ADAROUTER: LEARNING TO ADAPT PLANS FOR AGENTIC ROUTING VIA MULTIPLE LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic routing coordinates calls to heterogeneous large language models (LLMs) and is promising to balance the task performance and inference cost. However, contrary to human problem solving, existing methods for agentic routing lack the ability of reconsideration with feedback evidence gradually acquired during task execution. In this paper, we propose AdaRouter, an adaptive planning framework incorporating reconsideration with available evidence to bridge the gap between human problem solving and agentic routing by LLMs. Specifically, we design two mechanisms for reconsideration, *i.e.*, REFINE that incorporates the new evidence into the query for the next task and REVISE that changes the infeasible plan by updating the remaining tasks or their execution order, while maintaining a coherent guidance across worker calls. Furthermore, we develop a two-level strategy specifically for optimizing AdaRouter with the REFINE and REVISE mechanisms. We achieve trajectory-level GRPO to optimize the overall planning-and-routing behavior and token-level OPSD to distill task-relevant problem-solving skills into routing policy. Extensive experiments across five domains ranging from mathematics to general knowledge and multi-hop reasoning demonstrate that AdaRouter achieves state-of-the-art performance with substantially reduced worker calls in comparison to the strongest routing baseline.

1 INTRODUCTION

Routing among heterogeneous large language models (LLMs) seeks to balance the task performance and inference cost by selectively invoking models with different capabilities (Shnitzer et al., 2023; Hu et al., 2024; Chen et al., 2024). Beyond a single-round model selection, recent work studies agentic routing, where a router interleaves reasoning and worker calls (Zhang et al., 2026; Shao et al., 2025). Related approaches incorporate the task decomposition and workflow generation to organize calls to heterogeneous workers (Feng et al., 2026b). These developments broaden the scope of routing from the model selection to the coordination of task execution across heterogeneous workers. A broader discussion of LLM routing and multi-agent orchestration is provided in Appendix A.

Nevertheless, existing methods for agentic routing overlook reconsideration by either determining subsequent worker calls from the evolving interaction context or organizing them according to a task structure specified before execution. This differs from the problem solving process of a human, where a high-level plan guides the local actions towards an overall goal with continuously arriving new information prompting reconsideration of the plan. In fact, reconsideration is also studied in agent planning to balance the commitment to a plan during execution (Schut & Wooldridge, 2001; Paglieri et al., 2025). In agentic routing, an initial plan is tentative without support by worker observations, and cannot be necessarily justified with the success or failure to complete a task. Therefore, it is crucial to simultaneously maintain a coherent guidance across worker calls and reconsider the planning of remaining tasks with feedback evidence in execution.

Motivated by this, we propose AdaRouter, an adaptive planning framework for agentic routing that incorporates an explicit plan for guiding worker calls with online reconsideration with new evidence observed by the workers. We design two mechanisms for reconsideration, *i.e.*, REFINE that incorporates the new evidence into the query for the next task while preserving the task structure

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

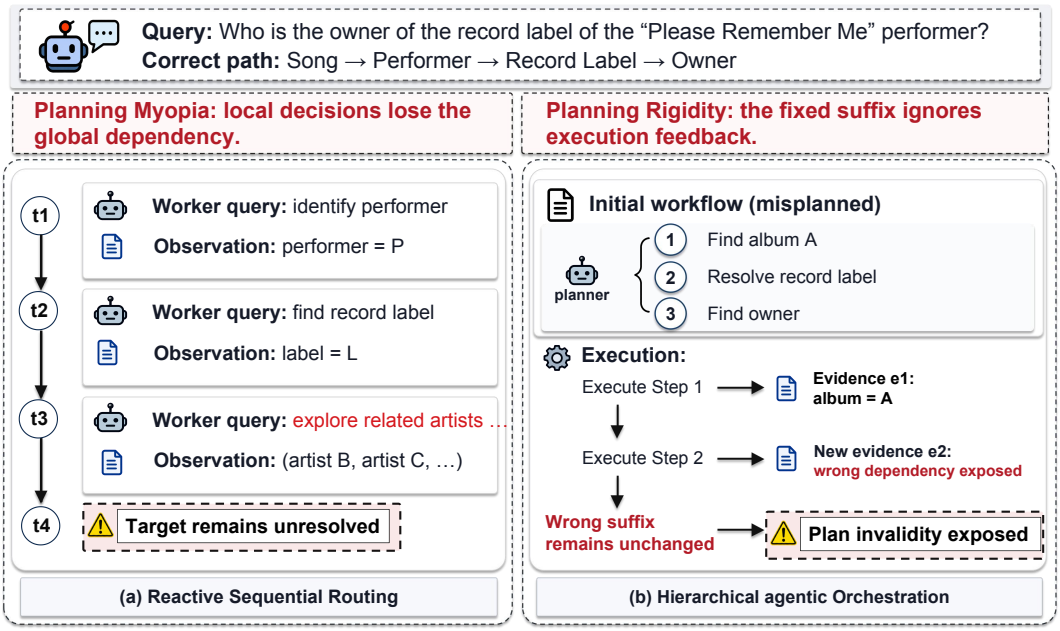


Figure 1: Comparison of two existing agentic routing paradigms and their typical failure modes. Reactive sequential routers make sequential routing decisions without persistent global planning guidance, which can lead to planning myopia, including short-sighted exploration and unnecessary backtracking. Hierarchical agentic orchestration constructs a task structure before execution, but insufficient reassessment under execution feedback can lead to planning rigidity and a mismatch between the remaining workflow and evolving information needs.

and REVISE that updates the remaining tasks or their execution order to address an infeasible plan. These mechanisms allow the router to use feedback within an existing plan or change the plan when necessary to coordinate worker calls throughout execution.

Furthermore, introducing REFINE and REVISE makes the optimization of agentic routing non-trivial, since plan adaptation simultaneously changes the current worker call and subsequent execution trajectory. Thus, it is difficult to attribute the final outcome to individual planning and routing decisions. We address this credit-assignment challenge with complementary supervision at two levels. At the trajectory level, we leverage GRPO (Shao et al., 2024) to optimize the overall planning-and-routing behavior using rewards for task performance, worker cost, and plan-adaptation quality. At the token level, we achieve OPSD to provide denser guidance by distilling task-relevant problem-solving skills from a skill-conditioned teacher into the same router policy, following SDAR (Lu et al., 2026). Overall, they provide global trajectory-level optimization and localized token-level guidance for learning when and how to REFINE or REVISE a plan.

We evaluate AdaRouter across five domains, including mathematics, code generation, world knowledge, general knowledge, and multi-hop reasoning. AdaRouter achieves a macro-average accuracy of 65.4% and outperforms the strongest routing baseline by 9.5% with substantially fewer worker calls than the evaluated multi-round baselines. Ablation studies are performed to validate the effectiveness of adaptive planning and training framework. Trajectory analyses characterize how the router revises and executes its plans under worker feedback, and show fewer unnecessary plan revisions and improved usefulness of the first worker call after revision. Further evaluations demonstrate generalization to unseen tasks and expanded worker pools without additional fine-tuning.

The contributions of this paper are summarized below.

- We propose AdaRouter, an adaptive planning framework that coordinates the worker calls through an explicit plan and designs the REFINE and REVISE mechanisms for learning to refine task queries or revise the remaining plan with newly arriving evidence.

- We develop two-level complementary optimization of agentic routing under the REFINE and REVISE mechanisms. We achieve trajectory-level GRPO to optimize the overall planning-and-routing behavior and token-level OPSD to distill task-relevant problem-solving skills into routing policy.
- We demonstrate state-of-the-art performance across five domains and allow generalization to the unseen tasks and expanded worker pools for AdaRouter. We validate the effectiveness of plan revision and training framework with comprehensive ablation studies and trajectory analyses.

2 PRELIMINARIES AND MOTIVATION

Agentic routing. Routing among multiple large language models (LLMs) aims to balance the answer correctness and inference cost. Single-round routers select a model for each input problem (Shnitzer et al., 2023; Hu et al., 2024; Chen et al., 2024), while the multi-round routers use intermediate observations to guide the subsequent model calls (Zhang et al., 2026; Shao et al., 2025). Recent approaches further incorporate the task decomposition and workflow generation to coordinate heterogeneous workers (Feng et al., 2026b). In this work, we study the agentic routing as a multi-turn decision process in which a router iteratively formulates queries, selects workers, and incorporates the returned observations until a final answer is produced.

Router–worker interaction. Given an input problem x, a router with policy π_θ interacts with a closed pool of frozen workers $\mathcal{M} = \{M_1, \ldots, M_K\}$ with heterogeneous capabilities and inference costs. At turn t, the router uses the input problem and the interaction history to select a worker M_t and formulate a self-contained query q_t. The resulting worker call is $a_t = (M_t, q_t)$, where $M_t \in \mathcal{M}$. The selected worker returns an observation $o_t = M_t(q_t)$ at cost c_t. The router incorporates this observation into the interaction history and continues until it produces a final answer $\hat{y}$ or reaches the turn budget. The complete interaction forms a trajectory $\tau \sim \pi_\theta(\cdot \mid x)$, comprising all the router-generated content and worker observations. The objective is to balance the answer correctness and the total worker cost $\sum_{t=1}^{T} c_t$, where T is the number of worker calls.

Motivation for adaptive planning. An initial plan reflects a tentative judgment about the tasks needed to solve the input problem and their execution order. Worker observations provide evidence for assessing this judgment, but the task completion alone does not determine whether the remaining plan is still appropriate. As illustrated in Table 1, the result of a completed task may provide evidence needed by the next task or contradict a premise underlying later tasks. When the current task is not completed, the observation may leave plan suitability unclear or reveal a dependency on a task scheduled later in the plan. These observations have different implications for subsequent work despite the same task completion status. This motivates reconsidering the remaining plan in light of worker feedback: preserving its tasks and order when they remain appropriate, and revising the unresolved portion when they do not.

Table 1: Task completion and plan suitability.

Current task completed?	Worker observation	Implication for the remaining plan
✓	Evidence obtained	Next task can proceed **(A)**
✓	Later premise contradicted	Planned path unsupported **(B)**
×	No usable result	Plan suitability unclear **(C)**
×	Depends on a later task	Task order unsuitable **(D)**

The current task is the task assigned in this worker call.
A–D: appendix cases.

3 METHODOLOGY

In this section, we present AdaRouter, an adaptive planning paradigm for agentic routing, and its corresponding training framework. We first describe how an explicit plan guides the multi-turn routing and is adapted in response to worker observations (§3.1). We then describe the joint training

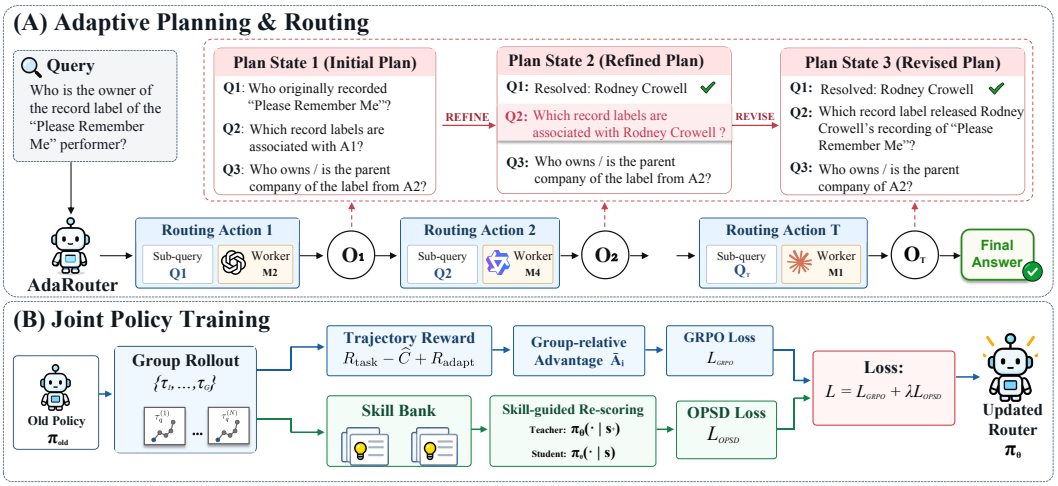


Figure 2: Overview of AdaRouter. (A) Adaptive Planning & Routing: the router constructs a persistent solution plan and executes plan-conditioned routing actions. Each worker observation is fed back to REFINE the next task while preserving the current plan structure or REVISE the unfinished plan when new evidence changes future information needs. (B) Joint Policy Training: GRPO jointly trains planning, routing, and adaptation using task, cost, and plan-adaptation rewards, while OPSD re-scores the same sampled actions under original and skill-augmented contexts to provide dense token-level supervision. The two objectives jointly update a single planning-and-routing policy.

framework, which incorporates trajectory-level reinforcement learning with GRPO (§3.2) and skill-guided on-policy self-distillation for the token-level guidance (§3.3).

3.1 Adaptive Planning and Routing

As shown in Fig. 2, AdaRouter constructs an explicit plan specifying the tasks required to solve the input problem and their execution order. For the simple problems that require no decomposition, the plan contains a single task. Following this plan, the router selects a worker for each task and formulates a self-contained query that specifies the task and includes the available evidence needed to complete it. The router then incorporates the returned observation into the interaction history for the subsequent tasks.

After each observation, the router assesses whether the remaining plan is still appropriate for solving the input problem. If so, the router performs Refine. It uses the available evidence to formulate a self-contained query for the next unresolved task, while preserving the objectives, number, order, and dependencies of the remaining tasks. Thus, Refine updates the query used to execute a task without changing the task itself.

When the observation shows that the remaining tasks or their execution order need to change, the router performs Revise. It preserves the completed prefix and modifies only the unresolved portion, which may involve changing task objectives or adding, removing, and reordering tasks. Changing the worker or merely rephrasing a query does not constitute plan revision. Subsequent routing follows the retained or revised plan, using the available evidence until a final answer is produced. The trajectory representation and inference workflow are detailed in Appendices I and K, with representative trajectories in Appendix M.

3.2 Trajectory-level Reinforcement Learning

Planning and routing decisions jointly affect the correctness of the final answer and the total cost of worker calls. We therefore optimize the router by using group relative policy optimization (GRPO) (Shao et al., 2024) with the trajectory-level rewards. For each input problem x, we sample a group of G complete multi-turn trajectories using the old router policy π_{old}:

$$\{\tau_i\}_{i=1}^{G} \overset{\text{i.i.d.}}{\sim} \pi_{\text{old}}(\cdot \mid x), \tag{1}$$

where τ_i denotes the i-th trajectory. The terminal reward combines the answer correctness, the normalized cost of worker calls, and the plan-adaptation quality, with a fixed penalty for the invalid output formats:

$$R_i = \begin{cases} (1-\alpha)R_{\text{task},i} - \alpha\widehat{C}_i + \rho R_{\text{adapt},i}, & \text{valid}(\tau_i) = 1, \\ -1, & \text{valid}(\tau_i) = 0, \end{cases} \qquad R_{\text{adapt},i} = R_{\text{refine},i} + R_{\text{revise},i}, \tag{2}$$

where $R_{\text{task},i}$ measures the correctness of the final answer, and $\widehat{C}_i$ is the normalized total cost of worker calls in τ_i. The coefficient α controls the trade-off between the correctness and cost, while ρ controls the contribution of the plan-adaptation rewards. The indicator $\text{valid}(\tau_i)$ equals one if the trajectory follows the required output format, and zero otherwise.

The refinement reward $R_{\text{refine},i}$ evaluates REFINE events in which the next unresolved task depends on evidence from the preceding observation. An event passes if the query correctly incorporates the required evidence, is self-contained, and introduces no unsupported information.

The revision reward $R_{\text{revise},i}$ evaluates every REVISE event. A passing event must satisfy two requirements. First, the observation makes the remaining plan unsuitable, and the revised plan addresses the identified issue. Second, the first query under the revised plan is consistent with its first task, is grounded in the available evidence, and is self-contained; the returned observation provides useful evidence for that task.

Each reward component equals one when at least one event is evaluated and all evaluated events pass, and zero otherwise. Appendix C specifies the evaluator inputs and trajectory-level aggregation.

The rewards are standardized within each group to obtain the trajectory-level advantage:

$$A_i = \frac{R_i - \mu_x}{\sigma_x + \varepsilon}, \tag{3}$$

where μ_x and σ_x are the mean and standard deviation of the G rewards for input problem x, and $\varepsilon > 0$ ensures numerical stability. All policy-generated tokens in τ_i share the same advantage A_i. Let $\mathcal{T}_i$ denote their positions. For each $\ell \in \mathcal{T}_i$, $y_{i,\ell}$ is the sampled token and $s_{i,\ell}$ is its generation context, including the input problem, preceding interaction history, and current-turn token prefix. Worker observations are included in the context but excluded from the loss. The probability ratio between the current and old policies is $r_{i,\ell} = \pi_\theta(y_{i,\ell} \mid s_{i,\ell})/\pi_{\text{old}}(y_{i,\ell} \mid s_{i,\ell})$. We minimize the GRPO objective:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{i,\ell\in\mathcal{T}_i}\Big[\min\{r_{i,\ell}A_i,\ \text{clip}(r_{i,\ell}, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})A_i\} - \eta\widehat{D}_{\text{KL},i,\ell}\Big], \tag{4}$$

where ϵ_{clip} is the clipping parameter, and $\widehat{D}_{\text{KL},i,\ell}$ estimates the KL divergence from the current policy π_θ to the reference policy π_{ref}, with regularization coefficient η.

3.3 SKILL-GUIDED ON-POLICY SELF-DISTILLATION

Learning feedback-driven plan adaptation requires the router to determine when to refine the query for the next unresolved task and when to revise the remaining plan. However, a shared trajectory-level advantage provides limited guidance on which of these decisions contributes to the final outcome. To support both these decisions and the execution of the retained or revised plan, we use OPSD to provide additional token-level guidance from task-relevant skills.

These skills summarize reusable problem-solving knowledge extracted from training trajectories and are stored in a static skill bank. We reuse the trajectories sampled by π_{old} without skills. Following SDAR (Lu et al., 2026), we construct the teacher π_T by adding a retrieved skill to the context of the frozen old router policy. The router policy π_θ serves as the student and uses the original context. Both compute probabilities for the same sampled tokens. We denote the teacher context by $s^+_{i,\ell}$, obtained by augmenting $s_{i,\ell}$ with the retrieved skill c^+_i. Details of the skill bank are provided in Appendix G.

We compute the teacher–student log-probability gap on each sampled token:

$$\Delta_{i,\ell} = \log \pi_T(y_{i,\ell} \mid s^+_{i,\ell}) - \log \pi_\theta(y_{i,\ell} \mid s_{i,\ell}). \tag{5}$$

Table 2: **Model performance comparison with router baselines across five scenarios.** Average Acc is the macro-average over the five domains, and total cost is reported over the full test set. **Bold** and underline indicate the best and second-best results.

Setting	Math	Code	Popular	WK	Multi-hop	Overall		
	Acc	Acc	Acc	Acc	Acc	Avg. Acc	Total cost (USD)	Avg. worker calls
Single-round routers								
Router-KNN	67.0%	63.8%	34.0%	47.0%	39.3%	50.2%	0.0744	1
Router-SVM	53.0%	63.8%	34.0%	42.0%	39.3%	46.4%	0.0813	1
Router-MLP	62.0%	63.8%	35.0%	39.0%	39.3%	47.8%	0.0728	1
RouterDC	67.0%	66.1%	45.0%	49.0%	42.0%	53.8%	0.0724	1
GraphRouter	48.0%	58.5%	30.0%	35.0%	30.0%	40.3%	0.0808	1
ICL-Router	72.0%	68.1%	47.0%	53.0%	39.3%	55.9%	0.0737	1
Multi-round routers								
Prompt-LLM	25.0%	57.4%	35.0%	47.0%	38.6%	40.6%	0.5613	12.4
AutoMix	62.0%	69.1%	49.0%	42.0%	39.3%	52.3%	0.1194	2.9
R2-Reasoner	52.0%	52.1%	34.0%	60.0%	38.6%	47.3%	0.3373	5.4
Router-R1	49.0%	53.2%	31.0%	56.0%	40.0%	45.8%	0.0360	1.8
GraphPlanner	67.0%	62.8%	50.0%	54.0%	42.6%	55.3%	0.3174	9.8
AdaRouter (RL only)	74.0%	79.8%	51.0%	59.0%	48.7%	62.5%	0.2087	1.7
AdaRouter	**77.0%**	**81.6%**	**52.0%**	**65.0%**	**51.3%**	**65.4%**	0.1654	1.5

A positive $\Delta_{i,\ell}$ indicates that the teacher assigns higher probability to the sampled token than the student. We assign larger auxiliary weights to these tokens and smaller weights to tokens with negative gaps through a detached sigmoid gate:

$$\delta_{i,\ell} = \mathrm{sg}(\Delta_{i,\ell}), \qquad g_{i,\ell} = \mathrm{sg}[\sigma(\kappa\delta_{i,\ell})], \tag{6}$$

where sg denotes stop-gradient, σ is the sigmoid function, and κ controls the gate scale. The auxiliary loss is

$$\ell_{i,\ell}^{\mathrm{OPSD}} = g_{i,\ell}\left[\mathrm{sg}\Big(\log \pi_T(y_{i,\ell} \mid s_{i,\ell}^{+})\Big) - \log \pi_\theta(y_{i,\ell} \mid s_{i,\ell})\right], \quad \mathcal{L}_{\mathrm{OPSD}} = \frac{\sum_{i,\ell} M_{i,\ell}\ell_{i,\ell}^{\mathrm{OPSD}}}{\sum_{i,\ell} M_{i,\ell}}, \tag{7}$$

where $M_{i,\ell}$ selects policy-generated tokens and is zero for trajectories without a matched skill. Gradients flow only through the student log-probabilities.

The overall training objective is

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{GRPO}}(\theta) + \lambda_{\mathrm{OPSD}}\mathcal{L}_{\mathrm{OPSD}}(\theta), \tag{8}$$

where λ_{OPSD} controls the contribution of the auxiliary self-distillation loss. On the same sampled trajectories, GRPO provides a shared trajectory-level advantage, while OPSD provides auxiliary guidance with token-dependent weights. The reward definition and the GRPO loss remain unchanged. Skills are used only in the teacher context during training and are not required at inference. More details on the reinforcement-learning objectives can be found in Appendix C.

4 Experiments

To assess the effectiveness of AdaRouter, we evaluate it on a diverse suite of benchmarks spanning multiple task domains and compare it against representative single-round and multi-round routing baselines. We first describe the experimental setup and evaluation protocol before presenting the empirical results. More implementation details can be found in Appendix D.

Datasets. To comprehensively evaluate the capability of AdaRouter, we first conduct in-domain experiments on 11 widely used LLM reasoning and question-answering benchmarks covering five task domains: (1) mathematical reasoning: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021); (2) code generation: HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021); (3) Popular: MMLU (Hendrycks et al., 2020) and GPQA (Rein et al., 2023); (4) multi-hop question answering: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue

(Trivedi et al., 2022); and (5) WK: Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017). In addition, to evaluate the generalization ability of the routing policy beyond the training distribution, we further include LogicGrid (Mitra & Baral, 2015), MGSM (Shi et al., 2022), CommonsenseQA (Talmor et al., 2019), and CommonGen (Lin et al., 2020) as out-of-distribution test sets, which respectively evaluate logical reasoning, multilingual mathematical reasoning, commonsense question answering, and commonsense generation. These four datasets are used exclusively for evaluation, ensuring that the router is tested on genuinely unseen domains and enabling a rigorous assessment of its generalization ability.

Routing infrastructure. All main variants of AdaRouter use Qwen3-4B-Instruct (Yang et al., 2025) as the trainable router. The worker pool consists of eight frozen heterogeneous LLMs ranging from 7B to 119B parameters, including Llama-3.1 (8B) (Grattafiori et al., 2024), Ministral (8B) (Liu et al., 2026), Qwen2.5 (7B) (Qwen et al., 2025), Ministral (14B) (Liu et al., 2026), Gemma-4 (31B) (Team et al., 2026), Qwen3-VL (30B) (Bai et al., 2025), Mistral-Small-4 (119B), and Llama-3.3 (70B) (Grattafiori et al., 2024). We further summarize the model scale, input price, and output price of each LLM in Appendix Table 11.

Baselines and evaluation metrics. We evaluate a variety of baseline methods across five experimental scenarios. These baselines are divided into two categories: (a) single-round routers (Shnitzer et al., 2023; Hu et al., 2024; Chen et al., 2024; Feng et al., 2025; Wang et al., 2026a), which route each query through a single LLM invocation; and (b) multi-round routers (Aggarwal et al., 2024; Shao et al., 2025; Zhang et al., 2026; Feng et al., 2026b), which solve a query through multiple LLM invocations. Following shared cost-reporting practice (Feng et al., 2026b), we evaluate all routing methods using accuracy (Acc) and total worker cost over the test set (USD). Here, Acc denotes the task-specific evaluation metric described in Appendix Table 9. For every method, total cost sums the worker costs across all test queries, using provider-returned input and output token counts and the same fixed token prices. More details on aggregation and efficiency reporting conventions can be found in Appendix H.

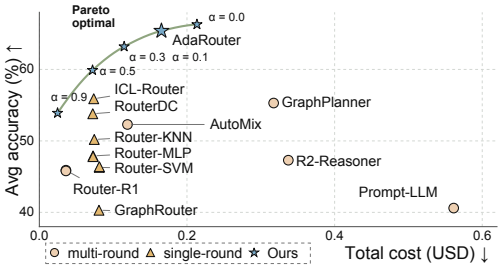


Figure 3: Compared with baseline routers, AdaRouter achieves consistently favorable accuracy–cost trade-offs across different operating points. We evaluate AdaRouter with $\alpha \in 0.0, 0.1, 0.3, 0.5, 0.9$ alongside representative single-round and multi-round routers.

4.1 Main Results

We compare AdaRouter with six single-round routers and five multi-round routers. The results are reported in Table 2, from which we make the following observations.

AdaRouter attains SOTA results across diverse scenarios. Across the five evaluated domains, AdaRouter achieves the best results among all compared methods and the highest average accuracy of 65.4%, exceeding the strongest baseline by 9.5 percentage points. Its advantage extends across both single-round routers and multi-round systems, including baselines that already coordinate repeated worker interactions. The gains are especially pronounced in Code and Multi-hop, where AdaRouter improves over the strongest baseline in each domain by 12.5 and 8.7 percentage points, respectively. These tasks require coordinating intermediate reasoning steps and using their results to determine subsequent queries, making them particularly relevant to feedback-driven planning. The observed gains are consistent with the ability of AdaRouter to maintain an explicit solution plan while adapting its unresolved steps to execution feedback. Popular shows a more modest improvement, whereas the advantages in Math and WK further extend the coverage of these gains. Together, the results establish the effectiveness of AdaRouter across diverse routing tasks, with particularly strong improvements in code generation and multi-hop reasoning. Table 2 also shows that AdaRouter achieves higher accuracy than every evaluated multi-round baseline while making fewer worker calls on average. Compared with GraphPlanner and Prompt-LLM, it further reduces total

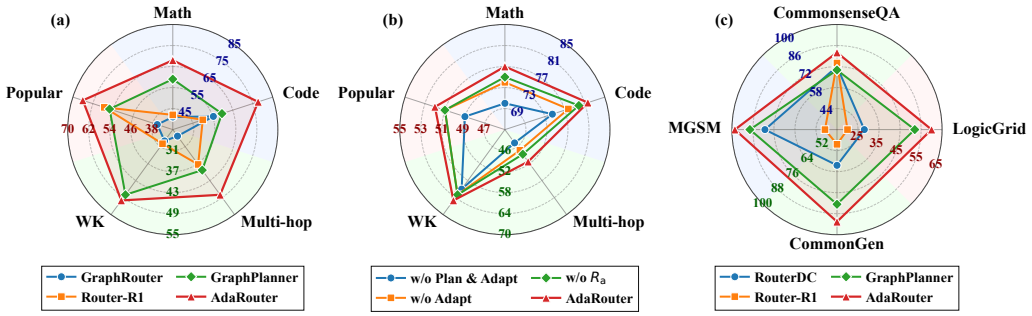


Figure 4: **Generalization and planning ablations of AdaRouter.** (a) Zero-shot generalization to unseen LLMs. (b) Planning and adaptation ablations. (c) Zero-shot generalization to unseen tasks.

worker cost. These results demonstrate that AdaRouter combines stronger routing performance with more economical worker use, achieving its accuracy advantage with fewer worker interactions.

AdaRouter effectively balances performance and cost. As shown in Figure 3, AdaRouter consistently lies on the Pareto frontier, offering more favorable accuracy–cost trade-offs than both single-round and multi-round routers. By adjusting α, AdaRouter flexibly trades off across operating points ranging from high-accuracy, high-cost configurations to low-cost, lightweight configurations. Compared with baseline methods, AdaRouter achieves higher accuracy at the same cost or lower cost at the same accuracy, demonstrating a more efficient and controllable performance–cost trade-off.

4.2 Generalization to unseen LLMs and Tasks

A key challenge in router design is whether the learned routing policy can generalize beyond the training distribution and adapt to entirely new tasks. To investigate this question, we evaluate AdaRouter under zero-shot settings on held-out tasks and analyze its robustness and adaptability.

AdaRouter effectively adapts to unseen LLMs in a zero-shot setting. As illustrated in Figure 4(a), AdaRouter demonstrates strong adaptability when evaluated with unseen LLMs not introduced during training. Compared with GraphRouter and Router-R1, AdaRouter consistently achieves superior performance across all task domains, indicating that its routing strategy generalizes effectively to new backbone models without additional fine-tuning. This highlights the robustness of AdaRouter in handling zero-shot scenarios where the underlying LLMs differ from those seen in training.

AdaRouter generalizes robustly to unseen tasks. As shown in Figure 4(c), AdaRouter demonstrates strong zero-shot generalization ability across LogicGrid, MGSM, CommonsenseQA, and CommonGen. The figure compares AdaRouter with representative single-round and multi-round routing baselines under the same zero-shot protocol. The results demonstrate that AdaRouter transfers to diverse unseen tasks without additional fine-tuning.

4.3 Ablation Studies

Figure 4(b) ablates the planning structure and adaptation reward under the same training and evaluation setup. w/o Plan&Adapt removes the explicit plan and both plan-adaptation operations; w/o Adapt retains the plan and evidence-grounded refinement of its next unresolved step, but disables structural revision of the unresolved plan; and w/o R_a retains both operations while removing the adaptation reward. Table 3 compares the base model with successive GRPO and OPSD training.

Effect of explicit planning and dynamic adaptation. Figure 4(b) shows that removing both the explicit plan and adaptation operations causes the largest accuracy drop across all five domains. Retaining the plan and REFINE recovers part of this loss, supporting the value of organizing worker calls around an explicit solution structure and grounding subsequent queries in resolved evidence. The full model further improves performance across all five domains by additionally assessing the plan under worker feedback and revising it when needed, indicating the benefit of feedback-conditioned plan reassessment beyond execution within a fixed plan structure. Removing the plan-adaptation

Table 3: Training stages of AdaRouter. + GRPO trains the full planning mechanism, including REFINE and REVISE, using the composite reward in Eq. (2). + OPSD adds skill-guided self-distillation.

Domain	AdaRouter Base	+ GRPO	+ OPSD
Math	63.0%	74.0%	77.0%
Code	59.6%	79.8%	81.6%
Popular	48.0%	51.0%	52.0%
WK	55.0%	59.0%	65.0%
Multi-hop	39.3%	48.7%	51.3%

Table 4: Plan-adaptation quality on Multi-hop. Unnec. Rev.: proportion of valid-plan decisions that unnecessarily revise. First Useful: proportion of first routes after revision yielding useful evidence. Recovery: final accuracy on trajectories requiring repair.

Metric	AdaRouter Base	+ GRPO	+ OPSD
Unnec. Rev.↓	5.1%	0.9%	0.9%
First Useful↑	10.0%	12.0%	15.6%
Recovery↑	17.0%	20.0%	22.9%

reward also lowers accuracy while leaving both operations available, showing that supervision of adaptation quality helps the router use these operations effectively.

Effect of training objectives. Table 3 shows that adding GRPO and then OPSD consistently improves accuracy across all five domains, supporting the complementary roles of trajectory-level and token-level supervision. GRPO evaluates complete routing trajectories through the composite task, cost, and plan-adaptation reward, jointly optimizing planning, worker selection, and revision toward better trajectory outcomes. Within each rollout, however, policy-generated tokens share the same group-relative advantage, providing limited guidance on which internal decisions deserve stronger reinforcement. OPSD supplies dense token-level supervision by re-scoring the same sampled tokens with a skill-conditioned teacher and using the teacher–student log-probability gap to modulate their learning weights. The Multi-hop behavior analysis in Table 4 is consistent with this distinction: GRPO markedly reduces unnecessary revisions, while OPSD further improves the usefulness of the first route after revision and recovery accuracy without changing the unnecessary-revision rate. These observations suggest that trajectory-level optimization helps control inappropriate adaptation, whereas dense skill-guided supervision further improves how the router acts on a revised plan.

Analysis of Model Scale. The router uses Qwen3-4B-Instruct in our main configuration. Figure 5 examines whether the proposed training framework scales to a larger controller by repeating GRPO and the full AdaRouter training on a Qwen2.5-7B-Instruct router under the same worker pool, prompt format, and inference budget. Both GRPO and AdaRouter remain effective with the 7B backbone across all five domains, while the relative improvement from GRPO to the full AdaRouter is consistently preserved. These results suggest that the gains of AdaRouter are not specific to the Qwen3-4B router, but transfer across router backbones of different scales.

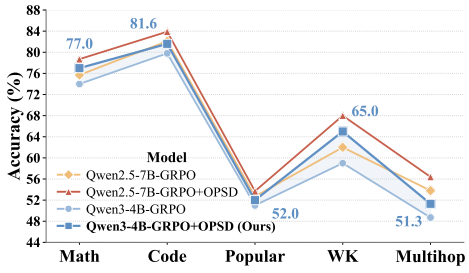


Figure 5: Router-backbone comparison: accuracy across five domains.

5 CONCLUSION

We identify two common failure modes in agentic routing for complex multi-model collaboration: intent drift under step-wise reactive routing and plan rigidity under established task structure. To address these problems, we propose AdaRouter, a paradigm for stable high-level intent with feedback-driven tasking, in which the router stabilizes the overall solution direction through explicit planning, while feedback-driven tasking grounds worker queries and reconstructs the unfinished plan according to execution observations. A training framework combining trajectory-level reinforcement learning and skill-guided self-distillation enables the LLM router to acquire the AdaRouter paradigm. Extensive experiments show that AdaRouter produces more coherent multi-step routing and adapts more effectively to execution feedback, yielding a better performance–cost trade-off.

AI USE STATEMENT

Large Language Models (LLMs) were used to assist with the writing and polishing of the manuscript, including sentence rephrasing, grammar checking, and improving clarity and readability. The LLM was not involved in the ideation, research methodology, or experimental design. All research concepts, implementations, experiments, analyses, and scientific conclusions were developed and conducted by the authors. The authors reviewed all AI-assisted text and take full responsibility for the final content of the manuscript.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. It involves no human-subject or animal experiments and introduces no new collection of personal data. All datasets used in our experiments, including GSM8K, MATH, HumanEval, MBPP, MMLU, GPQA, HotpotQA, 2WikiMultiHopQA, MuSiQue, Natural Questions, TriviaQA, LogicGrid, MGSM, CommonsenseQA, and CommonGen, are publicly available research datasets and were used in accordance with their intended research purposes and applicable usage guidelines. No personally identifiable information was collected or used in this study.

REPRODUCIBILITY STATEMENT

We provide the information necessary to reproduce our experiments in the main paper and appendices, including dataset and evaluation protocols, router and worker configurations, training objectives, reward definitions, prompts, hyperparameters, and hardware settings. All datasets used in this study are publicly available.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. Automix: Automatically mixing language models. *Advances in Neural Information Processing Systems*, 37:131000–131034, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37:66305–66328, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, et al. Multi-agent collaboration via evolving orchestration. *Advances in neural information processing systems*, 38:165025–165059, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026a.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. In *International Conference on Learning Representations*, volume 2025, pp. 26186–26203, 2025.

Tao Feng, Haozhen Zhang, Zijie Lei, Peixuan Han, and Jiaxuan You. Graphplanner: Graph memory-augmented agentic routing for multi-agent llms. *arXiv preprint arXiv:2604.23626*, 2026b.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 87362–87388, 2026.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. Commongen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1823–1840, 2020.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Arindam Mitra and Chitta Baral. Learning to automatically solve logic grid puzzles. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1023–1033, 2015.

Davide Paglieri, Bartłomiej Cupiał, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning when to plan: Efficiently allocating test-time compute for llm agents. *arXiv preprint arXiv:2509.03581*, 2025.

A Yang Qwen, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengpeng Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, pp. 2412–15115, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Martijn Schut and Michael Wooldridge. Principles of intention reconsideration. In *Proceedings of the fifth international conference on Autonomous agents*, pp. 340–347, 2001.

Chenyang Shao, Xinyang Liu, Yutang Lin, Fengli Xu, and Yong Li. Route-and-reason: Scaling large language model reasoning with reinforced model router. *arXiv preprint arXiv:2506.05901*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Chenxu Wang, Hao Li, Yiqun Zhang, Linyao Chen, Jianhao Chen, Ping Jian, Qiaosheng Zhang, and Shuyue Hu. Icl-router: In-context learned model representations for llm routing. In *Proceedings of the 40th AAAI Conference on Artificial Intelligence*, pp. 33413–33421, 2026a.

Jiayu Wang, Yifei Ming, Zixuan Ke, Shafiq Joty, Aws Albarghouthi, and Frederic Sala. Skillorchestra: Learning to route agents via skill transfer. *arXiv preprint arXiv:2602.19672*, 2026b.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. Wideseek-r1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning. *arXiv preprint arXiv:2602.04634*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. Masrouter: Learning to route llms for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15549–15572, 2025.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*, 2025.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 141233–141265, 2026.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025a.

Wentao Zhang, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *arXiv e-prints*, pp. arXiv–2506, 2025b.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *arXiv preprint arXiv:2402.16823*, 2024.

A Additional Related Work

Routing and Orchestration for Heterogeneous LLMs Prior work on coordinating heterogeneous LLMs has progressively expanded from one-shot model selection to sequential routing and workflow-level orchestration. Single-round routers assign each query to a model in one step, using query representations or learned selection functions, including RouterKNN and RouterMLP (Shnitzer et al., 2023), RouterSVM (Hu et al., 2024), RouterDC (Chen et al., 2024), and GraphRouter (Feng et al., 2025); ICL-Router instead represents candidate LLMs in context, enabling newly introduced models to be incorporated without retraining the router (Wang et al., 2026a). Multi-round routers extend model selection into a sequential decision process in which intermediate outputs can inform subsequent calls. AutoMix (Aggarwal et al., 2024) performs adaptive routing through self-verification with threshold-based or POMDP-based policies, whereas R2-Reasoner (Shao et al., 2025) and Router-R1 (Zhang et al., 2026) learn multi-step routing policies through reinforcement learning. Beyond sequential model selection, agentic systems introduce explicit task structure and coordination among specialized workers: planner–worker frameworks decompose problems into subtasks (Zhang et al., 2025b), while subsequent work learns or evolves agent compositions, communication topologies, and workflows (Yue et al., 2025; Wang et al., 2026b; Zhang et al., 2025a; Zhuge et al., 2024; Hu et al., 2025; Dang et al., 2026). GraphPlanner (Feng et al., 2026b) further connects LLM routing with agentic orchestration by jointly selecting heterogeneous LLM agents and composing query-specific workflows. Current paradigms face two complementary limitations: multi-round routers adapt local routing decisions to evolving interaction histories without explicitly maintaining a persistent solution plan, while workflow-based agentic routers construct structured coordination schemes but do not explicitly reassess whether the remaining plan is still appropriate under execution feedback. AdaRouter addresses both by maintaining an explicit solution plan and selectively preserving or revising its unresolved portion before subsequent routing decisions.

Agentic Reinforcement Learning and Self-Distillation Recent advances in reinforcement learning for LLMs extend from verifiable reasoning to agentic settings, where agents learn long-horizon behavior through environmental interaction. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) replaces the value network with a group-relative advantage estimator and underpins outcome-reward reasoning models such as DeepSeek-R1 (Guo et al., 2025). However, outcome-level group advantages can be coarse for multi-turn agentic rollouts, where rewards arrive only at termination. A recent line of work addresses this credit-assignment gap with finer-grained signals, including turn-level attribution (Zeng et al., 2025), action-level inner groups (Feng et al., 2026a), tree-advantage variants that reuse partial trajectories across branches (Ji et al., 2026), and breadth-axis extensions that share rollouts across cooperating agents (Xu et al., 2026). A complementary line of work provides dense supervision through on-policy self-distillation (OPSD), where a model acts as both student and teacher under different contexts and the privileged teacher provides token-level guidance on the student's own rollouts. Recent studies further combine reinforcement learning with such privileged self-distillation for multi-turn agents. SDAR (Lu et al., 2026) keeps reinforcement learning as the primary optimization backbone and introduces OPSD as a gated auxiliary objective, using the teacher–student log-probability gap to selectively modulate token-level distillation. Together, these studies motivate combining trajectory-level reinforcement learning for optimizing long-horizon outcomes with finer-grained self-distillation signals for supervising internal planning and routing decisions.

B Additional Multi-Agent Orchestration Baselines

Multi-Agent Orchestration. We additionally evaluate SkillOrchestra (Wang et al., 2026b), MasRouter (Yue et al., 2025), and Puppeteer (Dang et al., 2026), adapting each method to the same worker pool and evaluation protocol. These methods obtain average accuracies of 54.1%, 53.9%, and 51.0%, respectively, compared with 65.4% for AdaRouter. Although these methods organize multi-agent execution through agent selection and coordination, they do not explicitly reassess the remaining task-level plan according to worker feedback. In contrast, AdaRouter directly incorporates such feedback-driven plan adaptation into the routing process. The results suggest that, beyond general-purpose agent coordination, explicit plan reassessment can provide additional benefits for agentic routing. See Table 5 for detailed results.

Table 5: **Comparison with additional multi-agent orchestration baselines.** All methods are adapted to the same worker pool and evaluation protocol. Avg. Acc denotes the macro-average accuracy across the five domains.

Method	Math	Code	Popular	WK	Multihop	Avg. Acc
SkillOrchestra (Wang et al., 2026b)	64.0%	70.2%	41.5%	53.0%	42.0%	54.1%
MasRouter (Yue et al., 2025)	61.0%	71.3%	41.5%	53.0%	42.7%	53.9%
Puppeteer (Dang et al., 2026)	58.0%	66.0%	39.4%	51.0%	40.7%	51.0%
AdaRouter	**77.0%**	**81.6%**	**52.0%**	**65.0%**	**51.3%**	**65.4%**

C AdaRouter Training Details

All variants use the on-policy training loop in Section 3.2: for each task, the router samples G complete trajectories, executes the selected workers, computes the composite reward in Eq. (2), and updates only router-generated tokens.

GRPO. Sampling and group-relative advantages follow Eqs. (1)–(3). We normalize the total worker cost C_i in USD as $\widehat{C}_i = C_i/c_0$, with $c_0 = 10^{-3}$ USD. Zero-variance groups are retained with zero policy-gradient advantage; malformed trajectories receive the format penalty -1, whereas infrastructure failures are resampled and excluded. Every router token in a trajectory shares A_i, and training uses neither a critic nor step-level advantages.

OPSD. OPSD re-scores the same no-skill on-policy rollouts with the frozen old actor under a retrieved skill; unmatched trajectories receive an all-zero OPSD mask. Teacher log-probabilities remain fixed during the update, while the current actor supplies student log-probabilities. Skills are absent from student rollouts, the reference-KL branch, and inference; objective coefficients are reported in Appendix D.

Plan-adaptation rewards. The frozen DeepSeek-V4-Pro evaluator receives serialized planning transitions without the gold answer and supplies R_{refine} and R_{revise}. It evaluates evidence grounding for applicable Refine events and the timing and first routed step of each Revise event. A component equals one only when at least one applicable event occurs and all such events pass; otherwise it is zero. Appendix J provides the evaluator prompt.

D Implementation Details

Hardware. Reinforcement-learning runs use a single node with 2×NVIDIA RTX PRO 6000 Blackwell 96 GB GPUs. The frozen workers are accessed through their provider endpoints, while router rollout generation is served by vLLM (Kwon et al., 2023).

GRPO hyper-parameters and throughput. GRPO (Shao et al., 2024) is run over the 4,890-query training split. Each global step contains 64 distinct query groups with $G = 8$ complete trajectories per query, yielding 512 trajectories before one actor update. We use one optimization epoch, a 32-group mini-batch, maximum assistant turns $T_{\max} = 8$, response budget $L = 16{,}384$, and prompt cap 12,288. Training rollouts use temperature 1.0, top-$p = 1.0$, and top-$k = -1$; validation uses greedy decoding. AdamW uses learning rate 1×10^{-6}, clip ratio $\epsilon = 0.2$, and low-variance reference KL coefficient 10^{-2}. The terminal cost coefficient is $\alpha = 0.1$, and the plan-adaptation coefficient is $\rho = 0.05$. OPSD uses coefficient 0.01 and gate scale 5.0. Only policy-generated assistant tokens carry gradient; prompts, runtime state, worker observations, and role boundaries are loss-masked.

Checkpoint selection. During evaluation, greedy decoding is applied and the best model is selected by validation accuracy, with lower worker cost breaking ties. Each α point in Figure 3 is trained separately from the same initial model with all other settings fixed and follows this checkpoint-selection rule. The baseline points in Figure 3 use the same configurations and evaluation results as those reported in the main comparison in Table 2.

Plan-adaptation metrics. Table 4 pools counts across HotpotQA, 2WikiMultiHopQA, and MuSiQue using single-model offline annotation of the question, residual plan, and pre-decision execution feedback. Unnec. Rev. is the fraction of assessable valid-plan decisions that choose REVISE, excluding states where the answer is already available. First Useful counts first post-revision worker responses providing new, task-relevant evidence divided by assessable REVISE events; refusals and repeated information do not qualify. Recovery measures verifier accuracy among trajectories with at least one repair-needed state, including missed revisions.

E DATASET AND LLM BACKBONE DETAILS

Table 6: **The domains and corresponding tasks used in our experiments.** The suite spans six evaluation categories and 15 tasks; underlined tasks are held out from training to evaluate generalization to unseen tasks.

Domain	Tasks
Math	GSM8K, MATH
Code	MBPP, HumanEval
Multi-hop QA	HotpotQA, 2WikiMultiHopQA, MuSiQue
World Knowledge	Natural Questions, TriviaQA
Popular	MMLU, GPQA
Out-of-domain Testing	LogicGrid, MGSM, CommonsenseQA, CommonGen

Table 7: **The scales and corresponding worker LLMs used in our experiments.** Eleven LLMs are grouped by reported model size; underlined models are excluded from training and added only for unseen-worker evaluation.

Scale	LLMs
Small	Qwen2.5-7B-Instruct, Llama-3.1-8B, Ministral-8B, Ministral-14B, Qwen3-8B
Medium	Qwen3-VL-30B-A3B-Instruct, Gemma-4-31B, Qwen3-32B
Large	Llama-3.3-70B-Instruct, Mistral-Small-4-119B, Qwen3-Next-80B-A3B

Table 8: **Sample counts in the training and test sets across tasks.** In-domain counts follow train–test question deduplication; each out-of-domain task contributes 50 test queries and no training queries.

Domain	Tasks	Train Cases	Test Cases
Math	GSM8K (Cobbe et al., 2021)	500	50
	MATH (Hendrycks et al., 2021)	500	50
Code	MBPP (Austin et al., 2021)	374	50
	HumanEval (Chen et al., 2021)	120	44
Multi-hop QA	HotpotQA (Yang et al., 2018)	500	50
	2WikiMultiHopQA (Ho et al., 2020)	500	50
	MuSiQue (Trivedi et al., 2022)	496	50
World Knowledge	Natural Questions (Kwiatkowski et al., 2019)	500	50
	TriviaQA (Joshi et al., 2017)	500	50
Popular	MMLU (Hendrycks et al., 2020)	500	50
	GPQA (Rein et al., 2023)	400	44
Out-of-domain Testing	LogicGrid (Mitra & Baral, 2015)	0	50
	MGSM (Shi et al., 2022)	0	50
	CommonsenseQA (Talmor et al., 2019)	0	50
	CommonGen (Lin et al., 2020)	0	50
In-domain subtotal		4,890	538
Out-of-domain subtotal		0	200
Total		**4,890**	**738**

Table 9: **The tasks and corresponding evaluation metrics used in our experiments, organized by domain.**

Domain	Tasks	Metrics
Math	GSM8K (Cobbe et al., 2021)	Accuracy
	MATH (Hendrycks et al., 2021)	Accuracy
Code	MBPP (Austin et al., 2021)	Pass@1
	HumanEval (Chen et al., 2021)	Pass@1
Multi-hop QA	HotpotQA (Yang et al., 2018)	Exact match
	2WikiMultiHopQA (Ho et al., 2020)	Exact match
	MuSiQue (Trivedi et al., 2022)	Exact match
World Knowledge	Natural Questions (Kwiatkowski et al., 2019)	CEM
	TriviaQA (Joshi et al., 2017)	Exact match
Popular	MMLU (Hendrycks et al., 2020)	Accuracy
	GPQA (Rein et al., 2023)	Accuracy
Out-of-domain Testing	LogicGrid (Mitra & Baral, 2015)	CEM
	MGSM (Shi et al., 2022)	CEM
	CommonsenseQA (Talmor et al., 2019)	Accuracy
	CommonGen (Lin et al., 2020)	Coverage

Table 10: **Worker language models and fixed prices in USD per one million tokens.** Underlined models are used only for unseen-worker evaluation; model identifiers and provider details are given in Appendix F.

Scale	Model	Size	Input Price	Output Price
Small	Qwen2.5-7B-Instruct	7B	0.04	0.10
	Llama-3.1-8B	8B	0.02	0.03
	Ministral-8B	8B	0.15	0.15
	Ministral-14B	14B	0.20	0.20
	Qwen3-8B	8B	0.117	0.455
Medium	Qwen3-VL-30B-A3B-Instruct	30B-A3B	0.13	0.52
	Gemma-4-31B	31B	0.12	0.37
	Qwen3-32B	32B	0.08	0.28
Large	Llama-3.3-70B-Instruct	70B	0.13	0.40
	Mistral-Small-4-119B	119B	0.15	0.60
	Qwen3-Next-80B-A3B	80B-A3B	0.10	1.10

E.1 Task Descriptions

Our evaluation combines numerical and programmatic tasks with knowledge questions and evidence-dependent reasoning. The following descriptions distinguish the inputs and reasoning demands of the individual benchmarks; Tables 8 and 9 specify the sample counts and scoring conventions.

GSM8K. GSM8K evaluates grade-school mathematical reasoning through word problems whose solutions involve multiple arithmetic steps (Cobbe et al., 2021). Solving these problems generally requires identifying the relevant quantities, decomposing the task into simpler operations, and carrying intermediate results across steps. The benchmark has consequently been widely adopted in studies of chain-of-thought prompting and verifier-based solution selection. We report accuracy following the setup in Table 9.

MATH. The MATH benchmark contains 12,500 competition-style mathematics problems drawn from areas such as algebra, geometry, and number theory (Hendrycks et al., 2021). Unlike benchmarks centered on short numerical answers, MATH requires models to derive solutions to substantially more involved problems. Each problem is paired with a worked step-by-step solution, which also makes the dataset suitable for training approaches that supervise intermediate reasoning. We report accuracy as in Table 9.

MBPP. MBPP (Mostly Basic Python Problems) tests whether models can synthesize short Python functions from natural-language specifications (Austin et al., 2021). The problems target programming skills accessible to entry-level programmers, including basic algorithmic reasoning, use of the Python standard library, and general language fluency. Correctness is determined automatically using the unit tests provided with each task. We use pass@1 as the principal metric (Table 9).

HumanEval. HumanEval is a code-generation benchmark built around hand-written Python programming tasks (Chen et al., 2021). Each prompt specifies a function through its signature and docstring, while hidden unit tests determine whether the generated implementation satisfies the intended behavior. A task is solved only when the generated code passes all associated tests. HumanEval introduced the widely used pass@k evaluation framework; in our experiments, we report pass@1 in Table 9.

HotpotQA. HotpotQA evaluates question answering that depends on combining information distributed across multiple Wikipedia passages (Yang et al., 2018). Its multi-hop structure includes bridge questions, in which evidence about one entity is needed to reach another, as well as comparison questions involving properties of two entities. This design requires models to connect evidence rather than rely on a single supporting passage. We report exact match following Table 9.

2WikiMultiHopQA. 2WikiMultiHopQA targets multi-hop reasoning over questions derived from Wikipedia and Wikidata (Ho et al., 2020). The benchmark is organized around explicit compositional reasoning paths, so answering a question requires bringing together multiple supporting facts. In addition to answers, the dataset provides annotations for supporting evidence and reasoning paths, enabling direct analysis of intermediate reasoning structure. We report exact match as in Table 9.

MuSiQue. MuSiQue is designed to evaluate multi-hop question answering while limiting the usefulness of shortcut strategies (Trivedi et al., 2022). Its questions are composed so that two to four supporting facts must be connected, with intermediate entities and relations carrying information from one reasoning step to the next. As a result, the benchmark places particular demands on maintaining coherent reasoning across a longer evidence chain. We report exact match following Table 9.

Natural Questions (NQ). Natural Questions (NQ) is based on real, anonymized user queries associated with Wikipedia pages (Kwiatkowski et al., 2019). The dataset includes annotations for both short and long answers, requiring systems to interpret document-level context and identify the appropriate answer span. This setup provides a comparatively realistic test of open-domain reading comprehension. In our experiments, we evaluate case-insensitive exact match (CEM) following Table 9.

TriviaQA. TriviaQA pairs questions authored by trivia enthusiasts with evidence documents that may contain the information needed to answer them (Joshi et al., 2017). Its questions cover broad factual knowledge and are often accompanied by evidence that is noisy or requires information to be integrated across multiple sentences. The benchmark therefore tests both retrieval robustness and reasoning over diverse question types. We report exact match as in Table 9.

MMLU. MMLU (Massive Multitask Language Understanding) evaluates models using multiple-choice questions from 57 subjects across humanities, social sciences, STEM, and professional fields (Hendrycks et al., 2020). Rather than focusing on a single reasoning domain, it measures how broadly a model can apply factual knowledge and problem-solving ability across heterogeneous academic and professional settings. It is commonly used for broad model comparison under zero- or few-shot evaluation. We report accuracy per Table 9.

GPQA. GPQA (Graduate-Level Google-Proof Q&A) is a multiple-choice benchmark composed of expert-authored questions in biology, physics, and chemistry (Rein et al., 2023). The questions are constructed to require specialized scientific knowledge and multi-step reasoning rather than answers obtainable through straightforward web search. Their difficulty is intended to challenge both non-expert humans and strong language models. We report accuracy as summarized in Table 9.

LogicGrid. LogicGrid consists of natural-language logic-grid, or Zebra-style, puzzles that specify relationships among entities, attributes, and constraints (Mitra & Baral, 2015). To solve a problem, a model must convert the textual clues into a consistent set of constraints and use them jointly to infer the requested assignment. The benchmark therefore emphasizes global consistency and structured deductive reasoning rather than local pattern matching. We evaluate CEM as in Table 9.

MGSM. MGSM (Multilingual Grade School Math) extends grade-school mathematical reasoning evaluation to multiple languages through translated GSM8K problems (Shi et al., 2022). Because the underlying arithmetic structure is preserved while the language changes, the benchmark can be used to examine whether multi-step reasoning transfers across different scripts and linguistic systems. It is widely used in evaluations of multilingual chain-of-thought prompting. We report CEM per Table 9.

CommonsenseQA. CommonsenseQA evaluates commonsense reasoning through multiple-choice questions derived from relations in ConceptNet (Talmor et al., 2019). The incorrect options are constructed to remain plausible, reducing the effectiveness of superficial lexical cues. Answering correctly therefore depends on selecting the alternative that is most compatible with background knowledge and ordinary real-world expectations. We report accuracy as listed in Table 9.

CommonGen. CommonGen evaluates generative commonsense reasoning by requiring a model to produce a coherent sentence that incorporates a specified set of concepts (Lin et al., 2020). Since the concepts must be integrated into a meaningful relation rather than merely mentioned independently, the task tests compositional generalization under explicit semantic constraints. The benchmark is therefore commonly used to study controllable generation that depends on commonsense relations among concepts. We report coverage per Table 9.

E.2 Model Descriptions

Model diversity enters our experiments through the worker catalogue, not through updates to worker parameters. The eight training workers serve stateless requests, while optimization changes only the router. For unseen-worker evaluation, we extend the catalogue with Qwen3-8B, Qwen3-32B, and Qwen3-Next-80B-A3B-Instruct without retraining the router. Table 10 records the scale and token prices of each model.

Qwen2.5 (7B). Our pool uses the 7B instruction-following variant of Qwen2.5 as one of its smaller workers (Qwen et al., 2025). The family supports general language tasks as well as math and code, with multilingual and long-context capabilities. The worker remains frozen; the router learns when to select it under the prices in Table 10.

Llama-3.1 (8B). The 8B Llama-3.1 worker brings Meta's model family into the small-model portion of the catalogue (Grattafiori et al., 2024). Its instruction-following, multilingual, and extended-context capabilities are accessed through the same task interface as the other workers. We update neither its parameters nor its interface during router training.

Ministral (8B / 14B). The worker pool includes the 8B and 14B Ministral models through OpenRouter (Liu et al., 2026). Both receive self-contained worker queries under the common execution interface. Their input and output prices are listed separately in Table 10.

Qwen3-VL-30B-A3B-Instruct. This Qwen worker is accessed through OpenRouter under the 30B-A3B configuration (Bai et al., 2025). In our experiments it receives the same text-based task interface as the other workers. Its fixed input and output prices are given in Table 10.

Gemma-4 (31B). Gemma-4-31B is included as a frozen 31B worker through OpenRouter (Team et al., 2026). The router selects it from the shared worker catalogue and supplies a self-contained task query. Table 10 reports the prices used for reward computation and evaluation.

Llama-3.3 (70B). The 70B instruction-tuned Llama-3.3 worker provides a larger model configuration within the training pool (Grattafiori et al., 2024). It is accessed through OpenRouter and remains frozen throughout router training. Its input and output prices are listed in Table 10.

Mistral-Small-4 (119B). Mistral-Small-4-119B is the largest model configuration in the training worker pool. It uses the same stateless execution interface as the other workers and is accessed through OpenRouter. Table 10 gives the fixed prices used in our experiments.

Qwen3 (8B / 32B). The Qwen3-8B and Qwen3-32B workers are excluded from router training and added only for unseen-worker evaluation (Yang et al., 2025). Their two model sizes extend the test-time catalogue while retaining the same query and response interface. Provider identifiers and prices are reported in Appendix F.

Qwen3-Next (80B-A3B). Qwen3-Next-80B-A3B-Instruct is the third worker added for unseen-worker evaluation (Yang et al., 2025). Together with Qwen3-8B and Qwen3-32B, it expands the catalogue from eight to eleven workers. The trained router selects among these workers without further fine-tuning; its provider identifier and fixed prices are listed in Appendix F.

F WORKER MODEL POOL

For every method, worker cost is computed from provider-returned input and output token counts using the same fixed per-token prices reported in Table 11. The reported test-set cost sums these costs over all worker calls, and average worker calls are computed per query.

The closed-vocabulary worker pool comprises eight frozen large language models drawn from two provider gateways and ranging from 7B to 119B parameters. Every worker is exposed through an OpenAI-compatible chat endpoint and treated by the router as a stateless executor, so the policy gradient is confined to the router's own tokens and the same rollout code can target either provider gateway. Table 11 reports the fixed per-token prices used in both training reward and evaluation.

Table 11: Frozen worker pool used for training and in-domain evaluation. Prices are USD per one million input or output tokens.

Worker ID	Provider	Scale	Tier	Input	Output
Llama-3.1-8B (Grattafiori et al., 2024)	NVIDIA	8B	low	0.02	0.03
Ministral-8B (Liu et al., 2026)	OpenRouter	8B	low	0.15	0.15
Qwen2.5-7B-Instruct (Qwen et al., 2025)	OpenRouter	7B	low	0.04	0.10
Ministral-14B (Liu et al., 2026)	OpenRouter	14B	medium	0.20	0.20
Gemma-4-31B (Team et al., 2026)	OpenRouter	31B	medium	0.12	0.37
Qwen3-VL-30B-A3B-Instruct (Bai et al., 2025)	OpenRouter	30B-A3B	medium	0.13	0.52
Mistral-Small-4-119B	OpenRouter	119B	high	0.15	0.60
Llama-3.3-70B-Instruct (Grattafiori et al., 2024)	OpenRouter	70B	high	0.13	0.40

Unseen-worker evaluation. For the unseen-LLM experiment in Figure 4(a), we retain the eight training workers in Table 11 and add Qwen3-8B, Qwen3-32B, and Qwen3-Next-80B-A3B-Instruct. These three workers are excluded from router training. The trained router selects from the expanded eleven-model pool without further fine-tuning. Table 12 lists their provider model identifiers and fixed token prices.

G SKILL BANK

Following SkillRL (Xia et al., 2026), we construct a static SkillBank by distilling reusable skills exclusively from trajectories in our training split. Skills are used only to augment the teacher context during OPSD training. They are never exposed to the student rollout or inference policy, and do not provide the student with any additional input, tool, retrieval source, or inference-time computation. The bank contains no test-set examples, labels, answer keys, or other evaluation-time information. The frozen bank contains 82 domain-specific entries, of which 64 evidence-qualified entries are active for OPSD teacher re-scoring. We separately ablate OPSD to quantify the incremental contribution of skill-guided distillation relative to the RL-only model. Table 13 summarizes its semantic clusters and their functional contracts.

Table 13: Semantic categories in the static skill bank.

Skill category	Functional scope
Quantitative and symbolic reasoning	Preserve state, constraints, and valid solution branches in arithmetic, algebraic, geometric, and discrete reasoning.
Scientific and academic reasoning	Apply domain-specific constraints and evidence to specialized scientific or academic multiple-choice questions.
Program synthesis	Preserve the requested function contract, boundary conditions, and representation invariants during code generation.
Multi-hop evidence chaining	Ground each resolved entity in the next query and execute only the remaining typed relation in an evidence chain.
Entity–attribute retrieval	Bind a requested factual attribute to the correct entity, role, time, or institutional relation before answering.

We match normalized question text against task-specific keyword and structural-pattern rules in the frozen mapping. When multiple skills match, we select only the first in mapping order; teacher guidance and the OPSD loss are omitted if no skill matches or the selected skill is inactive.

Table 12: Additional workers used for unseen-LLM evaluation. All three are accessed through OpenRouter; prices are USD per million tokens.

Provider model ID	Scale	Input	Output
`qwen/qwen3-8b` (Yang et al., 2025)	8B	0.117	0.455
`qwen/qwen3-32b` (Yang et al., 2025)	32B	0.08	0.28
`qwen/qwen3-next-80b-a3b-instruct` (Yang et al., 2025)	80B-A3B	0.10	1.10

Each skill follows the representation $s = (\textit{title}, \textit{principle}, \textit{when to apply})$. Table 14 presents representative entries rather than the complete bank.

H Ablation implementation details

Training and evaluation protocol. Each structural variant is trained with its own planning representation and available actions, and uses the same configuration at evaluation. Training follows the common GRPO loop in Appendix C, with on-policy trajectories collected under the corresponding variant. The worker pool, router backbone, interaction budget, and final-answer interface are shared across variants. Structural variants retain the event-based reward definition: under the aggregation rule in Appendix C, a component is zero when no applicable event occurs. Disabling an operation therefore makes its reward component zero through the absence of corresponding events. The separate reward ablation retains both operations and sets $\rho = 0$.

Without explicit planning: reactive-routing counterpart. This variant removes the initial complete-plan field and all residual-plan state throughout training and evaluation. At each turn, the router selects the next worker and formulates its query directly from the original problem and accumulated interaction history. It therefore learns reactive multi-round routing through step-wise decisions. Since this variant has no REFINE or REVISE events, both plan-adaptation reward components are zero; task correctness, worker cost, and format rewards retain their definitions in Eq. equation 2.

Without plan revision (w/o Adapt): fixed-workflow counterpart. This variant retains the initial global task structure and feedback-conditioned REFINE, which grounds subsequent worker queries in the available observations. It disables REVISE during both training and evaluation, so feedback cannot change the number, order, or dependency structure of the unresolved steps. The resulting policy learns to execute a fixed task structure while incorporating intermediate evidence. Applicable REFINE events retain their reward, whereas the REVISE reward is zero. Task correctness, worker cost, and format rewards are unchanged.

Table 14: Representative skills in the static skill bank used by OPSD.

ID	Skill Title	Principle	When to Apply
`math.function_preimage_transform_pipeline`	Trace function transforms through values and preimages	Trace intermediate values and preimages explicitly, preserve all valid branches, and enforce the stated domain before counting solutions.	Function composition, transformed graphs, preimages, intersections, domains, or asymptotes.
`gpqa.spectroscopy_constraint_stack`	Eliminate structures with a quantitative spectroscopy constraint stack	Check formula, unsaturation, functional groups, spectral integrations, symmetry, coupling, and fragments jointly before retaining a candidate structure.	Questions combining molecular formula, IR, NMR signal counts or splitting, and mass fragments.
`human_eval.numeric_state_and_boundary_semantics`	State the numeric invariant before updating	Update from the prior state rather than partially updated values, and check the smallest legal and first boundary-crossing cases.	Iterative numeric construction, digit shifts, multiplicity tests, capacity filling, or operator-sequence evaluation.
`musique.authority_organization_chain`	Carry each organization node into the next authority query	After resolving an organization, explicitly name it and query only the next unresolved relation while preserving historical and jurisdictional qualifiers.	Multi-hop questions over ownership, membership, operation, leadership, headquarters, or organizational properties.
`natural_qa.office_authority_and_ownership`	Name the institutional edge before the person	Distinguish office holder, appointing authority, owner, founder, and parent organization; return the endpoint of the requested relation.	Questions asking who leads, appoints, chooses, owns, founded, or governs an institution or jurisdiction.

Without the plan-adaptation reward (w/o R_a). This variant retains the full planning representation and both REFINE and REVISE actions, but sets the plan-adaptation coefficient ρ to zero during training. It isolates the auxiliary reward from the availability of the planning operations themselves. The task, cost, and format reward components follow the full model.

I TRAJECTORY REPRESENTATION AND PLAN EXECUTION

The router generates plan content, selects workers, and formulates queries conditioned on the problem and accumulated evidence. XML tags delimit the resulting plan updates, worker requests, and answers for parsing and execution. The schema below specifies these output fields; Table 15 illustrates their use during plan execution.

Output format for plans, worker requests, and answers

```
<plan round="1">
1. ...
2. ...
</plan>

<route>MODEL_ID:self-contained worker query</route>

<control>REFINE</control>
<route>MODEL_ID:query for the next old-plan step</route>

<control>REVISE</control>
<plan round="R">
1. ... replacement residual-plan step ...
</plan>
<route>MODEL_ID:query for the replacement's first step</route>
```

```
<answer>final answer</answer>
```

The initial planning response contains only a catalog-blind ordered plan. A subsequent action either routes the active step or answers. After a worker observation, REFINE preserves the plan version and continues its unfinished portion using the latest evidence, whereas REVISE preserves the completed prefix, reconstructs only the unresolved suffix, and immediately routes the first step of the revised plan. The terminal ANSWER action has no control field. In AdaRouter, these controls implement feedback-driven tasking under a stable high-level intent: REFINE retains a valid residual plan, while REVISE changes only the tasking structure that new evidence has made obsolete.

Table 15: Examples of plan execution and adaptation in the trajectory representation.

Trajectory pattern	**Description**
single-step execution	one-step plan followed by one worker route and an answer
plan continuation	REFINE preserves the plan structure and advances its unfinished portion
plan reconstruction	REVISE emits a reconstructed residual plan and routes its first step
answer completion	the evidence is sufficient and the router emits only `<answer>`

J PROMPTS

This section records the prompt contracts used in the AdaRouter pipeline. The planner, sequential router, and worker use separate system prompts. Runtime state is inserted only between complete assistant generations; it never inserts XML inside an action being generated.

J.1 PROMPT 1: INITIAL PLAN CONSTRUCTION

```
You are the high-level planner.

Given an input problem, construct the shortest ordered solution plan
needed to solve the task. Use at most three worker-requiring steps.

Each step should specify a self-contained unresolved operation rather
than its solution, and should not introduce unknown entities,
intermediate results, or candidate answers.

A self-contained mathematics, multiple-choice, or code-generation task
should be represented as one complete solver operation rather than
decomposed into internal reasoning steps.

For multi-document or relation-chain tasks, decompose the problem
according to unresolved dependencies. Dependent relations whose inputs
are not yet known should remain separate, and the final operation must
obtain the target requested by the original problem.

Output only:

<plan round="1">
1. ...
2. ...
</plan>
```

J.2 PROMPT 2: DYNAMIC PLAN ADAPTATION AND ROUTING

```
You are the sequential router.

Given the current plan state, accumulated worker evidence, and worker
catalogue, either continue the current plan, revise its unresolved
portion, or terminate with a final answer.
```

```
REFINE:
If the remaining plan is still valid after incorporating the latest
worker evidence, preserve the current plan and ground the next
unresolved task in the updated evidence.

REVISE:
If the remaining plan has become invalid, contradicted, redundant,
misordered, incomplete, or otherwise unsuitable for the current
problem-solving state, preserve the completed prefix and reconstruct
only the unresolved suffix. A change in worker selection or query
formulation alone does not constitute plan revision.

ANSWER:
If the accumulated evidence is sufficient to resolve the original
task, terminate and provide the requested answer.

For each routing action, select a worker from the provided catalogue
according to expected task capability, using price only as a tie-breaker
between similarly capable workers. Workers are stateless; therefore,
each routed query must be self-contained and include any grounded
evidence needed for the assigned operation.
```

J.3 PROMPT 3: WORKER

```
Execute only the assigned subtask. The original problem provides
background context and should not be treated as an additional task.

Return the requested result directly without chain-of-thought or
structured wrappers. Preserve the answer type and output format
required by the assigned task.
```

J.4 PROMPT 4: PLAN-ADAPTATION EVALUATOR

```
You are a strict relevance evaluator for training a model-routing policy. Judge only the
supplied planning transitions. Do not solve the original question, do not judge the
final answer, and do not infer any hidden reference answer. Return one JSON object only.

REFINE event:
- dependency_relation="dependent" only when the next step logically needs a concrete
entity, value, constraint, or conclusion supplied by the immediately preceding worker
observation. Independent or parallel steps must be "independent", even when the next
query is otherwise good.
- grounding_quality="pass" only when dependency_relation="dependent" and the next worker
query accurately incorporates that evidence, is self-contained for a worker that sees no
history, and introduces no unsupported fact.
- When dependency_relation="independent", grounding_quality must be "not_applicable";
the event is masked rather than rewarded.
- Do not fail a dependent query merely for including additional relevant facts that are
supported by the input.

REVISE event:
- revision_timing="justified" only when the triggering observation makes the old
unfinished plan unsuitable (for example not found, contradiction, invalid premise,
missing operation, or a materially needed recovery) and the new residual plan materially
addresses that issue. Immediate revision is permitted; do not require repeated
failed attempts.
- first_route_quality="useful" only when the first new route matches the first step of
the revised plan, is grounded in the available evidence, is self-contained, and its
returned observation is relevant and useful for that step. Mere fluent text or absence
of a 'not found' phrase is insufficient.

Required JSON schema: {"events":[{"event_id":"exact input id","kind":"REFINE",
"dependency_relation":"dependent|independent","grounding_quality":"pass|fail|not_applicable",
"reason":"brief evidence-based reason"},{"event_id":"exact input id","kind":"REVISE",
"revision_timing":"justified|unjustified","first_route_quality":"useful|not_useful",
"reason":"brief evidence-based reason"}]} Preserve every event_id and kind exactly, emit
exactly one judgment per input event, and keep each reason under 40 words.
```

K AdaRouter Workflow

Appendix I specifies how plans, worker requests, and answers are represented during execution. Given an input question Q, the router first assesses its task structure. For a single-hop task, it constructs a one-step plan and routes a self-contained solver or retrieval query before generating the answer. For a multi-hop task, it decomposes the question into an ordered sequence of sub-questions and solves them sequentially with dynamic plan adaptation. During execution, REFINE retains a valid unfinished plan and grounds the next task in newly obtained evidence, whereas REVISE reconstructs only the unresolved suffix when that evidence changes the remaining information needs.

L Illustrative Cases for Plan Suitability

The four cases referenced in Table 1 share the same multi-hop problem, initial plan, and execution state. They differ only in the worker observation returned for the current task.

L.1 Shared problem and execution state

Problem. Consider the following multi-hop question:

> Which company owns the record label that released the self-titled debut studio album of the London-born artist who won Best New Artist at the 2019 Grammy Awards?

Suppose the router constructs the following initial plan:

> `<plan round="1">`
> 1. Identify a London-born artist matching the description. 2. Determine the record label that released the artist's self-titled debut studio album. 3. Verify that the artist won Best New Artist at the 2019 Grammy Awards. 4. Identify the company that owns the record label. `</plan>`

After Step 1, the first worker proposes Dua Lipa as the candidate artist. The current task is therefore:

> Determine the record label that released Dua Lipa's self-titled debut studio album.

Cases A–D start from this same state.

L.2 Case A: Successful task with a supported remaining plan

Table 16: Case A: successful completion of the current task while the remaining plan remains suitable.

Schema element	Content
Current task	Determine the record label that released Dua Lipa's self-titled debut studio album.
Worker observation	Dua Lipa's self-titled debut studio album was released by Warner Bros. Records.
Current task completed?	Yes.
Observation type	Useful evidence obtained.
Effect on remaining plan	The returned label can be used by the later ownership lookup. No planned dependency is contradicted.
Plan decision	Preserve the remaining plan and continue to the next unresolved task.
Control	REFINE.

The observation resolves the current task without changing the suitability of the remaining plan.

L.3 CASE B: SUCCESSFUL TASK THAT INVALIDATES A DOWNSTREAM PREMISE

Table 17: Case B: successful completion of the current task while a downstream premise of the remaining plan is invalidated.

Schema element	Content
Current task	Determine the record label that released Dua Lipa's self-titled debut studio album.
Worker observation	The self-titled debut album was independently released and has no associated record label.
Current task completed?	Yes.
Observation type	A downstream premise is contradicted.
Effect on remaining plan	The planned ownership lookup assumes the existence of a record label, which the observation now denies.
Plan decision	Revise the unresolved path rather than execute the ownership lookup under an unsupported premise.
Control	REVISE.

The current task is resolved, but its result invalidates a premise required by a later task.

L.4 CASE C: FAILED TASK WITH UNRESOLVED PLAN SUITABILITY

Table 18: Case C: failure to complete the current task without evidence that the remaining plan is unsuitable.

Schema element	Content
Current task	Determine the record label that released Dua Lipa's self-titled debut studio album.
Worker observation	I cannot determine the record label from the available evidence.
Current task completed?	No.
Observation type	No usable result.
Effect on remaining plan	The observation provides no evidence that the candidate artist, album relation, or label relation is incorrect.
Plan decision	Retain the current task structure and retry the unresolved operation with another worker or a revised query.
Control	No structural revision.

The call fails to resolve the task, but it provides no evidence that the plan structure itself is unsuitable.

L.5 CASE D: FAILED TASK THAT EXPOSES AN UNSUITABLE TASK ORDER

Table 19: Case D: failure to complete the current task because the unresolved tasks are in an unsuitable order.

Schema element	Content
Current task	Determine the record label that released Dua Lipa's self-titled debut studio album.
Worker observation	The artist identity has not yet been verified. The Best New Artist condition must first be resolved before the album and record label can be identified reliably.
Current task completed?	No.
Observation type	A dependency on a later task is exposed.
Effect on remaining plan	Step 2 depends on the identity verification currently scheduled as Step 3. The unresolved tasks are therefore in an unsuitable order.
Plan decision	Move the verification step before the label lookup, then continue using the verified artist identity.
Control	REVISE.

The failed call identifies a dependency problem in the unresolved plan.

M CASE STUDY

We present two representative successful trajectories to illustrate the decision-making logic of AdaRouter. Observation bodies are abridged to their conclusions.

CASE 1: EVIDENCE-GROUNDED PLAN CONTINUATION

Source: MuSiQue. **Pattern:** two REFINE decisions. **Task outcome:** correct.

Query. Identify the owner of the record label associated with the performer of "Please Remember Me."

Schema element	Content
`<plan round=1>`	identify the performer; determine the performer's record label; find the owner of that label
first `<route>`	ask who performed "Please Remember Me"
worker observation	Rodney Crowell
REFINE control	route the next step using Rodney Crowell as a grounded premise
worker observation	Warner Bros. Records
REFINE control	route the next step using Warner Bros. Records as a grounded premise
worker observation	Warner Music Group
`<answer>`	Warner Music Group

What it illustrates. The initial relation chain remains valid after each observation. Rather than reconstructing the plan, the router continues its execution by preserving the solution plan and converting each newly resolved entity into a self-contained query for the next relation. The trajectory therefore separates plan persistence from query grounding: REFINE retains the plan structure, while the routed query changes with the evidence.

CASE 2: RESIDUAL-PLAN REVISION AFTER MISSING CONTEXT

Source: MMLU. **Pattern:** REVISE followed by REFINE. **Task outcome:** correct.

Schema element	Content
`<plan round=1>`	evaluate the candidate expressions from the ticket-sale quantities
first `<route>`	ask a worker to determine the correct expression
worker observation	the supplied context is insufficient to interpret all quantities
REVISE control	replace the unresolved suffix with two operations: compute the known revenue, then resolve how the remaining quantities enter the expression
worker observation	candidate (A)
REFINE control	execute the second replacement step using the computed revenue as a grounded premise
worker observation	candidate (D)
`<answer>`	(D)

What it illustrates. The first worker exposes that the original one-step plan omits information needed to interpret the full expression. The router therefore performs REVISE, opens a new plan round, preserves the completed prefix, and reconstructs only the unresolved computation. Once the revised plan is adequate, REFINE advances within that plan rather than triggering another reconstruction.

N LIMITATIONS

Third-party inference remains non-stationary: quotas, outages, model updates, and drifting latency heterogeneously affect workers and cannot be controlled at training time. We therefore base the optimization signal on task-verifier outcomes, structured plan-adaptation rewards, and frozen token prices rather than wall-clock latency. A catalog-blind initial plan reduces dependence on a particular worker pool, while later routing decisions can still exploit the current catalogue.