FLOWROUTER: EXPERIENCE-GUIDED ROUTING FOR QUERY-ADAPTIVE MULTI-AGENT WORKFLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM)-based multi-agent systems can improve problem solving through collaborative reasoning, yet a uniform workflow may incur redundant computation for some queries while providing insufficient reasoning capacity for others. Current methods include both manually designed and automatically optimized workflows, but adapting complete workflow configurations to individual queries remains challenging. To address this, we propose FLOWROUTER, an experience-guided framework for query-adaptive multi-agent workflow routing. FlowRouter adopts an explicit graph-based factorization of workflow construction into conditionally dependent decisions over topology, collaboration scale, and position-wise LLM assignment. The resulting routing decisions are jointly optimized under a shared execution-level performance–cost reward, with retrieved workflow experience providing auxiliary guidance. Across four benchmarks, FLOWROUTER achieves the highest macro-average performance among the evaluated methods while maintaining a favorable inference-cost trade-off. Further analysis shows that topology selection varies significantly across MATH subfields but not across official difficulty levels, suggesting that workflow adaptation reflects characteristics beyond scalar difficulty.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems have emerged as a promising way to extend the capabilities of individual agents through iterative generation, refinement, verification, and aggregation (Zhuge et al., 2024; Hong et al., 2024; Qian et al., 2024; Zhao et al., 2024). Their effectiveness can depend not only on the capabilities of individual models, but also on how multiple LLM calls are organized and how intermediate information is exchanged throughout the reasoning process (Qian et al., 2025; Hu et al., 2025). However, these additional calls and interactions also incur extra inference cost (Zhuge et al., 2024; Wang et al., 2025). Multi-agent orchestration is therefore not simply a matter of invoking more agents, but of deciding how a complete workflow should be constructed for each query so that collaborative computation is used effectively.

Recent work has explored several directions toward multi-agent system design. Early agentic systems largely relied on manually designed workflows, following predefined coordination procedures analogous to standardized operating procedures in human organizations (Park et al., 2023; Hong et al., 2024). Such workflows specify how LLM calls are arranged, how intermediate results are exchanged, and what instructions individual agents follow, but their construction requires substantial human effort and often relies on domain expertise. This limitation has motivated increasing efforts toward automatic workflow design. Methods such as GPTSwarm (Zhuge et al., 2024), EvoAgent (Yuan et al., 2025), ADAS (Hu et al., 2025), AFlow (Zhang et al., 2025b), and AgentSquare (Shang et al., 2025) automate different aspects of agent construction, including communication, agent profiling, and workflow search, with ADAS and AFlow further adopting flexible programmatic representations to support automated workflow optimization. However, these designs typically rely on shared and relatively fixed workflows. This can incur unnecessary computation for some queries while providing insufficient adaptation for others.

More recent work has moved toward query-specific workflow construction. MaAS (Zhang et al., 2025a) samples a customized multi-agent architecture from an agentic supernet for each query, while DAAO (Su et al., 2026) dynamically adjusts workflow complexity according to estimated

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.

query difficulty. MasRouter further introduces query-dependent decisions over collaboration modes, roles, and LLMs (Yue et al., 2025). These developments reflect a shift toward increasingly adaptive workflow construction and motivate a common question of how query-specific workflow adaptation can be made sufficiently fine-grained to account for dependencies among configuration decisions.

Based on this, we study query-adaptive multi-agent workflow routing through a graph-based factorization, providing a natural way to organize multiple workflow decisions within a unified representation. A workflow is characterized by its topology, collaboration scale, and position-wise LLM assignment. The topology determines how intermediate information can propagate, while the scale determines how many computational positions are instantiated within the workflow. These structural choices establish the information context of each position: some positions operate directly on the query, whereas others receive intermediate outputs for refinement, aggregation, or verification (Zhuge et al., 2024; Qian et al., 2025). Model suitability can therefore vary with the position induced by the selected topology and scale. We capture this dependency by factorizing workflow construction into a sequence of conditionally dependent decisions over these dimensions. This factorization allows each dimension to remain individually configurable while maintaining their structural relationships throughout workflow construction.

In this work, we propose FLOWROUTER, an experience-guided framework for query-adaptive multi-agent workflow routing. Given a query, FLOWROUTER follows the proposed factorization to construct an executable directed acyclic workflow and assign models to its ordered agent positions. All routing decisions are jointly optimized under a shared execution-level reward that evaluates the resulting workflow as a whole by balancing task performance against inference cost. To further inform routing, FLOWROUTER maintains an experience memory of training-time executions and retrieves semantically related records whose observed outcomes and costs provide useful auxiliary guidance for subsequent decisions.

Experiments across four benchmarks show that FLOWROUTER is a high-performance and cost-efficient framework for query-adaptive multi-agent orchestration. It achieves the highest average task performance of 92.94% among the evaluated methods. It also improves efficiency at both inference and training time, exhibiting favorable performance–cost trade-offs and reducing normalized training API cost by more than 30% relative to the evaluated optimization and routing baselines. Beyond aggregate performance, the learned routing behavior also exhibits meaningful variation across queries. On MATH, topology selection is significantly associated with problem subdomains but not with the benchmark's official difficulty levels, suggesting that workflow structure reflects characteristics beyond scalar difficulty. Our main contributions are as follows:

- We introduce a structured view of LLM-based multi-agent workflows as jointly determined by collaboration topology, scale, and position-wise LLM assignment. This perspective enables separate control over the main workflow dimensions while preserving their structural dependencies.
- We propose FLOWROUTER, an experience-guided framework that progressively constructs query-specific heterogeneous workflows through topology selection, scale instantiation, and position-wise model assignment. All routing decisions are jointly optimized under a shared execution-level performance–cost objective, with retrieved execution experience providing auxiliary guidance.
- Experiments across four benchmarks demonstrate strong performance–cost trade-offs and show that FLOWROUTER adaptively matches workflow configurations to task characteristics.

2 RELATED WORK

Agentic Workflows. LLM-based agentic workflows organize multiple model invocations into structured reasoning processes (Li et al., 2026). Existing systems employ strategies such as iterative refinement, reflection, role-based interaction, and graph-structured communication (Wei et al., 2022; Du et al., 2024; Hong et al., 2024). Multi-agent workflows further coordinate multiple agents through predefined interaction patterns. For example, MacNet (Qian et al., 2025) represents collaboration as a directed acyclic graph and studies how network topology affects collective reasoning. Many such workflows are manually specified and apply the same collaboration structure across inputs, motivating automated workflow construction.

Automated Workflow Design. Recent work has increasingly automated agent and workflow construction. GPTSwarm (Zhuge et al., 2024) optimizes computational graphs, ADAS (Hu et al., 2025) searches over executable agentic programs, AFlow (Zhang et al., 2025b) performs code-based

workflow search, and AgentSquare (Shang et al., 2025) explores modular agent designs. These approaches substantially reduce manual workflow engineering, but are primarily oriented toward reusable task-level configurations. More recent methods move toward query-specific workflow construction. MaAS (Zhang et al., 2025a) samples query-dependent architectures from an agentic supernet, while DAAO (Su et al., 2026) adapts workflow complexity according to estimated query difficulty. These developments motivate workflow representations that can express input-specific structural variation while retaining dependencies among configuration decisions.

LLM and Multi-Agent Routing. LLM routing dynamically selects models according to query characteristics, model capabilities, and inference cost (Chen et al., 2024a; Ding et al., 2024; Ong et al., 2025). These methods aim to balance predictive quality and computational cost by matching different inputs with suitable model backbones. Most existing routing approaches focus on single-agent inference, where the primary decision concerns which model should process a given query. MasRouter (Yue et al., 2025) extends routing to multi-agent systems, broadening the scope of routing from individual model selection to collaborative inference. Our work further considers topology, collaboration scale, and position-wise LLM assignment within graph-structured workflow routing.

3 METHOD

Figure 1 illustrates the overall design of the proposed FLOWROUTER framework. Given a query Q, FLOWROUTER sequentially routes the collaboration topology and scale to instantiate a workflow graph, and then assigns an LLM to each ordered graph position. Retrieved execution experience provides auxiliary evidence for the routing decisions, while the resulting workflow is optimized using a shared performance–cost signal.

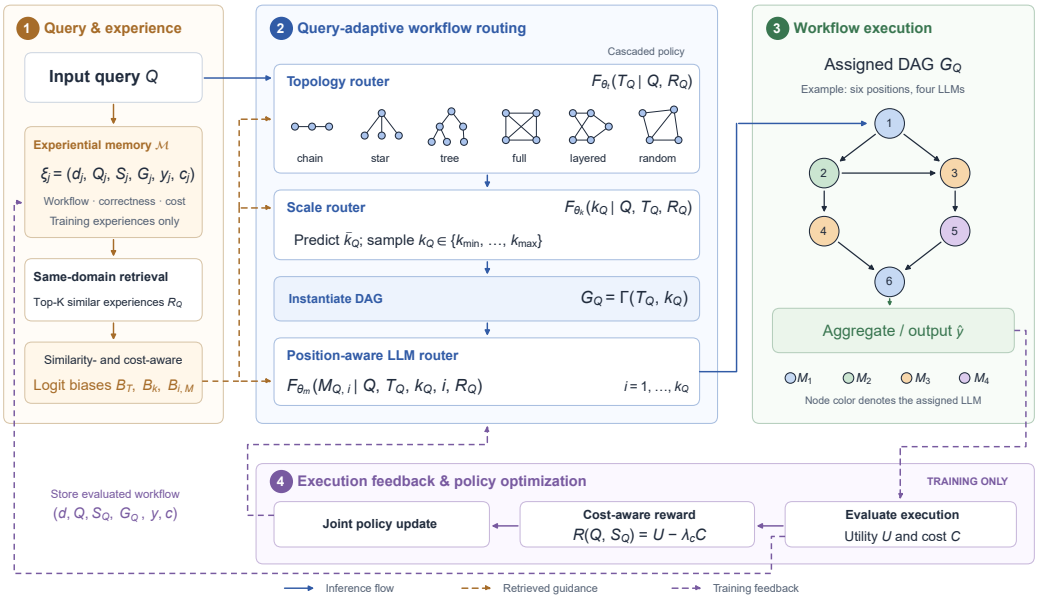


Figure 1: Overall framework of the proposed FLOWROUTER.

3.1 PROBLEM FORMULATION

Following graph-based multi-agent collaboration (Zhuge et al., 2024; Qian et al., 2025), we represent an executable workflow as a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of LLM-powered agent nodes and $\mathcal{E}$ is the set of directed edges that specify intermediate-output propagation. Agents are executed according to a valid topological ordering of $\mathcal{G}$.

Let $\mathcal{T}$ denote the set of candidate topology families and $\mathcal{L}$ the set of available LLM backbones. For query Q, the routed workflow configuration $\mathcal{S}_Q$ is

$$\mathcal{S}_Q = \left(T_Q,\, k_Q,\, \{M_{Q,i}\}_{i=1}^{k_Q}\right), \tag{1}$$

where $T_Q \in \mathcal{T}$ is the selected collaboration topology, k_Q is the number of instantiated agent positions, and $M_{Q,i} \in \mathcal{L}$ is the LLM assigned to position i. The topology and scale instantiate the executable graph $\mathcal{G}_Q = \Gamma(T_Q, k_Q)$, where Γ denotes the graph instantiation procedure.

Let $\mathcal{R}_Q$ denote the set of execution experiences retrieved for query Q. FLOWROUTER factorizes the routing distribution over $\mathcal{S}_Q$ as

$$\begin{aligned} F_\theta(\mathcal{S}_Q \mid Q, \mathcal{R}_Q) = F_{\theta_t}(T_Q \mid Q, \mathcal{R}_Q) F_{\theta_k}(k_Q \mid Q, T_Q, \mathcal{R}_Q) \\ \times \prod_{i=1}^{k_Q} F_{\theta_m}(M_{Q,i} \mid Q, T_Q, k_Q, i, \mathcal{R}_Q), \end{aligned} \tag{2}$$

where F_{θ_t}, F_{θ_k}, and F_{θ_m} are the topology, scale, and position-wise LLM routing distributions, respectively, and $\theta = \{\theta_t, \theta_k, \theta_m\}$ denotes their trainable routing parameters. This factorization makes the scale decision conditional on the selected topology and the model assignment conditional on the instantiated topology and scale.

3.2 QUERY-ADAPTIVE WORKFLOW ROUTING

Topology Routing. We consider $\mathcal{T} = \{\texttt{chain}, \texttt{star}, \texttt{tree}, \texttt{full}, \texttt{layered}, \texttt{random}\}$. Each topology candidate $T \in \mathcal{T}$ is associated with a natural-language description D_T. For query Q and topology T, their representations are $H_Q = g_{\phi_q}(f_\psi(Q))$ and $H_T = g_{\phi_t}(f_\psi([T; D_T]))$, where H_Q and H_T denote the query and topology representations, f_ψ is a frozen text encoder, and g_{ϕ_q} and g_{ϕ_t} are trainable transformations with parameters ϕ_q and ϕ_t.

The topology routing distribution is

$$F_{\theta_t}(T \mid Q, \mathcal{R}_Q) \propto \exp\left(\frac{s_{\theta_t}(H_Q, H_T)}{\tau}\right) G_T(\mathcal{R}_Q), \qquad T \in \mathcal{T}, \tag{3}$$

where s_{θ_t} is the query–topology compatibility function, $\tau > 0$ is the routing temperature, and $G_T(\mathcal{R}_Q)$ is the experience-guidance factor for topology T defined in Section 3.3.

Adaptive Collaboration Scale. Conditioned on query Q and the selected topology T_Q, FLOWROUTER predicts a continuous reference scale

$$\bar{k}_Q = k_{\min} + \delta_{\theta_k}(H_Q, H_{T_Q})\,(k_{\max} - k_{\min}), \tag{4}$$

where $\bar{k}_Q$ is the reference scale, $k_{\min}$ and $k_{\max}$ are the minimum and maximum allowed numbers of agents, and $\delta_{\theta_k}(\cdot) \in [0, 1]$ is a trainable scale predictor.

Let $\mathcal{K} = \{k_{\min}, \ldots, k_{\max}\}$ denote the candidate scale set. The discrete routing distribution is

$$F_{\theta_k}(k \mid Q, T_Q, \mathcal{R}_Q) = \frac{\exp\left(-\frac{(k-\bar{k}_Q)^2}{2\sigma_k^2}\right) G_k(\mathcal{R}_Q)}{\sum_{k' \in \mathcal{K}} \exp\left(-\frac{(k'-\bar{k}_Q)^2}{2\sigma_k^2}\right) G_{k'}(\mathcal{R}_Q)}, \tag{5}$$

where $k \in \mathcal{K}$ is a candidate scale, $\sigma_k > 0$ controls the dispersion around $\bar{k}_Q$, and $G_k(\mathcal{R}_Q)$ is the experience-guidance factor for scale k. After sampling k_Q, the selected topology and scale instantiate $\mathcal{G}_Q = \Gamma(T_Q, k_Q)$. For the `random` topology, a chain backbone is retained and additional forward edges are sampled to preserve connectivity.

Position-Aware LLM Routing. After $\mathcal{G}_Q = (\mathcal{V}_Q, \mathcal{E}_Q)$ is instantiated, FLOWROUTER assigns an LLM to each graph position. We index the nodes according to a fixed valid topological ordering such that $(v_i, v_j) \in \mathcal{E}_Q$ implies $i < j$, where v_i denotes the node at position i. Its position representation is $P_i = \mathrm{Emb}_{\mathrm{pos}}(i)$, where $\mathrm{Emb}_{\mathrm{pos}}$ is a learnable positional embedding function.

For each candidate model $M \in \mathcal{L}$, let H_M denote its model representation. The position-wise assignment distribution is

$$F_{\theta_m}(M \mid Q, T_Q, k_Q, i, \mathcal{R}_Q) = \frac{\exp\left(\frac{s_{\theta_m}(H_Q, H_{T_Q}, P_i, H_M)}{\tau}\right) G_{i,M}(\mathcal{R}_Q)}{\sum_{M' \in \mathcal{L}} \exp\left(\frac{s_{\theta_m}(H_Q, H_{T_Q}, P_i, H_{M'})}{\tau}\right) G_{i,M'}(\mathcal{R}_Q)}, \tag{6}$$

where M' ranges over candidate LLMs, s_{θ_m} is the compatibility function for position-wise model routing, and $G_{i,M}(\mathcal{R}_Q)$ is the experience-guidance factor for assigning model M to position i. The resulting model assignment therefore depends on both the query and the structural context established by the preceding topology and scale decisions.

3.3 Experience-Guided Routing

FlowRouter maintains an experience memory of workflows executed during training. For training query Q_j, we store $\xi_j = (d_j, Q_j, \mathcal{S}_j, \mathcal{G}_j, y_j, c_j)$, where d_j is the task domain, $\mathcal{S}_j$ the routed workflow, $\mathcal{G}_j$ the instantiated graph, and y_j and c_j the execution outcome and cost.

For a query Q, FlowRouter retrieves up to K semantically similar training experiences from the same domain. The retrieved experiences are aggregated according to their similarity, execution outcome, and cost to produce experience-guidance factors for the candidate routing decisions. We denote these factors by $G_T(\mathcal{R}_Q)$, $G_k(\mathcal{R}_Q)$, and $G_{i,M}(\mathcal{R}_Q)$ for topology selection, scale selection, and assigning model M to position i, respectively, where $\mathcal{R}_Q$ denotes the retrieved experience set. These factors modulate the corresponding routing distributions defined above. When no relevant experience is retrieved, the guidance factors are set to 1, leaving the learned routing distributions unchanged. Test queries may retrieve training experiences but are not written back to memory. Detailed computation of the experience-guidance factors is provided in Appendix A.

3.4 Optimization

The routing decisions in Eq. 2 are discrete, and workflow execution involves black-box LLM calls. We therefore optimize FlowRouter using REINFORCE. For a sampled workflow configuration $\mathcal{S}_Q$, the execution-level reward is

$$R(Q, \mathcal{S}_Q) = \mathcal{U}(\mathcal{O}(Q, \mathcal{S}_Q), a) - \lambda_c \mathcal{C}(Q, \mathcal{S}_Q), \tag{7}$$

where $R(Q, \mathcal{S}_Q)$ is the workflow reward, $\mathcal{O}(Q, \mathcal{S}_Q)$ is the workflow output, a is the reference answer, $\mathcal{U}$ is the task-utility function, $\mathcal{C}(Q, \mathcal{S}_Q)$ is the execution cost, and λ_c controls the performance–cost trade-off.

Let b denote an exponential moving average of historical rewards used as the REINFORCE baseline. The routing objective is

$$\mathcal{L}_{\text{FlowRouter}} = -\mathbb{E}\left[(R - b) \log F_\theta(\mathcal{S}_Q \mid Q, \mathcal{R}_Q) + \lambda_H \mathcal{H}(F_\theta(\cdot \mid Q, \mathcal{R}_Q))\right], \tag{8}$$

where $\mathcal{L}_{\text{FlowRouter}}$ is the training objective, $\mathbb{E}$ denotes expectation over sampled workflow executions, $\mathcal{H}$ is the entropy of the factorized routing distribution, and λ_H controls the entropy-regularization strength. Because $R(Q, \mathcal{S}_Q)$ is computed from execution of the complete workflow, topology, scale, and position-wise LLM routing are optimized under the same performance–cost signal. The complete training procedure is provided in Algorithm 1 in Appendix B.

4 Experiments

4.1 Experimental Setup

Benchmarks and Metrics. We evaluate FlowRouter on four widely used benchmarks spanning mathematical reasoning, knowledge-intensive reasoning, and code generation. **GSM8K** (Cobbe et al., 2021) targets grade-school mathematical reasoning, while **MATH** (Hendrycks et al., 2021) covers more challenging competition-level mathematics. **MMLU** (Hendrycks et al., 2020) assesses knowledge and reasoning across diverse academic subjects, and **MBPP** (Austin et al., 2021) evaluates natural-language-to-code generation with executable test cases. We report accuracy on GSM8K,

MATH, and MMLU, and pass@1 on MBPP. We additionally report inference cost to assess the performance–cost trade-off.

Baselines. We compare FLOWROUTER with representative methods spanning single-agent reasoning, multi-agent collaboration, automated workflow optimization, and LLM routing. Single-agent baselines include Vanilla inference, Chain-of-Thought (CoT) (Wei et al., 2022), and ComplexCoT (Fu et al., 2023). MacNet (Qian et al., 2025) serves as a representative fixed multi-agent collaboration framework, while AFlow (Zhang et al., 2025b) represents automated workflow optimization. For routing-based methods, we include RouteLLM (Ong et al., 2025), RouterDC (Chen et al., 2024b), and MasRouter (Yue et al., 2025). We additionally evaluate Vanilla inference with each candidate backbone to provide reference performance for the underlying LLM pool.

LLM Backbones. The candidate LLM pool consists of four heterogeneous backbones: `nemotron-3-ultra` (NVIDIA, 2026b), `nemotron-3.5-lightning` (NVIDIA, 2026a), `gpt-oss-120b` (OpenAI, 2025), and `deepseek-v4-flash` (DeepSeek-AI, 2026). We evaluate routing-based methods over this common model pool. For homogeneous baselines, we generally use `nemotron-3-ultra` as the shared backbone.

Implementation Details. We set the LLM generation temperature to 0.2, the cost penalty to $\lambda_c \in \{10, 30, 50\}$, the number of iterations to $E \in \{3, 5, 10\}$, and the number of agents to $k_{\min} = 1$ and $k_{\max} = 6$. For textual inputs that require encoding, we use a frozen sentence encoder [1], and the LLM backbones considered in our experiments constitute the candidate routing pool. The memory module retrieves the top-3 training experiences for each query, while test instances are never added to memory. Further implementation details are provided in Appendix C.

Table 1: Main results on four benchmarks. "Mul." denotes multi-agent collaboration, and "Rout." denotes routing. All results are reported in accuracy (%), except MBPP, which is reported in pass@1 (%). The best and second-best results are shown in **bold** and underline, respectively.

Method	LLM	Mul.	Rout.	GSM8K	MMLU	MATH	MBPP	Avg.
Vanilla	nemotron-3-ultra	×	×	88.14	90.64	81.67	80.33	85.20
	gpt-oss-120b	×	×	81.67	86.55	83.00	89.48	85.18
	nemotron-3.5-lightning	×	×	86.91	85.67	76.33	76.67	81.39
	deepseek-v4-flash	×	×	69.90	88.50	69.67	82.00	77.52
CoT (Wei et al., 2022)	nemotron-3-ultra	×	×	83.16	90.11	76.33	76.33	81.48
ComplexCoT (Fu et al., 2023)	nemotron-3-ultra	×	×	75.26	86.67	76.33	76.00	78.57
AFlow (Zhang et al., 2025b)	nemotron-3-ultra	✓	×	<u>96.93</u>	89.38	77.70	45.67	77.42
MacNet (Qian et al., 2025)	nemotron-3-ultra	✓	×	92.09	**91.58**	87.24	<u>89.67</u>	90.15
RouteLLM (Ong et al., 2025)	LLM Pool	×	✓	94.00	85.00	88.00	87.00	88.50
RouterDC (Chen et al., 2024b)	LLM Pool	×	✓	94.50	84.33	88.33	84.33	87.87
MasRouter (Yue et al., 2025)	LLM Pool	✓	✓	96.52	89.87	<u>89.70</u>	87.33	<u>90.86</u>
FlowRouter	**LLM Pool**	✓	✓	**97.98**	<u>91.15</u>	**90.64**	**92.00**	**92.94**

4.2 PERFORMANCE AND COST ANALYSIS

High-performance. The experimental results in Table 1 demonstrate that FLOWROUTER is effective at constructing query-specific multi-agent workflows. Specifically, FLOWROUTER achieves the

[1] Specifically, we use `all-mpnet-base-v2` (Reimers & Gurevych, 2019).

highest average performance of 92.94% across the four benchmarks. It exceeds the strongest Vanilla backbone by 7.74 percentage points on average and also outperforms CoT and ComplexCoT, showing clear gains over single-agent reasoning. Compared with multi-agent and automated workflow baselines, FLOWROUTER improves over MacNet by 2.79 points on average while also outperforming AFlow. It further surpasses the routing-based methods RouteLLM, RouterDC, and MasRouter by 4.44, 5.07, and 2.08 points, respectively. These results indicate that the fine-grained workflow adaptation enabled by FLOWROUTER contributes to effective multi-agent orchestration.

Cost-economical. Figure 2 shows that FlowRouter achieves the highest performance on GSM8K while remaining on the empirical Pareto frontier. Compared with AFlow and MacNet, FlowRouter improves performance by 1.05 and 5.89 percentage points while reducing the average inference cost by 84.07% and 97.10%, respectively. Results on the remaining benchmarks exhibit similar non-dominated trade-offs and are reported in Appendix D. These results demonstrate that FLOWROUTER is a cost-efficient framework for query-specific multi-agent orchestration, achieving strong performance without relying on uniformly expensive collaboration.

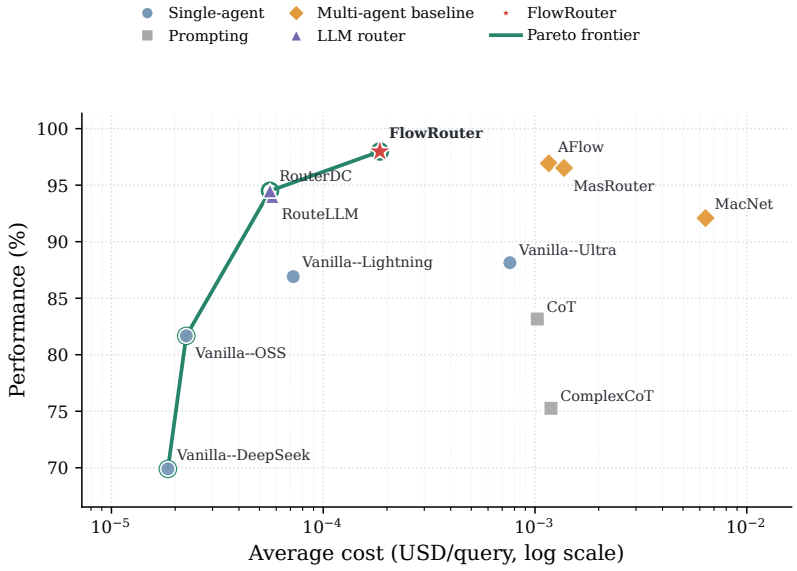


Figure 2: Performance versus average inference cost on GSM8K. The green curve denotes the empirical Pareto frontier.

Training resource efficiency.

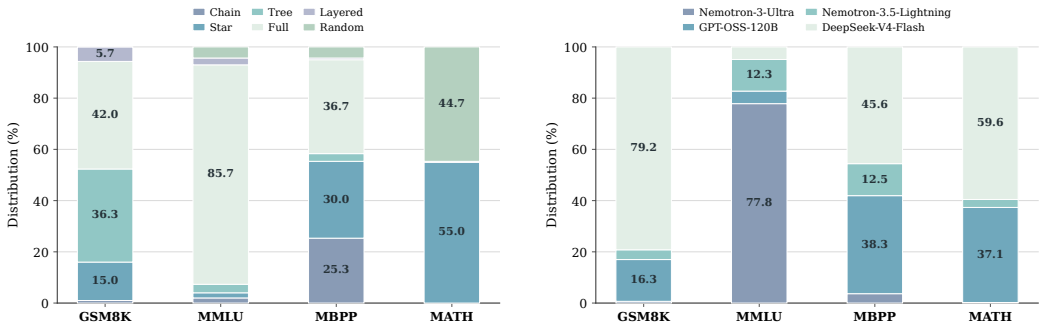
The results in Table 2 demonstrate that FLOWROUTER achieves strong training resource efficiency, as measured by the training API cost normalized by the total number of example evaluations across all optimization epochs. FLOWROUTER achieves the lowest normalized training cost on all four benchmarks, reducing API expenditure by 30.15% relative to AFlow and 38.46% relative to MasRouter. This efficiency may partly reflect reduced reliance on exhaustive exploration and repeated validation of candidate workflow structures during training.

Table 2: Training API cost per example evaluation ($\times 10^{-3}$ USD). "Avg." denotes the macro-average across benchmarks.

Method	GSM8K	MMLU	MATH	MBPP	Avg.
AFlow	1.365	1.648	3.091	3.911	2.504
MasRouter	1.477	2.177	5.600	2.112	2.842
FlowRouter	0.852	1.440	2.681	2.023	1.749

4.3 ROUTING BEHAVIOR

Figure 3 summarizes the test-time routing behavior of FLOWROUTER across benchmarks. As shown in Figure 3a, topology preferences vary substantially across tasks. GSM8K and MATH exhibit distinct topology distributions despite both being mathematical reasoning benchmarks, suggesting



(a) Topology selection.

(b) LLM assignment.

Figure 3: Test-time routing behavior across benchmarks.

that routing is not determined solely by coarse task categories. Within MATH, topology selection is significantly associated with mathematical subfield ($V_{\text{Cramér}} = 0.320$, permutation $p < 10^{-4}$; see Appendix E.2), but shows no significant association with the official difficulty levels ($p = 0.393$). This contrast suggests that the learned routing behavior is more closely associated with differences in problem structure and reasoning type than with a coarse scalar difficulty annotation.

Figure 3b further shows that model usage differs markedly across benchmarks. `Nemotron-3-ultra` accounts for 77.85% of routed agent positions on MMLU, but fewer than 4% on the remaining benchmarks, where lower-cost models are selected more frequently. This contrast indicates that the learned policy does not rely on a uniform model composition across tasks, but adapts the use of heterogeneous backbones to different benchmark distributions. Combined with the topology results, these observations reveal systematic variation in both collaboration structure and model composition across benchmarks.

4.4 Framework Analysis

Ablation Study. We assess the contribution of different components in FlowRouter by replacing query-specific topology routing with fixed Chain, Tree, or Random structures, fixing the collaboration scale to three agents, using a homogeneous LLM assignment, shuffling routed LLMs across workflow positions, and removing experience-based memory guidance, while keeping the remaining components unchanged.

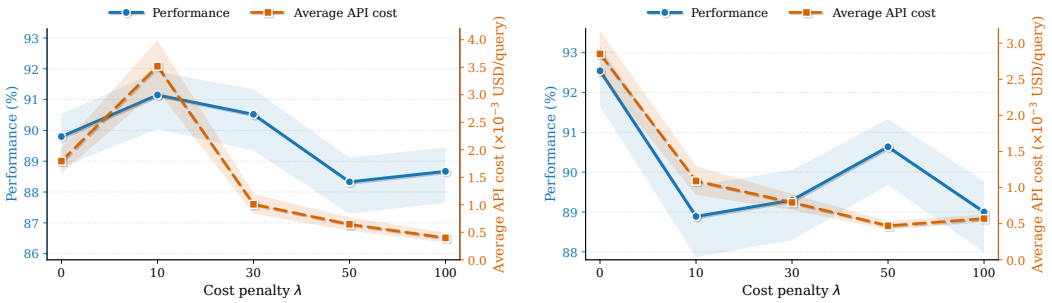
Table 3: Ablation study of FlowRouter. Performance is in percentages and cost in USD per query.

Variant	MATH		MBPP	
	Perf. ↑	Cost ↓	Perf. ↑	Cost ↓
FlowRouter	90.64	0.000471	92.00	0.000719
w/ Chain	89.00	0.000598	85.33	0.000723
w/ Tree	90.00	0.000506	84.33	0.000804
w/ Random	90.24	0.000428	89.33	0.000638
w/ Fixed Scale	87.25	0.000824	87.06	0.000705
w/ Fixed LLM Assignment	84.69	0.009872	83.00	0.009681
w/ Shuffled LLM Assignment	90.27	0.000535	90.00	0.000685
w/o Memory	88.29	0.000602	90.67	0.001074

Table 3 shows that topology, collaboration scale, and model allocation all contribute to the resulting performance–cost trade-off. Replacing topology routing with a fixed structure consistently reduces performance. Chain and Tree also incur higher inference cost than FlowRouter, whereas Random is cheaper but less accurate. None of the evaluated fixed topologies therefore matches FlowRouter across both metrics, supporting query-specific adaptation of the collaboration struc-

ture. Fixing the collaboration scale also reduces performance on both benchmarks, by 3.39 points on MATH and 4.94 points on MBPP, while its effect on cost differs across tasks. This suggests that scale adaptation is useful for adjusting collaborative computation to individual queries rather than uniformly reducing inference cost. A larger degradation occurs with fixed LLM assignment. Assigning `nemotron-3-ultra` to all workflow positions reduces performance by 5.95 and 9.00 points on MATH and MBPP, respectively. Shuffling the routed LLMs across positions, while preserving workflow structure, scale, and model composition, also leads to lower performance. These results suggest that uniformly using a stronger and more expensive backbone does not recover the gains of adaptive model allocation, whose effectiveness may also be sensitive to position-wise placement. Taken together, the topology, scale, and model ablations provide empirical support for jointly adapting the three dimensions of workflow construction. Removing memory guidance further degrades both effectiveness and efficiency: performance decreases by 2.35 and 1.33 points on MATH and MBPP, respectively, while inference cost increases by 27.81% and 49.37%. This result is consistent with retrieved execution experience providing useful auxiliary guidance for avoiding less effective or unnecessarily costly workflow configurations.

Sensitivity Analysis. We analyze the sensitivity of FlowRouter to the cost penalty λ on MMLU and MATH. As shown in Figure 4, introducing a moderate cost penalty is associated with substantially lower inference cost and relatively small performance changes. Compared with $\lambda = 0$, setting $\lambda = 50$ reduces the average cost by 64.07% on MMLU and 83.47% on MATH, while retaining 98.36% and 97.95% of the corresponding performance, respectively. This pattern suggests that λ can discourage unnecessarily expensive workflows and provide a practical mechanism for balancing performance against inference cost.



(a) MMLU (b) MATH

Figure 4: Sensitivity of performance and average inference cost to the cost penalty λ.

4.5 Case Study

We visualize representative workflows produced by FlowRouter to show how routing decisions translate into concrete multi-agent configurations. The examples vary in topology, scale, and model composition, illustrating query-level differences in information flow and model usage. Additional examples are provided in Appendix F.

5 Conclusion

This work studies query-adaptive multi-agent collaboration through workflow routing for effective and efficient orchestration. We view a multi-agent workflow as a structured configuration jointly defined by collaboration topology, scale, and position-wise LLM assignment. Building on this perspective, we introduced FlowRouter, an experience-guided framework that factorizes workflow construction along these dimensions and optimizes the routing decisions under a shared performance–cost objective. Experiments show that FlowRouter adaptively configures collaboration structure, scale, and model allocation for different queries, while achieving strong performance with favorable inference cost. Ablation results further confirm the contribution of each routed dimension to the overall workflow. These findings suggest that increasing the granularity of adaptation may be important for more flexible and deployment-efficient multi-agent orchestration. Our evaluation mainly considers benchmark performance and API cost, leaving latency, concurrency, and execution reliability less explored. Extending query-adaptive routing to broader deployment settings remains an important direction toward more general and resource-aware multi-agent systems.

AI USE STATEMENT

Generative AI tools were used for language editing and limited coding assistance, including improvements to clarity, grammar, readability, and implementation efficiency. The research ideas, methodology, experimental design, analysis, interpretation, and scientific content were developed by the authors. All AI-assisted text and code were reviewed, tested, and revised as necessary by the authors, who take full responsibility for the final manuscript and implementation.

REFERENCES

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.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856.

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, 2024b.

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.

DeepSeek-AI. Deepseek-v4-flash. `https://api-docs.deepseek.com/updates/`, 2026.

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*, 2024.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2024.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 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.

Junjie Li, Xi Xiao, Yunbei Zhang, Chen Liu, Lin Zhao, Xiaoying Liao, Yingrui Ji, Janet Wang, Jianyang Gu, Yingqiang Ge, et al. Agent harness engineering: A survey. *OpenReview preprint*, 2026.

NVIDIA. Nvidia nemotron-3.5 lightning 30b a3b. `https://build.nvidia.com/nvidia/nemotron-3.5-lightning-30b-a3b/modelcard`, 2026a.

NVIDIA. Nvidia nemotron-3 ultra 550b a55b. https://build.nvidia.com/nvidia/nemotron-3-ultra-550b-a55b/modelcard, 2026b.

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*, 2025.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. https://openai.com/index/gpt-oss-model-card/, 2025.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, volume 2025, pp. 41488–41505, 2025.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space. In *International Conference on Learning Representations*, volume 2025, pp. 3841–3865, 2025.

Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, and Lewei He. Difficulty-aware agentic orchestration for query-specific multi-agent workflows. In *Proceedings of the ACM Web Conference 2026*, pp. 2060–2070, 2026.

Qian Wang, Zhenheng Tang, ZICHEN JIANG, Nuo Chen, Tianyu Wang, and Bingsheng He. Agent-taxo: Dissecting and benchmarking token distribution of LLM multi-agent systems. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6192–6217, 2025.

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.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, LEI BAI, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Forty-second International Conference on Machine Learning*, 2025a.

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, 2025b.

Qinlin Zhao, Jindong Wang, Yixuan Zhang, Yiqiao Jin, Kaijie Zhu, Hao Chen, and Xing Xie. CompeteAI: Understanding the competition dynamics of large language model-based agents. In *Forty-first International Conference on Machine Learning*, 2024.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024.

A Experience-Guidance Computation

For query Q, let $\mathcal{R}_Q$ denote the retrieved training experiences. For each retrieved experience $\xi_j \in \mathcal{R}_Q$, s_j denotes its semantic similarity to Q, and w_j its signed experience weight. Let $r \in \{T, k, (m, i)\}$ index topology routing, scale routing, or model routing at position i, respectively, and let $\pi_r(\xi_j)$ denote the corresponding historical action. Here, T, k, and M denote a candidate topology, scale, and model, respectively.

For candidate action a, the aggregated experience bias is

$$B_a^{(r)}(\mathcal{R}_Q) = \frac{\beta}{\sum_{\xi_j \in \mathcal{R}_Q^{(r)}} s_j + \epsilon} \sum_{\xi_j \in \mathcal{R}_Q^{(r)}} w_j \, \mathbb{I}[\pi_r(\xi_j) = a] \,, \tag{9}$$

where $\mathcal{R}_Q^{(r)} \subseteq \mathcal{R}_Q$ contains experiences valid for routing stage r, β controls the evidence strength, $\epsilon > 0$ ensures numerical stability, and $\mathbb{I}[\cdot]$ is the indicator function.

The corresponding multiplicative guidance factor is

$$G_a^{(r)}(\mathcal{R}_Q) = \exp\left(\eta_r B_a^{(r)}(\mathcal{R}_Q)\right), \tag{10}$$

where η_r controls the guidance strength for stage r. We use $\eta_T = \eta_{(m,i)} = 1/\tau$ and $\eta_k = 1$, where τ is the routing temperature. This yields $G_T(\mathcal{R}_Q)$, $G_k(\mathcal{R}_Q)$, and $G_{i,M}(\mathcal{R}_Q)$ for topology, scale, and position-wise LLM routing, respectively. When no relevant experience is retrieved, $G_a^{(r)}(\mathcal{R}_Q) = 1$.

B Training Algorithm

Algorithm 1 summarizes the end-to-end training procedure of FlowRouter. For each training query, the controller retrieves relevant experience, progressively constructs a workflow, executes it, and updates the routing policy using the resulting performance–cost reward. The completed execution is then stored in memory to guide subsequent routing decisions.

Algorithm 1 Training Procedure of FlowRouter

Require: Training set $\mathcal{D}$, topology set $\mathcal{T}$, LLM pool $\mathcal{L}$, the number of iterations E, experience memory $\mathcal{M}$, trainable parameters Θ
Ensure: Optimized routing policy F_Θ

for $e = 1$ **to** E **do**
  for each $(d_Q, Q, a) \in \mathcal{D}$ **do**
    Retrieve experience: $\mathcal{R}_Q \leftarrow \mathrm{Retrieve}(Q, \mathcal{M})$
    Route topology: $T_Q \sim F_{\theta_t}(\cdot \mid Q, \mathcal{R}_Q)$
    Estimate scale: compute reference scale $\bar{k}_Q$
    Route scale: $k_Q \sim F_{\theta_k}(\cdot \mid Q, T_Q, \mathcal{R}_Q)$
    Instantiate workflow: $\mathcal{G}_Q \leftarrow \Gamma(T_Q, k_Q)$
    for $i = 1$ **to** k_Q **do**
      Assign model: $M_{Q,i} \sim F_{\theta_m}(\cdot \mid Q, T_Q, k_Q, i, \mathcal{R}_Q)$
    end for
    Construct configuration: $\mathcal{S}_Q \leftarrow (T_Q, k_Q, \{M_{Q,i}\}_{i=1}^{k_Q})$
    Execute workflow: $\hat{a} \leftarrow \mathcal{O}(Q, \mathcal{S}_Q)$
    Evaluate outcome: $y_Q \leftarrow \mathcal{U}(\hat{a}, a), \quad c_Q \leftarrow \mathcal{C}(Q, \mathcal{S}_Q)$
    Compute reward: $R(Q, \mathcal{S}_Q) \leftarrow y_Q - \lambda_c c_Q$
    Update policy: update Θ using Eq. 8
    Record experience: $\xi_Q \leftarrow (d_Q, Q, \mathcal{S}_Q, \mathcal{G}_Q, y_Q, c_Q)$
    Update memory: $\mathcal{M} \leftarrow \mathcal{M} \cup \{\xi_Q\}$
  end for
end for
return F_Θ

C Baseline Implementation Details

For reproducibility, we provide additional implementation details for the evaluated baselines. We generally use `nemotron-3-ultra` as the shared backbone and `all-mpnet-base-v2` as the sentence encoder when textual representations are required.

AFlow. We implement AFlow as a declarative workflow search over task-specific operator libraries, with at most 10 optimization rounds, four parent workflows per round, and early stopping after three unchanged rounds. Parent workflows are sampled using the prescribed 0.3 uniform and 0.7 score-softmax mixture.

MacNet. We implement MacNet as a fixed seed-1 connected random DAG with four actor nodes, edge-specific critics, artifact-only message propagation, hierarchical aggregation with unit size two, and an additional sink for multiple-output graphs. For MBPP and MATH, we use four graph-node calls followed by one final aggregation call.

RouteLLM. We retrain the RouteLLM TextMF router for 100 epochs using cached model responses from the training set, with its routing endpoints drawn from the common model pool used throughout our routing-based evaluation.

RouterDC. We retrain RouterDC over our four-model pool because the released checkpoint targets a different 11-model pool. The dual-contrastive objectives are optimized for 100 steps with batch size 16 and learning rate 5×10^{-5}.

MasRouter. We retain the released MasRouter controllers and `MAR.Graph.Graph` executor, allowing up to six agents, six collaboration modes, and the same four-model pool as FlowRouter. The router is trained for five epochs with batch size 16 and learning rate 10^{-2} using task-classification, cost-aware policy-gradient, and VAE regularization objectives.

D Additional Performance–Cost Results

Figures 5–7 show the performance–cost trade-offs on MMLU, MATH, and MBPP. FlowRouter achieves comparable performance to MacNet on MMLU with substantially lower inference cost, while obtaining higher performance and lower cost on both MATH and MBPP. These results further demonstrate a favorable performance–cost trade-off across the three benchmarks.

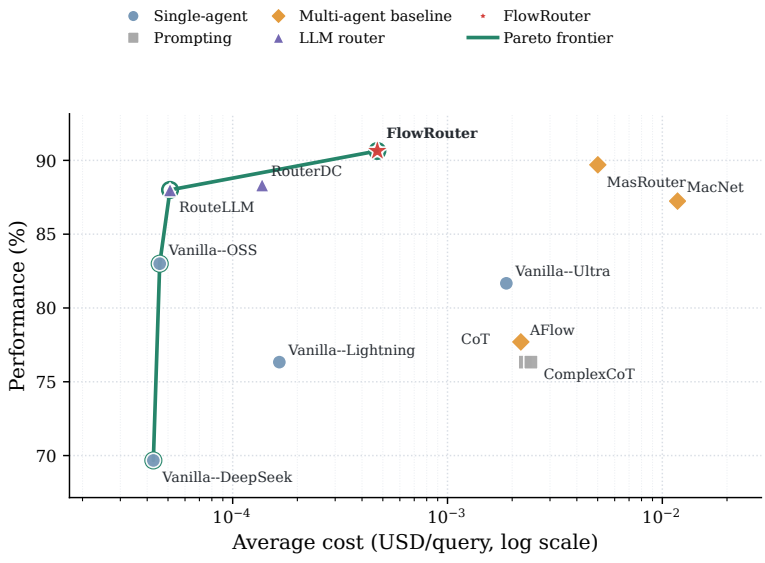


Figure 5: Performance versus average inference cost on MATH. Green curves denote the empirical Pareto frontier.

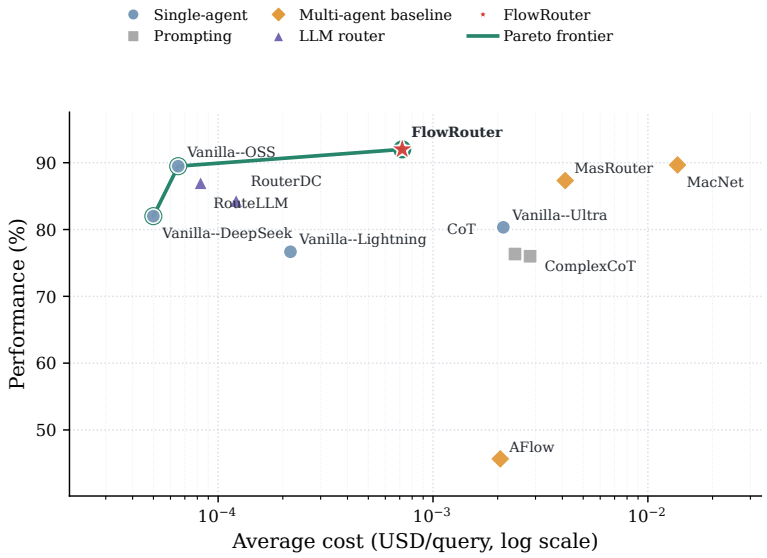


Figure 6: Performance versus average inference cost on MBPP. Green curves denote the empirical Pareto frontier.

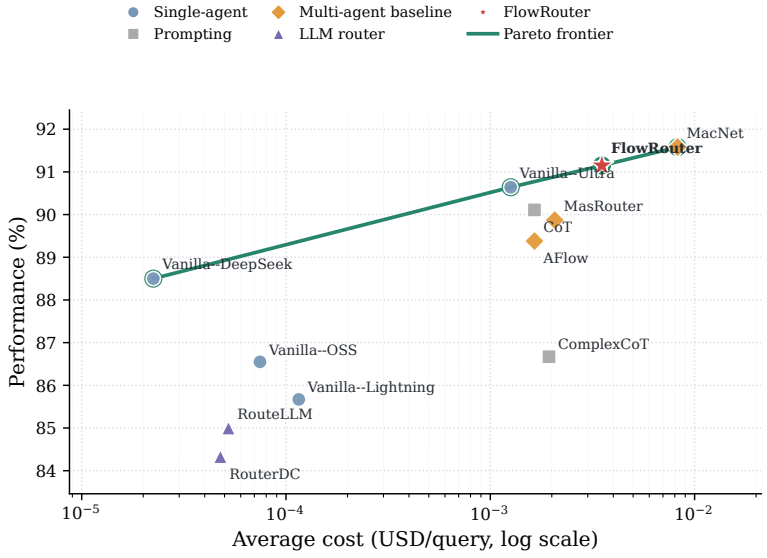


Figure 7: Performance versus average inference cost on MMLU. Green curves denote the empirical Pareto frontier.

E Additional Routing Analysis

E.1 Collaboration Scale Across Benchmarks

To further examine how FlowRouter allocates collaborative computation, we analyze the distribution of routed agent counts across benchmarks.

As shown in Figure 8, the routed collaboration scale varies across benchmarks. Three-agent workflows are the most frequently selected configuration across all four tasks, accounting for roughly

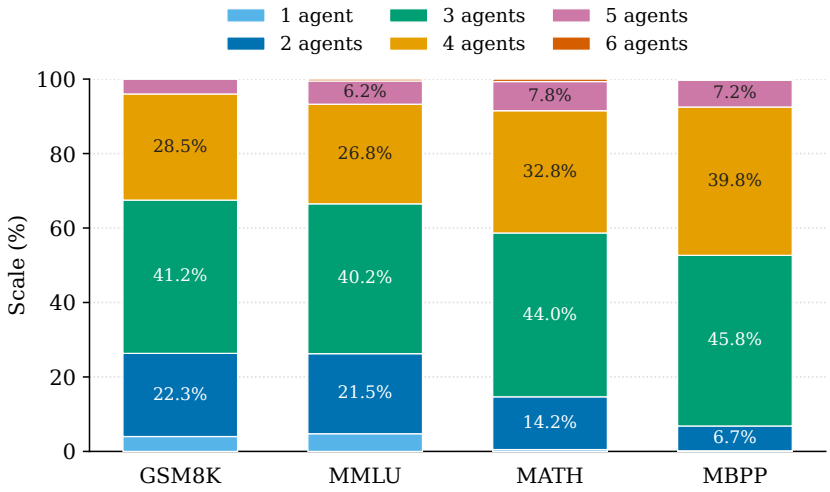


Figure 8: Distribution of collaboration scale across benchmarks.

40%–46% of queries. GSM8K and MMLU select two-agent workflows more often, whereas MATH and MBPP show a stronger preference for four-agent configurations, reaching 39.8% on MBPP. Larger workflows remain comparatively rare, indicating that FLOWROUTER does not simply increase the number of agents uniformly. Instead, the learned policy adjusts collaboration scale across tasks, providing additional evidence for adaptive resource allocation in query-specific workflow construction.

E.2 WITHIN-BENCHMARK CONDITIONAL ROUTING

To examine whether the observed routing behavior reflects more than benchmark-level specialization, we further analyze topology selection within MATH by grouping test queries according to their annotated mathematical subfields.

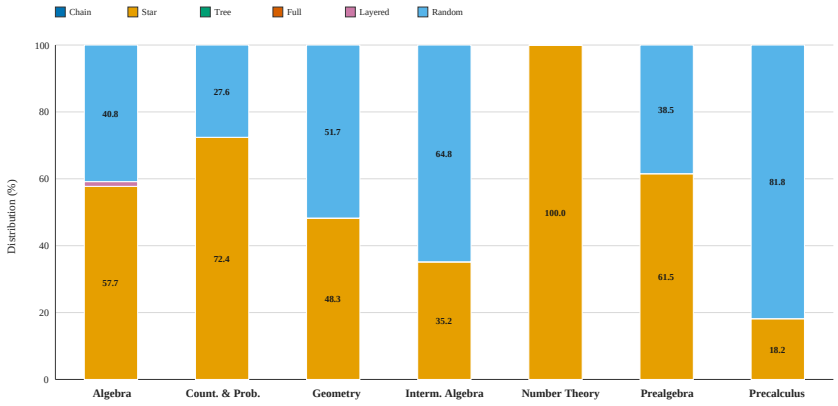


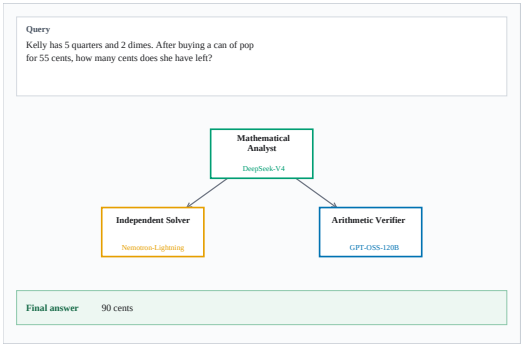
Figure 9: Topology selection across mathematical subfields in MATH. Topology distributions differ across subfields, with a statistically significant association between subfield and selected topology ($V_{\text{Cramér}} = 0.320$, permutation $p < 10^{-4}$).

As shown in Figure 9, topology preferences vary considerably across mathematical subfields. For example, Number Theory queries are routed exclusively to `star`, whereas `random` is selected for 81.8% of Precalculus queries and 64.8% of Intermediate Algebra queries. The association between mathematical subfield and topology selection is statistically significant ($V_{\text{Cramér}} = 0.320$, permutation $p < 10^{-4}$). Topology selection is not significantly associated with the official MATH difficulty

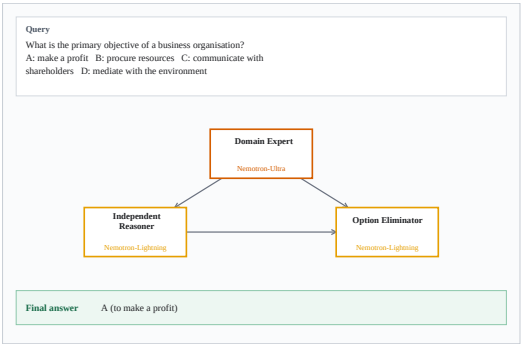
level ($p = 0.393$), suggesting that the learned routing behavior reflects characteristics beyond scalar difficulty. Together with the scale distributions above, these results indicate that query adaptivity in FLOWROUTER involves both how much collaborative computation is allocated and how that computation is structurally organized.

F CASE STUDIES

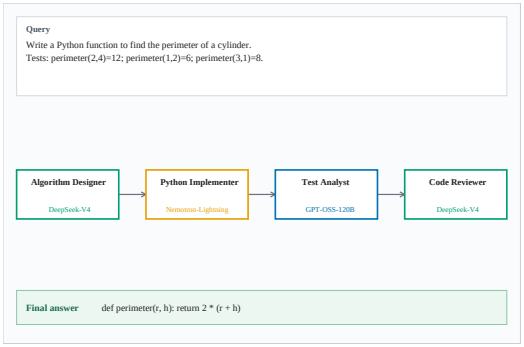
We present representative query-specific workflows from test instances to illustrate how FLOWROUTER adapts collaboration topology, scale, position-wise LLM assignment, and directed information flow across queries. Figure 10 includes examples from all four benchmarks, showing how different routing decisions lead to distinct multi-agent configurations. Figure 11 provides additional MBPP examples with different collaboration topologies to further demonstrate the diversity of routed workflows.



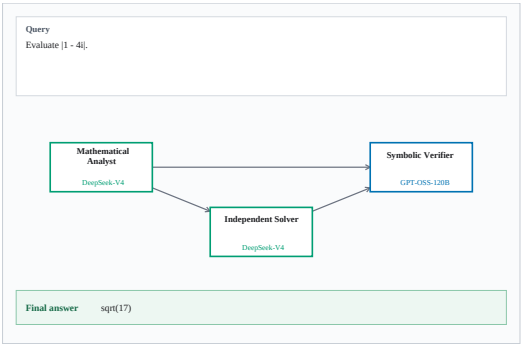
(a) GSM8K (`Tree`).



(b) MMLU (`Full`).

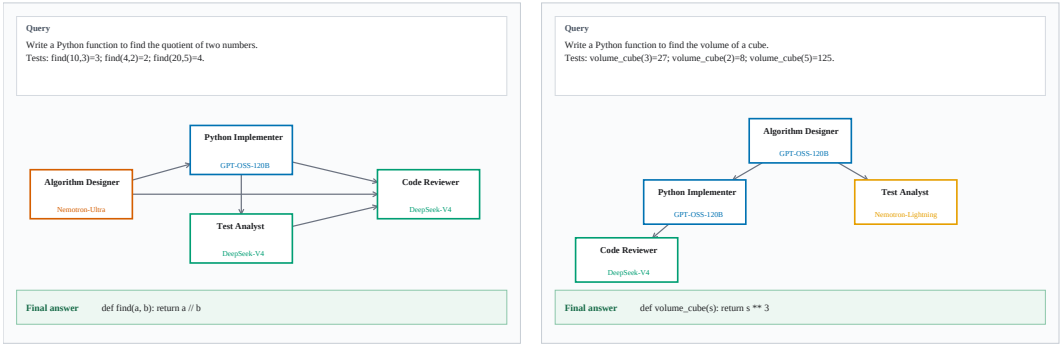


(c) MBPP (`Chain`).

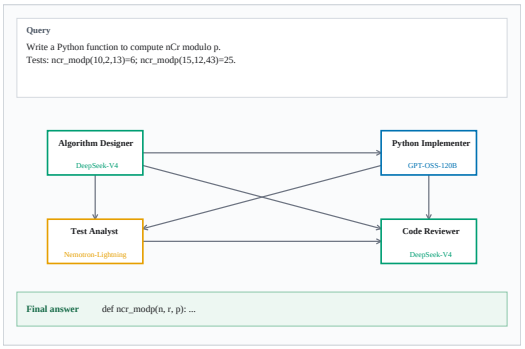


(d) MATH (`Random`).

Figure 10: Representative query-specific workflows routed by FLOWROUTER across GSM8K, MMLU, MBPP, and MATH. The examples illustrate variation in collaboration topology, collaboration scale, and position-wise LLM assignment.



(a) `Random`. (b) `Tree`.



(c) `Full`.

Figure 11: Representative workflows routed by FLOWROUTER under different collaboration topologies on MBPP.