ROUTEWEAVER: WEAVING MODE SELECTION AND EXECUTION INTO UNIFIED LLM ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM routing dynamically selects and coordinates heterogeneous language models across diverse tasks. However, existing approaches are largely built around a predefined execution paradigm, such as single-round, multi-round, or agentic routing, which fixes how model calls are organized and limits adaptation to different query needs. We propose `RouteWeaver`, a unified routing framework that integrates these routing modes within a single LLM policy, jointly learning which mode to use and how to select models within that mode. To jointly optimize mode selection and within-mode execution, we develop COMET-GRPO (**CO**ordinated **M**ode–**E**xecution **T**raining), which decouples the trajectory-level advantage into separate signals for mode selection and within-mode execution, enabling more targeted credit assignment to each decision level. We further introduce progressive exploration, which gradually shifts training from externally assigned modes to autonomous selection, so that the router first learns how to execute each mode before deciding when to use it. We evaluate `RouteWeaver` across mathematics, coding, knowledge, logical reasoning, and factual recall. `RouteWeaver` achieves an average accuracy of 61.4%, an 11.03% relative improvement over the strongest baseline, and outperforms all compared baselines in each of the five domains. Ablations show that both COMET-GRPO and progressive exploration are important for adaptive routing; without progressive exploration, the routing policy collapses entirely to single-round execution.

1 INTRODUCTION

The growing diversity of large language models (LLMs) has motivated LLM routing as a paradigm for leveraging complementary capabilities, where different models are selected and coordinated according to their strengths, costs, and suitability for each query (Chen et al., 2024a; Ding et al., 2024; Hu et al., 2024; Ong et al., 2025). Depending on how model calls are organized, current routing approaches generally follow three paradigms: single-round routing selects a single model for one-shot execution (Chen et al., 2024b; Feng et al., 2025b; Wang et al., 2026), multi-round routing makes successive model selections conditioned on intermediate outputs (Zhang et al., 2025; Shao et al., 2026), and agentic routing organizes multiple models into structured collaborative workflows (Yue et al., 2025; Feng et al., 2026b). Although routing decisions within each paradigm can adapt to the query, the execution paradigm itself is typically fixed in advance. As a result, the router can adapt *which models* to use, but not *how model calls should be organized* for different queries. This raises a natural question: *how can we unify these routing paradigms within a single policy that jointly adapts execution mode and model selection to each query?*

Unifying these routing paradigms within a single policy is non-trivial. Single-round, multi-round, and agentic routing follow different execution structures and decision processes, requiring a common formulation that accommodates all three while preserving their distinct behaviors. Beyond this structural challenge, two learning problems are particularly important. *First, different queries may favor different execution modes, yet determining which mode is appropriate requires reasoning about the task before its execution unfolds.* A unified router must therefore learn when single-round, multi-round, or agentic execution is most suitable for each query. *Second, selecting an appropriate mode alone is insufficient, since each mode demands a different form of model selection and coordination.* Single-round routing requires choosing an effective model directly, multi-round routing must adapt subsequent selections to intermediate outputs, and agentic routing additionally organizes

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.

models through roles and dependencies. The router must therefore learn not only *which mode to use*, but also *how to execute effectively within the selected mode*.

To address these challenges, we propose `RouteWeaver`, a unified routing policy that learns to select among single-round, multi-round, and agentic execution paradigms and perform model selection within the chosen paradigm. We first introduce a unified routing format that expresses all three paradigms through a common action grammar for mode selection and subsequent routing decisions. To jointly optimize these two levels of decisions, we develop COMET-GRPO (**CO**ordinated **M**ode–**E**xecution **T**raining), which decouples the trajectory-level advantage into separate signals for mode selection and within-mode execution, enabling targeted credit assignment to each decision level. We further introduce progressive exploration, which gradually shifts training from externally assigned modes to autonomous selection, allowing the router to first learn how to execute within each mode before deciding when to use it. We evaluate `RouteWeaver` on benchmarks spanning mathematics, coding, knowledge, logical reasoning, and factual recall. `RouteWeaver` achieves 61.4% average accuracy, an 11.03% relative improvement over the strongest baseline, and leads all baselines across the five domains. We further study performance–cost trade-offs and generalization on held-out benchmarks (AIME, GPQA-D, and BBH). With $\alpha = 0.1$, Worker inference cost decreases by 23% with only a 1.0-point accuracy drop, while `RouteWeaver` achieves 53.7% average accuracy on held-out benchmarks, an 18.28% relative improvement over the strongest baseline. Ablations and routing analyses further validate COMET-GRPO and progressive exploration and reveal adaptive execution choices across tasks. Our main contributions are summarized as follows:

- We formulate unified LLM routing across single-round, multi-round, and agentic execution paradigms, enabling a single policy to adapt both the execution mode and model selection to each query rather than committing to a predefined routing schema.
- We propose `RouteWeaver`, featuring COMET-GRPO, which decouples credit assignment for mode selection and within-mode execution, together with progressive exploration that gradually transitions the router from learning how to execute each mode to autonomously deciding when to use it for different queries.
- Extensive experiments across mathematics, coding, knowledge, logical reasoning, and factual recall demonstrate the effectiveness of unified routing over existing routers and fixed-mode strategies. Further analyses examine performance–cost trade-offs, routing behavior, and the contributions of the proposed training components.

2 Related Work

2.1 Execution Paradigms for LLM Routing

Existing LLM routing methods can be categorized by how model calls are organized. (1) **Single-round routing** selects one model for each query, typically based on model capability, query difficulty, or performance–cost trade-offs. RouteLLM (Ong et al., 2025) learns routing from preference data, while RouterDC (Chen et al., 2024b) and GraphRouter (Feng et al., 2025b) model query–model relations. ICL-Router (Wang et al., 2026) further improves routing through learned model representations. (2) **Multi-round routing** allows routing decisions to evolve across calls using intermediate outputs. Router-R1 (Zhang et al., 2025) formulates routing as a sequential decision process with repeated model selection and response aggregation, while R2-Reasoner (Shao et al., 2026) combines task decomposition with model allocation for intermediate reasoning. (3) **Agentic routing** organizes multiple models into collaborative workflows. MasRouter (Yue et al., 2025) jointly considers collaboration structures, agent roles, and model choices, while GraphPlanner (Feng et al., 2026b) generates query-dependent workflows with model–role assignments and explicit interactions. Despite adaptive routing within each paradigm, prior approaches fix the execution paradigm in advance. In contrast, `RouteWeaver` unifies single-round, multi-round, and agentic routing as selectable modes of one policy, jointly learning mode selection and within-mode execution.

2.2 Reinforcement Learning for Agentic LLMs

Reinforcement learning (RL) has become a central approach for improving the reasoning and interactive capabilities of LLMs. GRPO (Shao et al., 2024) replaces critic-based advantage estima-

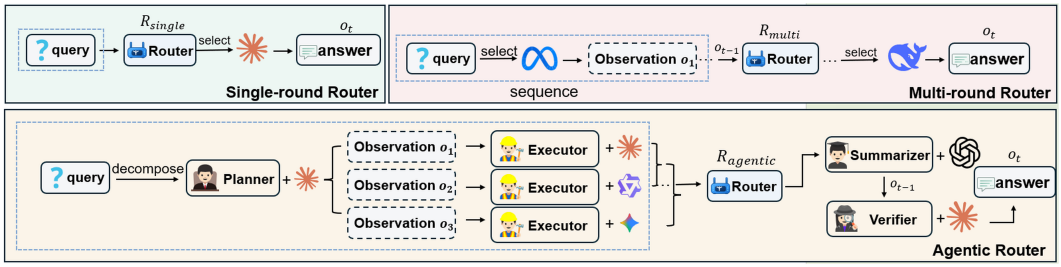


Figure 1: Three execution paradigms. **Top left, single-round:** the router selects one Worker for a single call. **Top right, multi-round:** the router selects Workers sequentially using accumulated observations. **Bottom, agentic:** the router selects Workers, assigns roles, and specifies dependencies between calls; the depicted workflow is illustrative, and its roles and dependencies vary across queries.

tion with group-relative rewards and has inspired several variants. Dr.GRPO (Liu et al., 2025) addresses optimization biases in GRPO, while DAPO (Yu et al., 2025) improves training stability through asymmetric clipping and dynamic sampling. GDPO (Liu et al., 2026) further decouples reward normalization for multi-reward optimization. RL has also been extended to interactive agents. Search-R1 (Jin et al., 2025) and ReTool (Feng et al., 2026a) train LLMs to interleave reasoning with search or tool interactions, while GiGPO (Feng et al., 2025a) introduces episode- and step-level relative advantages for finer credit assignment in multi-turn agents. Building on these advances, COMET-GRPO adapts group-relative optimization to unified LLM routing, where mode selection and within-mode execution form two coupled decision levels and receive separate credit signals.

3 PRELIMINARIES

Given a query x and a pool of Worker models $\mathcal{W} = \{w_1, \ldots, w_N\}$, a router $\mathcal{R}$ determines which Workers to invoke and how their calls are organized. We refer to the resulting pattern of Worker invocation and information flow as the *execution structure*. We distinguish three execution paradigms according to how routing decisions are structured (Figure 1).

Single-round routing. The router selects one Worker for the query and calls it once:

$$w = \mathcal{R}_{\text{single}}(x), \qquad o = w(x). \tag{1}$$

Here, $w \in \mathcal{W}$ is the selected Worker and o is its output. The resulting execution structure consists of a single Worker call: the routing decision precedes execution, with no subsequent Worker selection based on the returned response.

Multi-round routing. The router makes successive calls using earlier feedback. Let s_t denote the sub-query generated for the t-th Worker call and $h_{t-1} = \big((w_1, s_1, o_1), \ldots, (w_{t-1}, s_{t-1}, o_{t-1})\big)$ the ordered history of preceding interactions. Each routing decision specifies a Worker and its sub-query:

$$(w_t, s_t) = \mathcal{R}_{\text{multi}}(x, h_{t-1}), \qquad o_t = w_t(s_t). \tag{2}$$

The response is appended to the history before the next decision. This sequential execution structure allows intermediate outputs to guide model selection, task refinement, and further reasoning.

Agentic routing. Calls are organized into a workflow with explicit roles and dependencies. We describe the completed workflow after t calls by $G_t = (V_t, E_t)$, where V_t indexes completed Worker calls and an edge $(j, k) \in E_t$ indicates that call k uses the output of call j. A routing decision is

$$(w_t, s_t, r_t, P_t) = \mathcal{R}_{\text{agentic}}(x, G_{t-1}), \tag{3}$$

where r_t specifies the role, such as planning, execution, verification, or summarization, and $P_t \subseteq V_{t-1}$ is the set of completed predecessor calls selected as dependencies. The outputs $\{o_j : j \in P_t\}$ are supplied as context to Worker w_t, which executes sub-query s_t under role r_t, and its output is recorded in the workflow. By selecting P_t, the router determines the direct dependencies of each call. The resulting dependency structure supports branching and aggregation, while independent calls may execute in parallel. Agentic workflows can also be sequential; their defining distinction here is the explicit organization of roles and dependencies.

4 ROUTEWEAVER

Building on the above formulation, RouteWeaver unifies single-round, multi-round, and agentic routing in one policy. This raises three challenges. First, their heterogeneous execution structures require a common representation. Second, performance depends jointly on mode selection and within-mode execution, calling for differentiated learning signals. Third, mode selection requires sufficient experience across modes to avoid early bias. RouteWeaver addresses these challenges through a unified routing formulation (Section 4.1), COMET-GRPO for coordinated mode–execution learning (Section 4.2), and progressive exploration toward autonomous routing (Section 4.3).

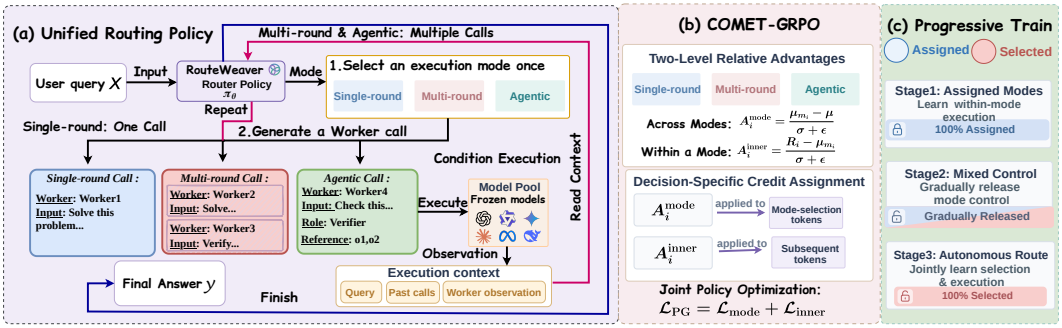


Figure 2: Overview of RouteWeaver. **(a)** One routing policy π_θ selects an execution mode, then calls frozen Workers;Multi-round calls automatically use the latest observation, while agentic calls include role assignments and router-selected references to earlier outputs.Returned observations are appended to its context until it emits the final answer. **(b)** COMET-GRPO computes mode-level and within-mode advantages per reward component, mixes them (Eq. (7)), and assigns A_i^{mode} to mode-selection tokens and A_i^{inner} to subsequent router tokens. **(c)** Progressive training moves mode selection from assigned modes to the router in three stages.

4.1 UNIFIED TRAJECTORY REPRESENTATION AND EXECUTION

To this end, we represent each routing process as a mode-conditioned trajectory. Given a query x, the router first selects an execution mode

$$m \sim \pi_\theta(\cdot \mid x), \qquad m \in \mathcal{M} = \{\text{single}, \text{multi}, \text{agentic}\},$$

and subsequently generates routing actions conditioned on the query and the accumulated interaction context. Each Worker call specifies a Worker w and a sub-query s; agentic calls additionally specify a role r and references P to completed calls. Multi-round execution passes the previous observation to each subsequent call, while agentic execution passes the referenced outputs to role-structured and potentially parallel calls. Single-round execution contains exactly one Worker call. The serialization format and mode-specific constraints are detailed in Appendix A.

Executing an action returns the Worker outputs as an observation o_t, which is appended to the router context before the next decision. The router continues issuing mode-compatible actions until it produces the final answer y, yielding

$$\tau = (m, a_1, o_1, \ldots, a_T, o_T, y). \tag{4}$$

The mode, routing actions, and final answer are generated by the router and optimized as policy tokens, whereas Worker observations and other environment-inserted content are used only as context.

Each trajectory receives a task reward and an efficiency reward,

$$(r_{\text{task}}, r_{\text{eff}}) = \begin{cases} (1,\ S_{\text{cost}}(\tau)), & \tau \text{ is valid and correct,} \\ (0,\ 0), & \tau \text{ is valid but incorrect,} \\ (-1,\ -1), & \tau \text{ is terminal-invalid,} \end{cases} \qquad S_{\text{cost}}(\tau) = \frac{1}{1 + C(\tau)/C_{\text{ref}}}. \tag{5}$$

Correctness is determined by the task-specific scorer (Appendix C), while $C(\tau)$ measures the fixed-reference inference cost of Worker calls, excluding router computation, and $C_{\text{ref}} > 0$ is a frozen reference cost (Appendix E). Terminal-invalid trajectories arise from grammar or execution-constraint violations. These trajectory-level signals jointly reflect mode selection and subsequent execution, motivating the decision-specific credit assignment introduced next.

4.2 COMET-GRPO

Joint mode selection and execution create an ambiguous credit-assignment problem. A low-reward trajectory may result from selecting an unsuitable execution mode, from executing an otherwise suitable mode poorly, or from both. Standard GRPO (Shao et al., 2024) does not distinguish these cases: it assigns the same trajectory-level relative advantage to the mode-selection tokens and all subsequent router-generated tokens. We therefore decompose relative feedback into two complementary comparisons: an across-mode comparison for mode selection and a within-mode comparison for execution conditioned on the selected mode. COMET-GRPO (*COordinated Mode–Execution Training*) uses these two signals to optimize the two decision levels separately within a shared policy.

Two levels of relative advantage. For each query, we sample a group of K rollouts; all statistics below are computed within this group and separately for each reward component. Let $\mathcal{V}$ be the rollouts retained for advantage estimation (Appendix E). For one component with values r_i, let μ and σ be the mean and standard deviation of $\{r_i\}_{i\in\mathcal{V}}$, so that GRPO assigns $A_i^{\mathrm{GRPO}} = (r_i - \mu)/(\sigma + \epsilon)$ to every router token of τ_i. In a *free* group, each rollout selects its own mode, so the group need not cover all modes. Let $\mathcal{V}_m \subseteq \mathcal{V}$ contain the retained rollouts that selected m, with mean μ_m of the same component. For rollout i with mode m_i,

$$A_i^{\mathrm{mode}} = \frac{\mu_{m_i} - \mu}{\sigma + \epsilon}, \qquad A_i^{\mathrm{inner}} = \frac{r_i - \mu_{m_i}}{\sigma + \epsilon}. \tag{6}$$

Intuitively, A_i^{mode} measures how the average reward of the selected mode compares with the overall rollout-group average for the same query, whereas A_i^{inner} measures how this particular trajectory compares with the average trajectory under the same mode. Both use the group standard deviation σ rather than a within-mode one, keeping the two signals on a common scale.

Following the reward-wise normalization of GDPO (Liu et al., 2026), we compute these advantages separately for the task and efficiency rewards and then mix them:

$$A_i^l = (1-\alpha)\, A_{i,\mathrm{task}}^l + \alpha\, A_{i,\mathrm{eff}}^l, \qquad l \in \{\mathrm{mode}, \mathrm{inner}\}. \tag{7}$$

With $\alpha = 0$ only the task signal is used; $\alpha > 0$ adds the efficiency signal. The weight α is fixed within a run and is independent of the exploration schedule of Section 4.3, and no further batch-level normalization is applied. Component-wise definitions are given in Appendix E.

Decision-specific credit assignment. Having separated the two sources of feedback, we apply each signal only to the decisions it is intended to supervise. The mixed A_i^{mode} is assigned to the tokens of the mode decision, S_i^{mode}, and A_i^{inner} to the subsequent router tokens, S_i^{inner}: Worker choices, roles, references, sub-queries, and the final answer. Observations, injected content, and padding are masked. Following the PPO clipped surrogate objective (Schulman et al., 2017), for a router token z_{ik} with context H_{ik} (indexed by k to distinguish tokens from interaction steps), the clipped token loss is

$$\ell_{ik}(A) = -\min\big[r_{ik}A,\ \mathrm{clip}(r_{ik}, 1-\eta_-, 1+\eta_+)A\big], \qquad r_{ik} = \frac{\pi_\theta(z_{ik} \mid H_{ik})}{\pi_{\mathrm{old}}(z_{ik} \mid H_{ik})}, \tag{8}$$

with dual clipping for negative advantages as specified in Appendix E.

Joint policy optimization. Each of the two losses is normalized by its own global count of effective tokens in the mini-batch, and the results are added:

$$\mathcal{L}_{\mathrm{PG}} = \frac{\sum_i \sum_{k\in S_i^{\mathrm{mode}}} \ell_{ik}(A_i^{\mathrm{mode}})}{N_{\mathrm{mode}}} + \frac{\sum_i \sum_{k\in S_i^{\mathrm{inner}}} \ell_{ik}(A_i^{\mathrm{inner}})}{N_{\mathrm{inner}}}, \tag{9}$$

where N_{mode} and N_{inner} count the effective mode and inner tokens in the mini-batch, and an empty term is skipped. Mode tokens are typically far fewer than execution tokens, so averaging over all tokens together would give the mode term a small aggregate weight merely because it is short; separate normalization removes this dependence on length. The policy is regularized toward the frozen initial router π_{ref}, distinct from π_{old}, using a KL penalty with coefficient β:

$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{N_{\mathrm{KL}}} \sum_{(i,k)\in\Omega_{\mathrm{KL}}} \kappa_{ik}, \tag{10}$$

where Ω_{KL} is the full effective policy mask, N_{KL} its token count, and κ_{ik} a per-token estimate of the KL divergence to π_{ref}. The estimator, masks, and degenerate groups are described in Appendix E.

4.3 PROGRESSIVE EXPLORATION

If the router chose modes autonomously from the start, early preferences could reduce the training opportunities of the other modes. Progressive exploration therefore first provides execution experience within each mode and then gradually hands mode selection to the router, in three phases indexed by the data-batch index b. Each query is *forced* with probability

$$\rho_b = \begin{cases} 1, & b < b_0, \\ 1 - \dfrac{b - b_0}{b_1 - b_0}, & b_0 \le b < b_1, \\ 0, & b \ge b_1, \end{cases} \tag{11}$$

and otherwise *free*; all K rollouts of a query share this condition. A forced query receives one externally assigned mode, with modes distributed across the batch, and its injected mode prefix is masked.

Cold start ($b < b_0$). This phase teaches execution within each mode before the router selects modes. All queries are forced. We denote by $\mathcal{L}_{\text{GRPO}}^{\text{global}}$ the ordinary GRPO loss, obtained by averaging $\ell_{ik}(A_i^{\text{GRPO}})$ over all effective router tokens (Eq. 17). The router is updated with $\mathcal{L}_{\text{GRPO}}^{\text{global}}$ on the task reward alone; advantage decomposition and the support term are inactive. Runs with $\alpha > 0$ resume from the checkpoint at the end of this phase (Appendix F).

Forced/free transition ($b_0 \le b < b_1$). Forced groups decrease and free groups increase. Free groups use the advantages of Eq. 7. Forced groups compute A_i^{GRPO} for each enabled reward component, mix them with the same weights, and apply the result to their execution tokens, which enter the inner term of Eq. 9; their injected mode tokens remain masked (Eq. 18). To keep probability support on every mode and prevent it from concentrating prematurely, we add a one-sided penalty on $\bar{p}_m$, the normalized score of candidate mode m averaged over the unique queries in the batch (Appendix E):

$$\mathcal{L}_{\text{support}} = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \max(0, \log \delta - \log(\bar{p}_m + \epsilon)) \,. \tag{12}$$

Because it acts on batch-level averages, individual queries can still favor different modes. Its coefficient $\lambda_b = \lambda_0 \rho_b$ applies only during the transition; the term is inactive in the other phases.

Fully autonomous routing ($b \ge b_1$). All queries are free, and the router jointly learns mode selection and execution with COMET-GRPO.

Phase objectives. Across the three phases, the router minimizes

$$\mathcal{L}_b = \begin{cases} \mathcal{L}_{\text{GRPO}}^{\text{global}} + \beta\, \mathcal{L}_{\text{KL}}, & b < b_0, \\ \mathcal{L}_{\text{PG}} + \beta\, \mathcal{L}_{\text{KL}} + \lambda_b \mathcal{L}_{\text{support}}, & b_0 \le b < b_1, \\ \mathcal{L}_{\text{PG}} + \beta\, \mathcal{L}_{\text{KL}}, & b \ge b_1, \end{cases} \tag{13}$$

where $\mathcal{L}_{\text{GRPO}}^{\text{global}}$ uses the task-only GRPO advantage and $\mathcal{L}_{\text{PG}}$ uses the α-mixed advantages, including the mixed GRPO advantages of forced groups in its inner term. Phase boundaries and coefficients are reported in Appendix F.

5 EXPERIMENTS

We evaluate `RouteWeaver` across five task domains to examine whether jointly selecting and executing different routing modes improves over methods restricted to a fixed execution paradigm. We further study its performance–cost trade-off, isolate the contributions of COMET-GRPO and progressive exploration, and analyze how execution-mode choices vary across tasks. Additional dataset, evaluation, and implementation details are provided in the appendices.

5.1 EXPERIMENTAL SETUP

Datasets. Our evaluation comprises nine benchmarks spanning five domains: **Math**, MATH (Hendrycks et al., 2021b) and Omni-MATH (Gao et al., 2025); **Code**, APPS (Hendrycks

Table 1: Accuracy (%) on the evaluation benchmarks; `RouteWeaver` is trained with $\alpha = 0$. Domain accuracy is the unweighted mean of its constituent dataset accuracies; Avg is the unweighted mean of the five domain accuracies. Best results are in **bold**; numbers in parentheses give the relative improvement of `RouteWeaver` over the strongest baseline in each column.

Method	Math	Code	Knowledge	Reason	Recall	Avg
Single-round Router						
Router-KNN	61.7	34.8	38.3	68.3	40.4	48.7
Router-MLP	58.7	33.2	40.0	59.2	34.2	45.1
RouterDC	61.3	29.4	34.2	57.5	35.0	43.5
GraphRouter	64.6	36.1	34.2	69.2	39.6	48.7
Multi-round Router						
Prompt LLM	45.8	33.7	29.2	54.2	30.0	38.6
Router-KNN-MR	40.8	33.2	30.8	48.3	29.6	36.6
R2-Reasoner	45.8	30.7	28.3	50.0	23.3	35.6
Router-R1	66.3	52.8	48.9	68.7	39.8	55.3
Agentic Router						
MasRouter	54.2	37.6	41.7	62.5	38.3	46.8
GraphPlanner	47.5	48.3	30.0	54.2	39.2	43.8
`RouteWeaver`	**74.6** (+12.52%)	**58.2** (+10.23%)	**55.0** (+12.47%)	**76.7** (+10.84%)	**42.5** (+5.20%)	**61.4** (+11.03%)

et al., 2021a), LiveCodeBench (Jain et al., 2025), and TACO (Li et al., 2023); **Knowledge**, SuperGPQA (Du et al., 2025); **Reason**, SATBench (Wei et al., 2025); and **Recall**, TriviaQA (Joshi et al., 2017) and SimpleQA-Verified (Haas et al., 2025). Test questions do not overlap with the training questions used for router optimization. Dataset composition, training pools, and evaluation sets are provided in Appendix B.

Router and Worker models. The router is Qwen3-4B-Instruct-2507 (Yang et al., 2025) with all parameters fine-tuned. The Worker pool contains six frozen models: Qwen3-8B and Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025), Llama-3.1-8B-Instruct and Llama-3.3-70B-Instruct (Llama Team, AI @ Meta, 2024), Gemma-3-27B-IT (Gemma Team, 2025), and Gemini-2.5-Flash-Lite (Gemini Team, Google, 2025). The router refers to Workers only by anonymous IDs with capability descriptions and never sees model names or prices. Model identifiers and serving providers are listed in Appendix D.

Baselines. We compare `RouteWeaver` with ten routers spanning all three execution paradigms. Single-round baselines include Router-KNN and Router-MLP (Hu et al., 2024), RouterDC (Chen et al., 2024b), and GraphRouter (Feng et al., 2025b); multi-round baselines include Prompt LLM and Router-KNN-MR following Zhang et al. (2025), R2-Reasoner (Shao et al., 2026), and Router-R1 (Zhang et al., 2025); and agentic baselines include MasRouter (Yue et al., 2025) and GraphPlanner (Feng et al., 2026b).

5.2 Performance Across Execution Paradigms

`RouteWeaver` substantially outperforms fixed-paradigm routers. Table 1 compares `RouteWeaver` with single-round, multi-round, and agentic routing baselines across the five evaluation domains. `RouteWeaver` reaches an average accuracy of 61.4%, an 11.03% relative improvement over the strongest baseline (55.3%, Router-R1). Its relative improvements over the strongest single-round, multi-round, and agentic baselines are 26.08%, 11.03%, and 31.20%, respectively, and it leads all baselines in each of the five domains.

Adaptive execution provides broad but task-dependent gains. `RouteWeaver` leads the strongest baseline in every evaluation domain, with relative improvements ranging from 5.20% on Recall to 12.52% on Math. Importantly, the strongest fixed-paradigm baseline varies across domains: multi-round routers perform best among the baselines on Math, Code, and Knowledge, whereas single-round routers are strongest on Reason and Recall. This variation suggests that no single execution paradigm is uniformly preferable across tasks, motivating `RouteWeaver`'s query-

Table 2: Ablation of the two learning components, all trained with $\alpha = 0$ under the same budget. Accuracy (%) is reported by domain. Mode usage is reported as the aggregate proportion of evaluation queries assigned to single-round (S), multi-round (M), and agentic (A) execution.

	Accuracy (%)						Mode usage
Variant	**Math**	**Code**	**Knowledge**	**Reason**	**Recall**	**Avg**	**S / M / A**
`RouteWeaver`	74.6	58.2	55.0	76.7	42.5	61.4	.35 / .41 / .24
w/o COMET-GRPO	72.3	57.2	49.2	74.1	41.2	58.8	.13 / .82 / .05
w/o progressive exploration	70.2	48.6	50.3	68.3	40.4	55.6	1.00 / .00 / .00

adaptive mode selection. Section 5.5 further examines how `RouteWeaver` distributes queries across execution modes.

5.3 Performance–Cost Trade-off

Efficiency preferences allow `RouteWeaver` to operate at different accuracy–cost points. The main results use $\alpha = 0$ and optimize task performance only, whereas different deployments may require different balances between accuracy and cost. We therefore evaluate `RouteWeaver` with $\alpha \in \{0, 0.1, 0.3, 0.5\}$, keeping all other settings fixed; Figure 3 shows the resulting operating points. As α increases, Worker cost per query decreases monotonically, accompanied by a gradual decrease in accuracy. Relative to $\alpha = 0$, $\alpha = 0.1$ lowers cost by 23% while losing 1.0 point of accuracy, and $\alpha = 0.3$ lowers it by 61% while losing 3.1 points; at $\alpha = 0.5$, cost falls by 86% while accuracy drops by 11.2 points. These results illustrate different operating points that trade accuracy for efficiency under different deployment preferences.

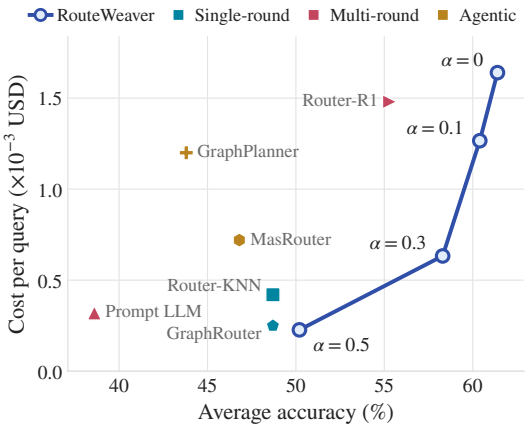


Figure 3: Fixed-reference Worker inference cost per query against average accuracy. The curve shows `RouteWeaver` trained with $\alpha \in \{0, 0.1, 0.3, 0.5\}$; markers show the two strongest baselines of each execution paradigm in Table 1. The $\alpha = 0$ cost is reconstructed from token usage.

Cost-aware routing provides different operating points under the same unified policy. All operating points are obtained from the same unified routing framework by changing only α, without manually designed strategies for different efficiency preferences. Section 5.5 analyzes how `RouteWeaver` distributes execution modes across domains.

5.4 Ablation of COMET-GRPO and Progressive Exploration

Table 2 ablates the two learning components of `RouteWeaver`. Both variants share the router initialization, Worker pool, training data, evaluation protocol, and training budget of the full model, and all use $\alpha = 0$.

w/o COMET-GRPO. Replacing COMET-GRPO with ordinary GRPO reduces average accuracy from 61.4% to 58.8% and shifts mode usage from .35/.41/.24 to .13/.82/.05, strongly concentrating the policy on multi-round execution. The largest drop occurs on Knowledge (5.8 points). This suggests that decision-specific credit assignment helps preserve differentiated mode selection rather than letting one execution mode dominate.

w/o progressive exploration. Removing the forced-to-free schedule causes a larger average drop, from 61.4% to 55.6%, with particularly large decreases on Code (9.6 points) and Reason (8.4 points). More strikingly, the learned policy collapses entirely to single-round execution (1.00/.00/.00). This supports the role of progressive exploration in maintaining cross-mode exploration before fully

Table 3: Accuracy (%) on benchmarks held out from router training across three domains: AIME (Math), GPQA-Diamond (Knowledge), and BBH (Reason). We compare one representative router from each execution paradigm.

Method	AIME	GPQA-D	BBH	Avg
GraphRouter (single-round)	23.5	49.5	24.7	32.6
Router-R1 (multi-round)	45.6	62.1	28.6	45.4
MasRouter (agentic)	28.1	43.8	28.2	33.4
`RouteWeaver`	**57.0**	**69.2**	**35.0**	**53.7**

autonomous mode selection. Notably, both ablated variants (58.8% and 55.6%) still exceed the strongest fixed-paradigm baseline (55.3%, Router-R1).

5.5 Routing Behavior and Generalization

Routing behavior across domains. Figure 4 reveals strongly task-dependent execution preferences. Math uses all three modes, with 19% of queries routed to single-round, 50% to multi-round, and 31% to agentic execution. Code is dominated by multi-round routing (84%), whereas Knowledge primarily uses single-round execution (72%). The contrast is strongest for Reason and Recall: 99% of reasoning queries are routed to agentic execution, while 96% of recall queries use single-round routing. These results show that `RouteWeaver` does not uniformly favor more complex execution, but instead learns substantially different mode preferences across task domains.

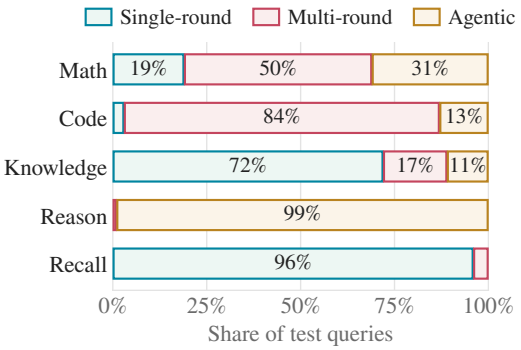


Figure 4: Share of selected execution modes across evaluation domains for `RouteWeaver` ($\alpha = 0$). Each bar shows the proportion of test queries routed to single-round, multi-round, and agentic execution.

Generalization to held-out benchmarks. To evaluate whether the learned routing policy transfers beyond the datasets used for router training, we further test `RouteWeaver` on AIME (AIMO, 2024; Zhang & Math-AI Team, 2025) for Math, GPQA-Diamond (GPQA-D) (Rein et al., 2024) for Knowledge, and BBH (Suzgun et al., 2023) for Reason. We compare against one representative baseline from each execution paradigm (Table 3). `RouteWeaver` achieves the best accuracy on all three held-out benchmarks, reaching an average of 53.7%, an 18.28% relative improvement over Router-R1 (45.4%). The gains are consistent across all three benchmarks, with improvements of 11.4, 7.1, and 6.4 points over Router-R1 on AIME, GPQA-D, and BBH, respectively. The learned execution behavior also varies across held-out benchmarks, with agentic execution accounting for 75% of AIME queries but only 12% of BBH queries. These results indicate that `RouteWeaver`'s adaptive routing behavior extends to unseen benchmarks within the same task domains.

6 Conclusion

We presented `RouteWeaver`, a unified LLM routing framework that treats the execution paradigm itself as a routing decision, jointly learning mode selection and within-mode execution. By combining COMET-GRPO with progressive exploration, `RouteWeaver` learns when to use single-round, multi-round, or agentic execution and how to route effectively within the selected mode. Across five task domains, `RouteWeaver` consistently outperforms fixed-paradigm routing baselines, exhibits task-dependent execution preferences, and supports different performance–cost operating points. These results suggest that effective LLM routing should adapt not only which models to invoke, but also how their computation is organized. Extending this unified perspective to richer execution structures and tool-augmented environments is an interesting direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementation and debugging, drafting and editing portions of the manuscript for clarity, and surveying and summarizing relevant literature. All AI-assisted code and manuscript content were reviewed by the authors, and AI-assisted code was inspected and tested before use. The authors take full responsibility for the final content, claims, and artifacts presented in this work.

ETHICS STATEMENT

This work studies routing among existing language models using established benchmarks and does not involve human-subject experiments or the collection of private user data. `RouteWeaver` inherits the limitations and potential harms of its underlying Worker models, including biased, unsafe, or factually incorrect outputs, which may propagate across multiple model calls. Our experiments are intended to evaluate routing methodology rather than safety-critical deployment. Practical deployments should therefore apply appropriate model-level safety mechanisms and comply with the licenses and usage policies of the underlying models and datasets.

REPRODUCIBILITY STATEMENT

We provide the complete formulation and training objectives of `RouteWeaver` in Section 4, and describe the datasets, router and Worker models, and baselines in Section 5.1. The appendices further document the routing serialization and execution constraints (Appendix A), dataset construction and evaluation sets (Appendix B), task-specific evaluation procedures (Appendix C), model and serving configurations (Appendix D), reward and cost definitions and optimization details (Appendix E), and training hyperparameters, decoding settings, and call budgets (Appendix F). Together, these sections document the training and evaluation setup used for `RouteWeaver`.

REFERENCES

AI-MO. AIMO validation AIME: AIME 2022–2024 problems. https://huggingface.co/datasets/AI-MO/aimo-validation-aime, 2024.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Trans. Mach. Learn. Res.*, 2024, 2024a. URL https://openreview.net/forum?id=cSimKw5p6R.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b. URL https://openreview.net/forum?id=7RQvjayHrM.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=02f3mUtqnM.

Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixin Deng, Shawn Gavin, Shian Jia, Sichao Jiang, Yiyan Liao, Rui Li, Qinrui Li, Sirun Li, Yizhi Li, Yunwen Li, David Ma, Yuansheng Ni, Haoran Que, Qiyao Wang, Zhoufutu Wen, Siwei Wu, Tyshawn Hsing, Ming Xu, Zhenzhu Yang, Zekun Moore Wang, Junting Zhou, Yuelin Bai, Xingyuan Bu, Chenglin Cai, Liang Chen, Yifan Chen, Chengtuo Cheng, Tianhao Cheng, Keyi Ding, Siming Huang, Yun Huang, Yaoru Li, Yizhe Li, Zhaoqun Li, Tianhao Liang, Chengdong Lin, Hongquan Lin, Yinghao Ma, Tianyang Pang, Zhongyuan Peng, Zifan Peng, Qige Qi, Shi Qiu, Xingwei Qu, Shanghaoran Quan, Yizhou Tan, Zili Wang, Chenqing Wang, Hao Wang, Yiya Wang, Yubo Wang, Jiajun Xu, Kexin Yang, Ruibin Yuan, Yuanhao Yue, Tianyang Zhan, Chun Zhang, Jinyang Zhang, Xiyue Zhang, Xingjian Zhang, Yue Zhang, Yongchi Zhao, Xiangyu Zheng, Chenghua Zhong, Yang Gao, Zhoujun Li, Dayiheng Liu, Qian Liu, Tianyu Liu, Shiwen Ni, Junran Peng,

Yujia Qin, Wenbo Su, Guoyin Wang, Shi Wang, Jian Yang, Min Yang, Meng Cao, Xiang Yue, Zhaoxiang Zhang, Wangchunshu Zhou, Jiaheng Liu, Qunshu Lin, Wenhao Huang, and Ge Zhang. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. doi: 10.52202/085713-3766. URL https://papers.nips.cc/paper_files/paper/2025/hash/a3c5af1f56fc73eef1ba0f442739f5ca-Abstract-Datasets_and_Benchmarks_Track.html.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. ReTool: Reinforcement learning for strategic tool use in LLMs. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026a. URL https://openreview.net/forum?id=tRk1nofSmz.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025a. URL http://papers.nips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A graph-based router for LLM selections. In *International Conference on Learning Representations (ICLR)*, 2025b. URL https://openreview.net/forum?id=eU39PDsZtT.

Tao Feng, Haozhen Zhang, Zijie Lei, Peixuan Han, and Jiaxuan You. GraphPlanner: Graph memory-augmented agentic routing for multi-agent LLMs. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://arxiv.org/abs/2604.23626.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=yaqPf0KAlN.

Gemini Team, Google. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. URL https://arxiv.org/abs/2507.06261.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025. URL https://arxiv.org/abs/2503.19786.

Lukas Haas, Gal Yona, Giovanni D'Antonio, Sasha Goldshtein, and Dipanjan Das. SimpleQA Verified: A reliable factuality benchmark to measure parametric knowledge. *arXiv preprint arXiv:2509.07968*, 2025. URL https://arxiv.org/abs/2509.07968.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks)*, 2021a. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c24cd76e1ce41366a4bbe8a49b02a028-Abstract-round2.html.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks)*, 2021b. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

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. URL https://arxiv.org/abs/2403.12031.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Ö. Arık, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Mandar Joshi, Eunsol Choi, Daniel 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. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*, 2023. URL https://arxiv.org/abs/2312.14852.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. GDPO: Group reward-decoupled normalization policy optimization for multi-reward RL optimization. *arXiv preprint arXiv:2601.05242*, 2026. URL https://arxiv.org/abs/2601.05242.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Second Conference on Language Modeling (COLM)*, 2025. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Llama Team, AI @ Meta. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

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. In *First Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Chenyang Shao, Xinyang Liu, Yutang Lin, Fengli Xu, and Yong Li. Route-and-reason: Energy-efficient scaling of LLM reasoning via reinforced model routing. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*. Association for Computing Machinery, 2026. doi: 10.1145/3774904.3793038. URL https://doi.org/10.1145/3774904.3793038.

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

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 AAAI Conference on Artificial Intelligence (AAAI)*, volume 40, pp. 33413–33421, 2026. doi: 10.1609/aaai.v40i39.40628. URL https://ojs.aaai.org/index.php/AAAI/article/view/40628.

Anjiang Wei, Yuheng Wu, Yingjia Wan, Tarun Suresh, Huanmi Tan, Zhanke Zhou, Sanmi Koyejo, Ke Wang, and Alex Aiken. SATBench: Benchmarking LLMs' logical reasoning via automated puzzle generation from SAT formulas. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33832–33849. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1716. URL https://aclanthology.org/2025.emnlp-main.1716/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.757. URL https://aclanthology.org/2025.acl-long.757/.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-R1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=DWf4vroKWJ.

Yifan Zhang and Math-AI Team. American invitational mathematics examination (AIME) 2025. https://huggingface.co/datasets/math-ai/aime25, 2025.

A Action Format

The router's output is written with the four elements illustrated in Table 4: one `mode` element, followed by `route` elements interleaved with environment-inserted `observation` elements, and a final `answer` element. Workers are named by their anonymous IDs.

Table 4: Schematic of the shared trajectory representation.

Element	Illustrative form	Produced by
Mode selection	`<mode>agentic</mode>`	Router
Worker call	`<route model="worker_2" role="verifier" refs="n1"> Check the proposed solution.</route>`	Router
Returned output	`<observation id="n2">...</observation>`	Environment
Final response	`<answer>...</answer>`	Router

Every Worker receives the original query together with the sub-query written in its `route` element. A trajectory must follow the grammar and the execution constraints of its mode:

- **Single-round:** exactly one call `<route model="`w`">`s`</route>`, followed by the final answer.
- **Multi-round:** two to four sequential calls of the same form, one per router turn; each call after the first also receives the previous observation.
- **Agentic:** calls of the form
  `<route model="`w`" role="`r`" refs="`P`">`s`</route>`,
  with $r \in \{$planner, executor, verifier, summarizer$\}$ and P a comma-separated list of node IDs (empty for the first layer). Several routes in one router turn form a layer and run in parallel; a route may reference only outputs of earlier layers, which the environment inserts into that Worker's input. Each output returns as `<observation id="n`k`" role="`r`">`. At most eight Worker calls and ten router action turns are allowed.

The first action that violates the grammar or an execution constraint of the mode terminates the trajectory with rewards $(-1, -1)$ (Appendix E). Figure 5 shows one abbreviated trajectory per mode from the evaluation of the reported router.

B Dataset Details

Training data. The router is trained on 3,200 queries from seven datasets (Table 5). Each training batch contains 32 queries with fixed domain quotas: 7 Math, 7 Code (4 APPS and 3 LiveCodeBench), 6 Knowledge, 7 Reason, and 5 Recall.

Evaluation data. Evaluation uses 1,060 test questions that do not overlap with the training questions: 120 each from MATH, Omni-MATH, APPS, LiveCodeBench, SuperGPQA, SATBench, TriviaQA, and SimpleQA-Verified, and 100 from TACO.

C Evaluation Protocol

Per-dataset scoring. Every question receives a binary score, and we report accuracy per dataset.

Single-round

```
<mode>single</mode>
<route model="worker_3">If $x$ and $y$ are positive integers with $3^x 5^y = 225$,
  what is the value of $x + y$?</route>
<observation>...</observation>
<answer>4</answer>
```

Multi-round

```
<mode>multi</mode>
<route model="worker_3">Calculate the number of three-digit integers from 100 to 999
  whose digit sum is between 7 and 11 inclusive.</route>
<observation>...</observation>
<route model="worker_3">Verify that the total number of three-digit integers with
  digit sum between 7 and 11 inclusive is 224, and confirm the last two digits
  are 24.</route>
<observation>...</observation>
<answer>24</answer>
```

Agentic

```
<mode>agentic</mode>
<route model="worker_3" role="planner" refs="">Split the problem into sub-tasks: ...
  </route>
<observation id="n1" role="planner">...</observation>
<route model="worker_3" role="executor" refs="n1">Compute the number of unit cubes
  with 0 gold faces and exactly 1 gold face for $n = 9$, ...</route>
<observation id="n2" role="executor">...</observation>
<route model="worker_3" role="verifier" refs="n1,n2">Verify that the formulas ...
  </route>
<observation id="n3" role="verifier">...</observation>
<route model="worker_3" role="summarizer" refs="n1,n2,n3">Reconcile the planner's
  decomposition, the executor's computation, and the verifier's audit ...</route>
<observation id="n4" role="summarizer">...</observation>
<answer>9</answer>
```

Figure 5: Abbreviated trajectories of the reported router, one per mode (Omni-MATH evaluation questions). Observation bodies and parts of long sub-queries are elided (. . .), the routing-state messages that the environment inserts between agentic layers are omitted, and line breaks inside routes are added for display.

Table 5: Training data by domain.

Domain	Dataset	#Queries	Per batch
Math	Omni-MATH	700	7
Code	APPS, LiveCodeBench	400, 300	4, 3
Knowledge	SuperGPQA	600	6
Reason	SATBench	700	7
Recall	TriviaQA, PopQA (Mallen et al., 2023)	200, 300	5
Total		3,200	32

- **MATH, Omni-MATH:** symbolic mathematical equivalence with the reference answer.
- **APPS:** the extracted program is executed in a sandbox on standard input and is correct if its output matches on all test cases.
- **LiveCodeBench, TACO:** extracted code is executed and is correct if it passes all retained test cases. At most 40 test cases are kept per problem; for LiveCodeBench, test cases are additionally kept in their original order only while their cumulative serialized size stays within 256 KB (at least one is always kept). Both rules are deterministic, so the full official test suites are not always used.
- **SuperGPQA:** the last option letter in the response must match the reference exactly.
- **SATBench:** the last occurrence of SAT or UNSAT in the response must match the reference label.
- **SimpleQA-Verified:** normalized exact match, or the reference answer appears in the response as a whole-word match.
- **TriviaQA:** exact match with any reference alias after SQuAD-style normalization (lowercasing and removing punctuation, articles, and extra whitespace).

Scored candidates. The scorer does not look only at the router's final answer. For non-code tasks it checks only the router's final <answer>. For code tasks, a union scorer treats code blocks in Worker outputs as candidates in addition to the router's answer. A single system rollout can therefore expose more than one candidate, and the number of candidates depends on the execution mode, so the metric is not standard pass@1.

Aggregation. Domain accuracy is the unweighted mean of the accuracies of its constituent datasets, and Avg is the unweighted mean of the five domain accuracies.

D Model Details

Router. Qwen3-4B-Instruct-2507, with full-parameter fine-tuning. The KL reference policy is the frozen initial router. Only the router is trained.

Workers. The six Workers are frozen and are exposed to the router under fixed anonymous IDs. The router's system prompt lists each ID with a fixed capability description (Table 6) and never contains model names or prices. worker_1 and worker_2 are deployed locally on two NVIDIA A100 GPUs, and worker_3 to worker_6 are accessed through OpenRouter; the serving setup is the same in training and evaluation.

Table 6: Worker pool and the capability descriptions shown to the router.

ID	Model	Capability description in the router prompt
worker_1	Qwen3-8B	Dense language model with 8.2B parameters. Supports multilingual instruction following, dialogue, mathematical and logical reasoning, and code generation.
worker_2	Llama-3.1-8B-Instruct	Dense language model with 8B parameters. Instruction-tuned for multilingual dialogue, with capabilities in general reasoning, text generation, and coding.
worker_3	Qwen3-30B-A3B-Instruct-2507	Mixture-of-experts language model with 30.5B total and 3.3B active parameters. Supports instruction following, text comprehension, logical reasoning, mathematics, science, and coding.
worker_4	Gemma-3-27B-IT	Dense language model in the 27B parameter class. Supports multilingual question answering, summarization, reasoning, mathematical problem solving, and code generation.
worker_5	Llama-3.3-70B-Instruct	Dense language model with 70B parameters. Instruction-tuned for multilingual dialogue, with capabilities in reasoning, instruction following, and text and code generation.
worker_6	Gemini-2.5-Flash-Lite	Sparse mixture-of-experts model with undisclosed parameter counts. Supports translation, classification, information extraction, coding, mathematics, and reasoning.

E Training Details

This section completes the definitions of Section 4.

Rewards and excluded rollouts. Each trajectory receives the two rewards of Eq. 5. Correctness is judged by the dataset-specific scorers of Appendix C, whose candidate set can extend beyond the router's final answer. The first action that violates the grammar or an execution constraint of the mode (Appendix A) terminates the trajectory, which receives $(-1, -1)$, remains in $\mathcal{V}$, and enters the group statistics like any other rollout. A rollout affected by an infrastructure failure, such as a Worker call returning no response, is not attributed to the router and is not treated as terminal-invalid; it is excluded from $\mathcal{V}$ and hence from all group statistics, the policy loss, and the KL term. When the efficiency reward is enabled ($\alpha > 0$), trajectories with incomplete cost records are excluded from $\mathcal{V}$ in the same way, so they enter neither reward component nor the corresponding updates.

Cost measure. $C(\tau)$ sums the inference cost of every Worker call in τ, computed from the call's input and output tokens at the fixed reference prices of Table 7 (in USD per million tokens). The same frozen table is used for the efficiency reward and for all reported costs. These prices serve only to measure Worker inference cost; they are not the amounts actually paid, and the router's own cost is not included. The normalization constant $C_{\text{ref}} = 0.0070716653$ USD is the 95th percentile of the per-trajectory Worker cost over 1,024 rollouts (128 training queries, 8 rollouts each) sampled from the step-25 cold-start checkpoint with the training decoding settings, excluding only infrastructure failures and trajectories with incomplete usage records. It is frozen for all runs, so that $S_{\text{cost}}(\tau) = 1/2$ when $C(\tau) = C_{\text{ref}}$.

Table 7: Fixed reference prices (USD per million tokens).

ID	Model	Input	Output
worker_1	Qwen3-8B	0.020	0.050
worker_2	Llama-3.1-8B-Instruct	0.020	0.050
worker_3	Qwen3-30B-A3B-Instruct-2507	0.100	0.300
worker_4	Gemma-3-27B-IT	0.119	0.200
worker_5	Llama-3.3-70B-Instruct	0.100	0.320
worker_6	Gemini-2.5-Flash-Lite	0.100	0.400

Component-wise group statistics. For a reward component $c \in \{\text{task}, \text{eff}\}$ with values $r_{i,c}$, a group with retained rollouts $\mathcal{V}$ and mode subsets $\mathcal{V}_m$ has

$$\mu_c = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} r_{i,c}, \qquad \mu_{m,c} = \frac{1}{|\mathcal{V}_m|} \sum_{i \in \mathcal{V}_m} r_{i,c}, \qquad \sigma_c = \text{std}\left(\{r_{i,c}\}_{i \in \mathcal{V}}\right), \tag{14}$$

and advantages

$$A^{\text{GRPO}}_{i,c} = \frac{r_{i,c} - \mu_c}{\sigma_c + \epsilon}, \qquad A^{\text{mode}}_{i,c} = \frac{\mu_{m_i,c} - \mu_c}{\sigma_c + \epsilon}, \qquad A^{\text{inner}}_{i,c} = \frac{r_{i,c} - \mu_{m_i,c}}{\sigma_c + \epsilon}. \tag{15}$$

Here σ_c is the sample (Bessel-corrected) standard deviation over all retained rollouts of the group, not a per-mode one, and $\epsilon = 10^{-6}$. The advantages used in training mix the components with the run's fixed weight α,

$$A^l_i = (1 - \alpha)\, A^l_{i,\text{task}} + \alpha\, A^l_{i,\text{eff}}, \qquad l \in \{\text{GRPO}, \text{mode}, \text{inner}\}, \tag{16}$$

which extends Eq. 7 to the ordinary GRPO advantage used by forced groups. Each component is normalized by its own group statistics before mixing; the rewards are never combined into a scalar and normalized jointly, and no further batch-level normalization or whitening is applied.

Component-wise gating. Each component is gated separately: if $\sigma_c < 10^{-4}$, component c contributes no advantage for that group. A group whose rollouts are all correct but differ in cost therefore keeps its efficiency signal when $\alpha > 0$, although its task signal is gated. A group whose rollouts are all excluded from $\mathcal{V}$ contributes no gradient.

Ordinary GRPO loss. With $S_i = S^{\text{mode}}_i \cup S^{\text{inner}}_i$ and $N_{\text{pol}} = \sum_i |S_i|$,

$$\mathcal{L}^{\text{global}}_{\text{GRPO}} = \frac{1}{N_{\text{pol}}} \sum_i \sum_{k \in S_i} \ell_{ik}(A^{\text{GRPO}}_i). \tag{17}$$

In cold start every mode prefix is injected, so $S^{\text{mode}}_i = \varnothing$ and S_i contains only execution tokens, and only the task component is used, $A^{\text{GRPO}}_i = A^{\text{GRPO}}_{i,\text{task}}$.

Dual clipping. For negative advantages, the token loss of Eq. 8 is additionally bounded as $\ell_{ik}(A) \leftarrow \min\{\ell_{ik}(A), -cA\}$ for $A < 0$, with $c = 3$, which limits the loss of a negative-advantage token whose ratio has grown large. The clipping ranges are $\eta_- = \eta_+ = 0.2$.

KL regularization. $\mathcal{L}_{\text{KL}}$ in Eq. 10 is computed against the frozen initial router, the original Qwen3-4B-Instruct-2507 rather than the cold-start checkpoint, with coefficient β (Appendix F); κ_{ik} is the low-variance (k_3) estimator, and the KL term enters the loss rather than the reward. It is normalized by its own token count N_{KL} over Ω_{KL}, not by the segment counts of Eq. 9. Ω_{KL} uses the same router-token mask as the policy loss, so rollouts excluded from $\mathcal{V}$ by infrastructure failures are removed from both.

Mixed-batch token advantages. During the transition, the token advantage used in Eq. 9 is

$$\hat{A}_{ik} = \begin{cases} A_i^{\text{mode}}, & k \in S_i^{\text{mode}}, \text{ free group}, \\ A_i^{\text{inner}}, & k \in S_i^{\text{inner}}, \text{ free group}, \\ A_i^{\text{GRPO}}, & k \in S_i^{\text{inner}}, \text{ forced group}, \end{cases} \tag{18}$$

where all advantages are the α-mixed values of Eq. 16. A group is either entirely forced or entirely free. For forced groups the injected mode tokens are masked, so $S_i^{\text{mode}} = \varnothing$ and the execution tokens receive A_i^{GRPO}, which equals A_i^{inner} because all rollouts of the group share one mode; these tokens count toward N_{inner}.

Support-term scores. Let $s_\theta(m, x)$ be the summed log probability of the content tokens of candidate mode m for query x. The scores are normalized per query and averaged over the set $\mathcal{U}$ of unique queries in the batch:

$$\tilde{p}_\theta(m \mid x) = \frac{\exp s_\theta(m, x)}{\sum_{m' \in \mathcal{M}} \exp s_\theta(m', x)}, \qquad \bar{p}_m = \frac{1}{|\mathcal{U}|} \sum_{x \in \mathcal{U}} \tilde{p}_\theta(m \mid x). \tag{19}$$

$\bar{p}_m$ is thus an average of normalized candidate-mode scores, not the frequency with which m is sampled, and the support term does not ensure that every mode is sampled in every batch.

Degenerate groups. For each component, if a mode has only one retained rollout, then $\mu_{m_i,c} = r_{i,c}$ and that rollout's inner advantage is zero; if all retained rollouts select the same mode, then $\mu_{m_i,c} = \mu_c$ and all mode advantages are zero. Gating is described above.

Relation to GRPO advantages. For each component, $A_{i,c}^{\text{mode}} + A_{i,c}^{\text{inner}} = A_{i,c}^{\text{GRPO}}$, and hence the mixed advantages satisfy $A_i^{\text{mode}} + A_i^{\text{inner}} = A_i^{\text{GRPO}}$. This is an identity between advantage values only. The two terms are assigned to disjoint token sets rather than added on any token, and the resulting losses are normalized separately, so the identity does not imply that the COMET-GRPO objective or gradient coincides with that of GRPO; even under a single normalization, the identity alone would not establish such an equivalence. Likewise, mixing separately normalized advantages is not equivalent to normalizing a fixed weighted combination of the raw rewards, and we do not claim that the update follows an unbiased gradient of any fixed weighted raw reward.

F Implementation Details

Training. Batch size 32 queries, 8 rollouts per query, learning rate 10^{-6}, KL coefficient 0.001 against the frozen initial router, entropy coefficient 0, and 90 training steps. Steps 1–25 form the cold-start phase: forced modes, standard GRPO on the task reward with the standard-deviation gate, global token reduction, and no support term. Training then resumes from the step-25 checkpoint, restoring model, optimizer, and dataloader state, and enables the mode/inner decomposition and segment-wise reduction. The forced ratio ρ_b decreases linearly from 1 at data batch index 25 to 0 at batch index 75, and the support coefficient is $0.002\,\rho_b$. Runs with $\alpha \in \{0.1, 0.3, 0.5\}$ share this task-only cold start: each resumes from the same step-25 checkpoint and continues the original training and curriculum progress, with α fixed for the whole run and the KL reference still the frozen initial router. Every run, including $\alpha = 0$, is trained for 90 steps and evaluated at its final checkpoint. The support floor is $\delta = 0.05$ with $\lambda_0 = 0.002$, the clipping ranges are $\eta_- = \eta_+ = 0.2$, and dual clipping uses $c = 3$. The optimizer is AdamW with $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 0.01, using a constant learning rate without warmup or decay and global gradient-norm clipping at 1.0. Each step performs a single on-policy update over all $32 \times 8 = 256$ trajectories of the batch, so π_{old} is the policy that generated them. In the ablation without progressive exploration, $\rho_b = 0$ for all b, so the support term ($\lambda_b = \lambda_0 \rho_b$) is inactive as well.

Decoding. Router, training: temperature 0.8, top-p 0.95, top-k disabled. Router, testing: greedy decoding. Workers, training and testing: temperature 0.2, at most 4,096 output tokens; top-p and top-k are not set and follow provider defaults. Worker requests carry a fixed seed (42), which providers may ignore, and disable thinking/reasoning modes. Because Workers sample, system-level evaluation is not fully deterministic. Each evaluation question receives one system rollout.

Call budgets and lengths. Identical in training and evaluation. Single-round: exactly one Worker call. Multi-round: two to four sequential calls. Agentic: at most eight Worker calls and at most ten router action turns. Prompts are limited to 6,144 tokens, the multi-turn response budget to 12,288 tokens, the mode-declaration turn to 64 tokens, and each observation to 2,048 tokens.