jules Verne in Paris.
> SEARCH2: Failed to link Jules Verne to the Order of Merit writer constraint.
> SEARCH3: No record of Jules Verne tutoring in the US.
> REFLECT: Failed to resolve the contradictions, defaulting to the strongest Search 1 candidate.
> Final Answer Produced: Verne
--------------------------------------------------------------------------
Result: [Incorrect] (Ground Truth: Reclus)
```

D.5 Base Model Rollouts: HotpotQA (Multi-Hop Retrieval)

Task: HotpotQA (qid: 5ae7535c5542997b22f6a6d8) – Ground Truth: 1998

The 53rd National Hockey League All-Star Game took place at the indoor arena that was completed in what year?

Sample 1

```
<thinking>
  To solve the question about the 53rd NHL All-Star Game and the year the indoor arena was
      completed, we need to gather relevant information about the event and the specific arena. A
      single agent can retrieve the necessary information and provide a final answer.
</thinking>
<agent>
  <agent_id>SEARCH</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Retrieves recent and cited factual information from the
      internet.</agent_description>
  <required_arguments>
    <agent_input>The 53rd National Hockey League All-Star Game took place at the indoor arena
      that was completed in what year? Include a citation.</agent_input>
```

```
  </required_arguments>
</agent>
<agent>
  <agent_id>FINAL</agent_id>
  <agent_name>CoTAgent</agent_name>
  <agent_description>Write the final, user-ready answer with a citation.</agent_description>
  <required_arguments>
    <agent_input>Summarize the information about the 53rd NHL All-Star Game and the year the
      indoor arena was completed, including the citation.

\${SEARCH}</agent_input>
  </required_arguments>
</agent>
<edge>
  <from>SEARCH</from>
  <to>FINAL</to>
</edge>

--------------------------------------------------------------------------
> Executing WebSearchAgent [SEARCH]...
> SEARCH: Retrieved articles about the 2003 NHL All-Star Game at the Office Depot Center. Search
     engine failed to resolve the secondary "completion year" hop from the monolithic query
     string.
> Executing CoTAgent [FINAL]...
> FINAL: Secondary hop information missing from context.
> Final Answer Produced: 2003
--------------------------------------------------------------------------
Result: [Incorrect] (Ground Truth: 1998)
```

D.6 Our Trained Orchestrator Rollouts: HotpotQA (Multi-Hop Retrieval)

Task: HotpotQA (qid: 5ae7535c5542997b22f6a6d8) – Ground Truth: 1998

The 53rd National Hockey League All-Star Game took place at the indoor arena that was completed in what year?

Trained Model: Sample 8 (Sequential Search-to-CoT Pipeline)

```
<thinking>
  This question requires factual information about the year the arena where the 53rd National
     Hockey League All-Star Game took place was completed. This specific historical and niche
     information cannot be reliably recalled from the model's static knowledge. A single agent
     cannot provide this historical and precise data without risking inaccuracy. Using a
     WebSearchAgent will ensure we get the correct and up-to-date information. The FINAL agent
     will synthesize this information into a user-ready format.
</thinking>

<agent>
  <agent_id>SEARCH</agent_id>
  <agent_name>WebSearchAgent</agent_name>
  <agent_description>Retrieves recent and cited factual information from the
     internet.</agent_description>
  <required_arguments>
    <agent_input>The 53rd National Hockey League All-Star Game took place at the indoor arena
     that was completed in what year? Include a citation.</agent_input>
  </required_arguments>
</agent>

<agent>
  <agent_id>FINAL</agent_id>
  <agent_name>CoTAgent</agent_name>
  <agent_description>Write the final, user-ready answer citing the source.</agent_description>
  <required_arguments>
    <agent_input>Summarize the year the indoor arena where the 53rd National Hockey League
     All-Star Game took place was completed, and provide the source citation.

    \${SEARCH}</agent_input>
  </required_arguments>
</agent>
```

```
<edge>
  <from>SEARCH</from>
  <to>FINAL</to>
</edge>

--------------------------------------------------------------------------
> Executing WebSearchAgent [SEARCH]...
> SEARCH: The 53rd NHL All-Star Game was played on Feb 2, 2003, at the Office Depot Center (now
     Amerant Bank Arena) in Sunrise, Florida. Construction on the arena was completed in 1998.
> Executing CoTAgent [FINAL]...
> FINAL: Synthesized search context. The arena (Office Depot Center) was completed in 1998.
> Final Answer Produced: 1998
--------------------------------------------------------------------------
Result: [Correct] (Ground Truth: 1998)
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize our contributions and scope, and all claims are supported by our experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the Conclusion section and Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not include any theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the implementation details are included in the Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The data and code are released.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: They are included in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report mean $\pm$ standard deviation computed via bootstrap resampling of evaluation questions (Table 5).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The specifications are included in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics, and our research conforms with it in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The discussion of broader impacts is included in Appendix A.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models and datasets are used in compliance with their respective licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release our code and preference dataset with documentation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing is involved in this research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are only involved for writing improvement.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.