ProgRouter: Online Progress-Guided Routing for Efficient Collaborative LLM Agentic Workflows

Songyuan Li
S.Y.Li@qmul.ac.uk
Queen Mary University of London

Ahmed M. Abdelmoniem
ahmed.sayed@qmul.ac.uk
Queen Mary University of London

Shiqiang Wang
S.Wang9@exeter.ac.uk
University of Exeter

1 Introduction

We have been witnessing the large language models (LLMs) [3][4] that demonstrate remarkable task reasoning and problem-solving capabilities. These advances have catalyzed a new paradigm of AI systems known as *collaborative agentic workflows* [5][2]. Multiple specialized LLM-powered AI agents are orchestrated to tackle a complex, open-ended task through decomposing it into sub-tasks and resolving each through multiple steps. Each step entails an agent's LLM inference call, and iterative agent steps substantially increase cumulative response time and operational cost (e.g., energy consumption) compared with a single-turn LLM response. Consequently, managing the trade-off between the workflow's solution quality and task-solving efficiency (e.g., time and operational cost) has been a critical challenge in the deployment of practical collaborative LLM-agent workflows.

Now, we ask the fundamental question: *How can we design an algorithm for dynamically routing LLM agents across steps to preserve solution quality under time and operational-cost budgets?* However, there is no silver-bullet solution to this problem. Routing more powerful LLM agents yield better task-solving performance, but incur substantial operational cost and time overhead. In contrast, lightweight LLM agents are faster and more cost-effective yet less capable when handling difficult sub-tasks, thus weakening the overall workflow outcome. Rather, effective LLM agent routing should be dynamic and online adaptive. Subject to operational time and cost constraints, the routing policy should make online decisions based on the workflow progress and the specific sub-task (e.g., task difficulty) being addressed by the current agent step. Therefore, we propose ProgRouter, an online routing framework that adapts LLM-agent selection to the evolving progress of collaborative agentic workflows.

2 Methodology

Workflow Progress Evaluation. We define a task progress score $\phi(\mathbf{s}) \in [0, 1]$ over the workflow state $\mathbf{s}$ to measure how close the current workflow is to a successful task outcome. The score captures progression from no viable artifact, to partially correct outputs, and finally to a fully verified solution. ProgRouter evaluates $\phi(\mathbf{s})$ across consecutive agent steps and uses it as the feedback signal for routing decisions.

Smooth progress estimation is essential for reliable routing. Sharp progress-score fluctuations can cause ProgRouter to misinterpret the workflow state: overestimated workflow progress may lead to premature selection of lightweight LLM agents, while underestimated progress may trigger unnecessary escalation to stronger and more costly LLM agents. To stabilize the routing signal, we introduce a progress adjustment term ε that captures progress trends across consecutive workflow steps, including improvements in sub-task completion, similarity between intermediate artifacts, and unresolved failures. In this way, ε smooths the evolution of $\phi(\mathbf{s})$ over the workflow and supports more reliable LLM agent routing decisions.

Task Progress Predictor. ProgRouter forecasts how much task progress each candidate LLM agent would contribute before invoking it. Formally, we learn a predictor $\hat{f}(\mathbf{s}_k, m)$ that estimates the one-step progress improvement $\hat{q}_k(m) = \mathbb{E}[\phi(\mathbf{s}_{k+1}) - \phi(\mathbf{s}_k) \mid \mathbf{s}_k, m]$ for selecting LLM agent m given the current workflow state $\mathbf{s}_k$.

To build a lightweight yet effective predictor, we address two points. *First,* the workflow state contains heterogeneous information, including tabular state features (e.g., sub-task stagnation counters, failure modes, and LLM agent routing history) and unstructured semantic features (e.g., the overall task description and current sub-task description). We therefore combine tabular base-learners with sentence-embedding base-learners and fuse their predictions through a meta-learner, enabling the predictor to jointly capture structured workflow dynamics and rich semantic task context. *Second,* the predictor should learn transferable task progress prediction knowledge rather than memorizing task-specific patterns. We therefore train and evaluate the predictor using offline agentic workflow trajectories with leave-one-task-out cross-validation.

Online LLM Agent Routing. At each agent step, ProgRouter selects the most appropriate LLM agent $m_k^* = \arg\max_{m \in \mathcal{M}} \text{Score}(m, \mathbf{s}_k)$ online according to a weighted LLM agent routing score:

$$\text{Score}(m, \mathbf{s}_k) = (1 - \phi(\mathbf{s}_k)) \cdot \hat{q}_k(m) - \alpha_t \cdot \frac{t_m}{T_{\text{bgt}}} \; - \; \alpha_c \cdot \frac{c_m}{C_{\text{bgt}}},$$

where $\mathcal{M}$ denotes the set of candidate LLM agents, t_m and e_m denote the per-step response time and operational cost of LLM agent m, and T_{bgt} and E_{bgt} denote the workflow's time and operational-cost budgets, respectively. The first term $[\hat{q}_k(m) \cdot (1 - \phi(\mathbf{s}_k))]$ defines the quality objective. It rewards LLM agents that are expected to make meaningful task progress, while avoiding both underpowered agents that contribute little progress (i.e., $\hat{q}_k(m) \to 0$) and unnecessarily strong agents for workflow steps that are already close to completion (i.e., $\phi(\mathbf{s}_k) \to 1$). This steers workflow routing toward the cost-appropriate LLM agent for the current task progress. The second and third terms penalize LLM agents whose incurred response time and operational cost represent a large fraction of the time and cost budgets T_{bgt} and C_{bgt}, with α_t and α_c weighing each constraint. These three terms encode a routing objective that is quality-driven when task progression is lacking, efficiency-aware when budgets are tight, and naturally self-balancing across the workflow.

3 Results

We evaluate ProgRouter on *HumanEval* [1], a popular code-generation benchmark (totally 164 tasks) for evaluating LLM agents. We prototype a collaborative agentic workflow with seven specialized worker agent roles (planner, implementer, reviewer, tester, debugger, finisher). The LLM agent zoo $\mathcal{M}$ consists of four Qwen2.5-Coder variants at 0.5B, 7B, 14B, and 32B parameters.

Method	Efficiency		Quality	Routing Distribution (%)			
	Time (s)	Energy (J)	Pass (%)	0.5B	7B	14B	32B
ProgRouter	3.51	1,604	98.8	45.6	43.8	8.8	1.8
Fixed 0.5B	5.63	1,239	8.5	100	—	—	—
Fixed 7B	3.86	1,697	90.2	—	100	—	—
Fixed 14B	6.40	3,158	92.7	—	—	100	—
Fixed 32B	18.70	10,074	98.2	—	—	—	100
Random	9.45	4,802	94.5	44.3	7.5	0.0	48.3

ProgRouter achieves the best quality-efficiency trade-off, attaining the highest task-solving pass rate of 98.8% while using far less time and energy than the Fixed 32B baseline. These gains stem from selectively routing more powerful LLM agents only when needed, rather than using them at all workflow steps. Our results show that *when* a right LLM agent is routed is more important than simply invoking stronger LLM agents throughout.

References

[1] Mark Chen, Jerry Tworek, et al. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG] https://arxiv.org/abs/2107.03374
[2] CrewAI. 2024. The Leading Multi-Agent Platform. https://crewai.com/.
[3] Daya Guo, Dejian Yang, et al. 2025. Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Nature* 645 (2025), 633–638.
[4] Qwen Team. 2026. Qwen-Scope: Turning Sparse Features into Development Tools for Large Language Models. (2026). https://huggingface.co/collections/Qwen/qwen-scope Technical report released on 2026-04-30.
[5] Qingyun Wu, Gagan Bansal, et al. 2024. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversations. In *Annual Conference on Language Modeling (COLM)*. 1–46.