GRANULE-R1: SELF-ADAPTIVE GRANULARITY MANAGEMENT FOR AGENTIC QA VIA BOUNDARY-AWARE MULTI-LLM ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Current Large Language Model (LLM) routers predominantly operate at a coarse, query-level granularity or rely on static decomposition-then-allocation pipelines. These approaches often overlook the self-adaptive decision-making potential of the router itself—specifically, the ability to discern whether a task requires holistic reasoning or granular divide-and-conquer. We identify a critical opportunity to optimize this process by empowering the router to understand its own capability boundaries and those of a heterogeneous model pool. To address this, we present Granule-R1, an agentic reinforcement learning framework that redefines the router as a self-adaptive decision-making agent. Unlike prior methods that force a fixed decomposition structure, Granule-R1 autonomously evaluates the inherent difficulty of a query to decide between direct solving and strategic decomposition. By mastering the "capability-cost" trade-off, Granule-R1 identifies the most suitable model—not necessarily the strongest or cheapest—to address specific task granules. The system operates through a refined set of logic primitives—Think, Decompose, Search, Commit, and Answer—and is trained via Reinforcement Learning with a multi-objective reward that simultaneously optimizes for answer accuracy and inference efficiency. Evaluations on seven multi-hop and general QA benchmarks show that Granule-R1 significantly outperforms state-of-the-art methods. It achieves a 3.4-point absolute improvement in average Exact Match (EM) while reducing operational costs by 17.8%. Most importantly, Granule-R1 enables a system of 7B-tier models to surpass the performance of whole-query routing using 27B-tier models at less than half the cost. These findings demonstrate that router-level self-adaptivity and granularity management are critical dimensions for building the next generation of cost-effective, high-performance LLM systems.

1 INTRODUCTION

Large language models (LLMs) span a wide range of capabilities, costs, and latencies, motivating LLM routing as a means of dynamically selecting models to balance answer quality and inference cost (Chen et al., 2023; Ong et al., 2024; Hu et al., 2024; Feng et al., 2024). Yet across this growing body of work, the router itself remains a remarkably *passive* component. Whether implemented as a learned classifier over query embeddings (Ong et al., 2024; Hu et al., 2024), a graph-based predictor of query–model affinity (Feng et al., 2024), a multi-round agent that interleaves reasoning with external model calls (Zhang et al., 2025a), or a recent pipeline-style method that hands off a fixed decomposition to an independent allocator, current routers map a query to a model under a fixed assumption about *what* should be routed and *at what granularity*. The router is never asked to introspect on two prerequisite questions that arguably matter more than the model choice itself: **(i) How difficult is the current query, and does it warrant being broken down at all?** and **(ii) Where do the capability boundaries of the available models lie for this kind of subproblem?** Without these two competencies, prior decomposition work (Zhou et al., 2023; Khot et al., 2023; Press et al., 2023; Jin et al., 2025) treats the solving model as fixed and addresses only reasoning quality, while recent pipeline-style methods commit to a static plan generated by an isolated decomposer and rely on offline oracle search to label optimal allocations—the router never learns the two introspective competencies as part of a single coherent policy.

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.

We argue that the next generation of LLM routers must be *self-adaptive decision-making agents* with two learned competencies: **task-difficulty introspection**—judging whether a query should be answered holistically or strategically decomposed into an ordered set of subquestions—and **model-capability awareness**—reasoning about which model in a heterogeneous pool offers the best capability–cost match for a given subproblem, rather than defaulting to the strongest or the cheapest. Figure 1 illustrates the cost of this missing self-adaptivity through a representative compositional question: *"Which film has the director who was born earlier, Film A or Film B?"* A conventional single-query router observes only that the question requires multi-hop reasoning over two entities and a comparison, and—lacking introspection into the question's internal structure—commits the entire query to a single large model at high cost. Yet the difficulty here is largely structural: the question decomposes into locally simple subquestions about each film's director and birth date, each well within a 7B-tier model's reach. Once these subanswers return, the comparison between two dates is a trivial symbolic operation requiring no further LLM invocation; a self-adaptive router can absorb the returned facts back into its own reasoning context and produce the final answer itself. This *trajectory-aware integration*—where the router actively closes the loop rather than handing aggregation off to another model—is precisely what allows global question difficulty to be decoupled from local model requirements.

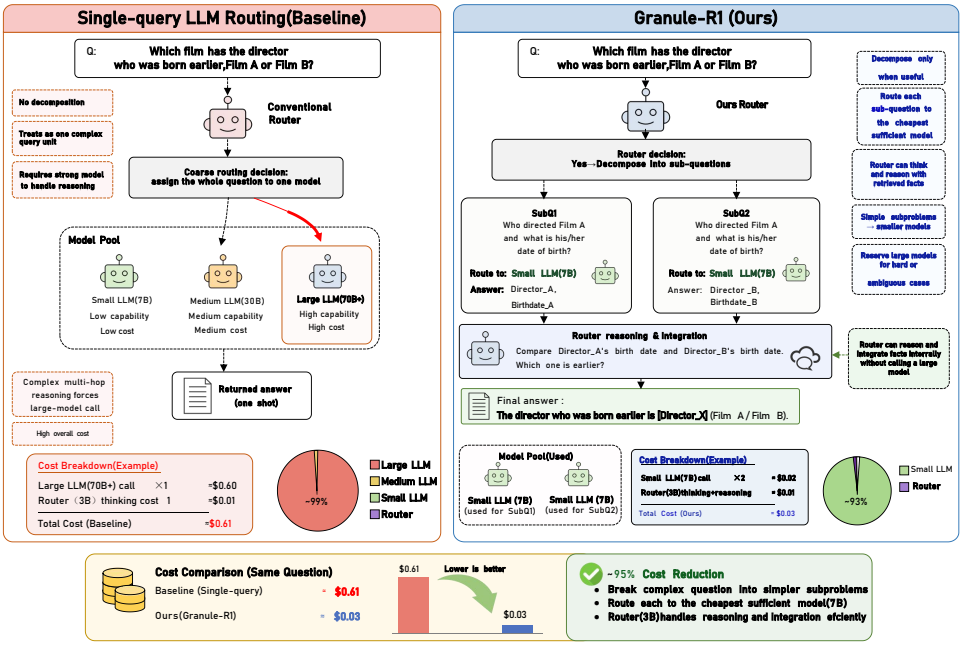

Figure 1: Conventional single-query routing (left) treats compositional questions as indivisible units and assigns them to a single expensive model. Granule-R1 (right) introspects on whether decomposition is beneficial, generates ordered subquestions, routes each to a cost-appropriate smaller model, and integrates the returned facts through the 3B router's own reasoning—answering complex questions by composing locally simple subproblems.

We propose **Granule-R1**, a self-adaptive LLM router trained with agentic reinforcement learning. Granule-R1 unifies decomposition decisions, model selection, and trajectory-aware integration into a *single policy* trained end-to-end without ground-truth decomposition labels or oracle allocation supervision. The policy operates through five logic primitives—**Think, Decompose, Search, Commit, and Answer**—interleaved within a multi-round interaction loop, allowing the router to refine its model selection and per-subquestion reasoning as intermediate evidence emerges within a single committed plan. By treating decomposition as a learned action rather than a preprocessing step, Granule-R1 shifts LLM routing from *query-level model selection* to *subquestion-level execution-plan construction*, deciding not only which model to call but at what problem granularity each call should be made. We train Granule-R1 with a multi-objective reward that simultaneously credits final-answer correctness, structural validity of the decomposition–aggregation trajectory, and per-subquestion efficiency that rewards successful resolution by the most capability-cost appropriate model, with

failure-aware adjustments that penalize wasted invocations and reward principled escalation when a smaller model proves insufficient. To externalize these learned capability assessments into a form that persists across episodes, we maintain a **capability-cost profile** that records, for each subquestion category, the top-k models with the strongest historical capability-cost performance and surfaces them to the router as in-context hints—turning accumulated routing experience into a queryable representation of model capability boundaries. These signals teach the router to internalize both query difficulty and model capability boundaries from interaction alone. Across general and multi-hop QA benchmarks, Granule-R1 outperforms strong single-round, cost-aware, and multi-round routing baselines on both accuracy and cost, and—most strikingly—enables a small-model-only execution stack to match the performance of large-model whole-query routing at substantially lower cost, empirically validating that router-level self-adaptivity is the decisive lever for cost-effective LLM coordination.

Our contributions are threefold:

- We identify the **lack of router self-adaptivity** as the central limitation of existing LLM routers, and reframe LLM routing as a problem requiring two learned introspective competencies: task-difficulty introspection and model-capability awareness.
- We introduce **Granule-R1**, an agentic RL framework that unifies decomposition decisions, model selection, and trajectory-aware integration into a single policy trained end-to-end without oracle decomposition or allocation labels, eliminating the rigid two-module pipelines required by prior decompose-then-route systems. The framework further externalizes accumulated capability assessments through a cross-episode **capability-cost profile** that operationalizes model-capability awareness as a queryable structure shared across all routing decisions.
- We empirically show that subquestion-level routing under a self-adaptive policy enables a small-model-only execution stack to match the QA performance of large-model whole-query routing at substantially lower cost, demonstrating that granularity management—not model capacity—is the decisive lever for cost-effective LLM coordination.

2 Related Work

Our work studies cost-efficient question answering over heterogeneous LLMs, with a focus on compositional and multi-hop questions. A broad line of work improves QA by strengthening the reasoning or evidence-access ability of a single model. Chain-of-thought prompting elicits intermediate reasoning steps (Wei et al., 2022), while retrieval-augmented generation couples parametric generation with retrieved evidence (Lewis et al., 2020). Agentic QA systems further train LLMs to interact with external tools or search engines during reasoning; Search-R1, for example, uses reinforcement learning to teach models to interleave reasoning and search queries (Jin et al., 2025).

LLM routing balances answer quality and inference cost across heterogeneous model pools. FrugalGPT reduces cost through cascaded LLM calls (Chen et al., 2023), Hybrid LLM routes each query to a small or large model according to predicted difficulty and target quality (Ding et al., 2024), and RouteLLM learns strong-versus-weak model routing from preference data (Ong et al., 2024). RouterDC improves query-to-model matching with dual contrastive learning (Chen et al., 2024), while GraphRouter models task–query–LLM interactions with a heterogeneous graph (Feng et al., 2024). Benchmarks such as RouterBench, RouterEval, and LLMRouterBench standardize the evaluation of routing systems and reveal both model complementarity and the remaining gap to oracle routing (Hu et al., 2024; Huang et al., 2025; Li et al., 2026). Recent work makes routing sequential or reasoning-aware: Router-R1 formulates multi-LLM routing and aggregation as a sequential decision process (Zhang et al., 2025a); Route-to-Reason jointly selects the language model and reasoning strategy under budget constraints (Zhang et al., 2025b); R2-Router treats output-length budget as an additional routing variable (Xue et al., 2026); and UniRoute studies routing under dynamic model pools with unseen test-time models (Jitkrittum et al., 2025). Closest to our setting are fine-grained collaboration methods such as R2-Reasoner, which combines a task decomposer with a subtask allocator (Shao et al., 2026), and RouteGoT, which performs node-adaptive routing inside graph-of-thought reasoning pipelines (Liu et al., 2026). These methods validate multi-model collaboration and fine-grained allocation, but many routers still treat the original question as the atomic routing unit, while subtask-level methods are often tied to a fixed decomposition stage, graph scaffold, or

specialized execution pipeline. Granule-R1 instead learns an adaptive *decompose-or-route* policy for QA: it can skip decomposition for simple questions, decompose compositional questions into model-appropriate subquestions, and route each subquestion to the cheapest sufficient model.

3 GRANULE-R1

Granule-R1 jointly learns whether to answer directly, route a whole question, or construct and execute an ordered subquestion plan. A single policy produces decomposition, model selection, subanswer commitments, and final integration without oracle decomposition or allocation labels. Figure 2 summarizes its interaction with the external model pool.

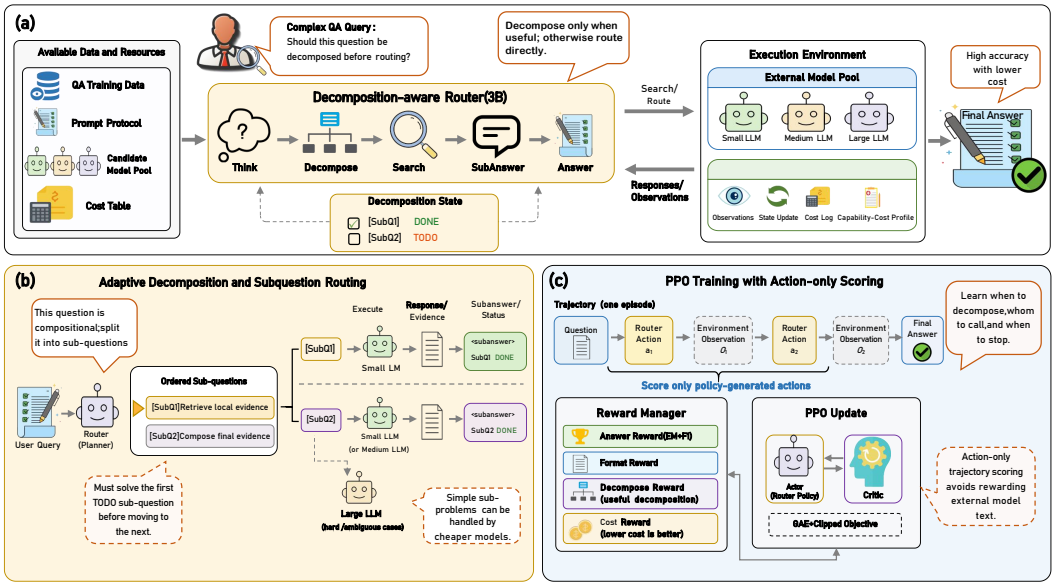

Figure 2: Granule-R1. A 3B policy controls decomposition, model invocation, and subanswer commitment through a stateful environment. Joint correctness and efficiency rewards train its granularity and allocation decisions; PPO scores only router-generated actions.

3.1 GRANULARITY-AWARE ROUTING STATE

Given question x, the router accesses a nonempty pool $\mathcal{P}$ with model descriptors d_L and fixed output-price coefficients $p_L > 0$. Its decision state is

$$s_t = (x, H_t, D_t, C_t, \mathrm{Desc}(\mathcal{P})), \qquad D_t = \{(k, q_k, z_k, \hat{y}_k, e_k)\}_{k=1}^{B_t}, \tag{1}$$

where H_t is the transcript, C_t the external-call cost, and each plan entry stores a subquestion q_k, status $z_k \in \{\mathrm{TODO}, \mathrm{DONE}\}$, committed answer $\hat{y}_k$, and ordered call log e_k. Initially $D_1 = \emptyset$ and $C_1 = 0$. Direct answering leaves $B = 0$; whole-query routing creates a singleton; explicit decomposition creates a plan of width $2 \le B \le B_{\max}$. The semi-MDP trajectory and objective are given in Appendix A.2.

3.2 STRUCTURED EXECUTION

Each decision contains THINK text followed by one executable action: DECOMPOSE$(q_1, \ldots, q_B)$ creates the plan; SEARCH(k, L, m) sends message m to model L and appends its response; COMMIT$(k, \hat{y})$ stores a subanswer and marks it DONE; and ANSWER(y_a) outputs the router's integrated answer. These correspond to `<decompose>`, `<search>`, `<subanswer>`, and `<answer>`. Algorithm 1 handles Search and Commit; full validation, corrective feedback, and termination handling are in Appendix A.2.1.

Algorithm 1 Ordered execution of Search and Commit

Require: Validated Search or Commit action targeting index k; state (x, H, D, C)
1: Copy D; for the first Search on an empty plan, create a singleton TODO entry for x.
2: Find the first TODO; if absent or its index differs from k, reject and return.
3: **if** the action is Search **then**
4: Call the model; log model, message, and response in order; add its output charge to C.
5: **else**
6: Store the subanswer and mark the entry DONE; add no external charge.
7: **end if**
8: Save the working plan to D; append the observation and plan status to H.

One-shot decomposition creates at most one plan, only while no plan exists. Joint correctness and efficiency training learns when and how to decompose and allocate models. The trained one-shot policy achieves higher accuracy with fewer calls in the matched comparison; inference-time replanning adds cost without a material gain (Appendix A.9). Model selection and evidence acquisition remain adaptive within the fixed plan.

3.3 Correctness-Gated Efficiency Reward

Let $c_{\text{ans}} = \mathbf{1}[\text{EM}(y_a, \mathcal{Y}^\star) = 1]$, where gold answers $\mathcal{Y}^\star$ are used only for training rewards. With $f = \text{F}_1(y_a, \mathcal{Y}^\star)$, the outcome reward is

$$R_{\text{out}} = c_{\text{ans}} + \gamma(1 - c_{\text{ans}}) \min\left\{1, \max\left(0, \frac{f - \delta}{1 - \delta}\right)\right\}, \tag{2}$$

where $\gamma = 0.2$ caps non-exact F1 shaping and $\delta = 0.4$ sets its activation threshold. For the final recorded plan of width B, let $\mathcal{K}_{\text{ext}}$ contain committed subquestions with external calls and $\mathcal{K}_{\text{self}}$ those committed without one. For $k \in \mathcal{K}_{\text{ext}}$, $L_k^\star$ is the last invoked model and ι_k its response-informativeness indicator. With $p_{\min} = \min_{L \in \mathcal{P}} p_L$, efficiency credit is

$$R_{\text{eff}} = \begin{cases} \dfrac{1}{B}\left(|\mathcal{K}_{\text{self}}| + \displaystyle\sum_{k \in \mathcal{K}_{\text{ext}}} \iota_k \frac{p_{\min}}{p_{L_k^\star}}\right), & B > 0, \\ 0, & B = 0. \end{cases} \tag{3}$$

Router-only commitments incur no external price and receive unit efficiency credit before plan averaging; exact final correctness gates this credit. The total reward is

$$\boxed{r_\phi(x, y) = R_{\text{out}} + \lambda_{\text{fmt}} \bar{R}_{\text{fmt}} + \lambda_{\text{dec}} R_{\text{dec}} + c_{\text{ans}} \lambda_{\text{eff}} R_{\text{eff}} + R_{\text{adj}}.} \tag{4}$$

$\bar{R}_{\text{fmt}}$ penalizes format errors; R_{dec} credits explicit-plan structure and utility; R_{adj} penalizes repeated failures and rewards successful conditional escalation. Full definitions, weights, and boundary cases are in Appendix A.2.2. Correctness gating removes efficiency credit for wrong final answers, but does not verify intermediate answers or guarantee necessary decomposition (Appendix A.12).

3.4 Policy Optimization

PPO scores only router-generated actions. External responses and environment state updates remain in the context for subsequent decisions, but are excluded from the action likelihood. For router segment $a_t = (y_{m_t}, \ldots, y_{n_t})$, the chain rule gives

$$\log \pi_\theta(a_t \mid s_t) = \sum_{i=m_t}^{n_t} \log \pi_\theta(y_i \mid s_t, y_{m_t:i-1}). \tag{5}$$

The trajectory law P_θ includes router generation and external responses; we optimize

$$\max_\theta \; \mathbb{E}_{x \sim \mathcal{D},\, y \sim P_\theta(\cdot|x)}[r_\phi(x, y)] - \beta\, \mathbb{E}_t\, \text{KL}(\pi_\theta(\cdot \mid s_t) \,\|\, \pi_{\text{ref}}(\cdot \mid s_t)), \tag{6}$$

with frozen reference π_{ref} and averaging over visited router decision states. Appendix A.2.3 gives the PPO surrogate, critic and advantage estimator; segment likelihood factorization does not make segment clipping equivalent to token-wise clipping.

An optional capability–cost profile (CCP) converts past calls into inference-time hints, without trainable parameters. CCP is off in all main experiments; construction and the pool-shift case study are in Appendices A.2.4 and A.13.

4 Experimental Setup

We evaluate Granule-R1 on seven question-answering benchmarks spanning two regimes. The general QA suite consists of Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023), in which most questions are answerable from a single fact or short reasoning chain. The multi-hop QA suite consists of HotpotQA (HpQA) (Yang et al., 2018), 2WikiMultiHopQA (2Wiki) (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Bamb) (Press et al., 2023), which require composing multiple facts or comparing entities. Together they exercise both the simple-input regime, where the router should answer directly or use a single model call, and the compositional regime, where decomposition becomes profitable. We report Exact Match (EM) and token-level F1, evaluating on 500 randomly sampled instances per dataset; Bamboogle uses its full 125-instance test set.

The router is initialized from Qwen2.5-3B-Instruct (Yang et al., 2024) and trained with veRL-style PPO (Sheng et al., 2024) using vLLM (Kwon et al., 2023) rollouts on a 14K-example mixture of NQ and HotpotQA. We use actor learning rate 10^{-6}, critic learning rate 10^{-5}, KL coefficient $\beta = 10^{-3}$, PPO clip ratio $\epsilon = 0.2$, maximum turns $T = 6$, and maximum plan width $B_{\max} = 4$. The Full Pool contains Qwen2.5-7B (Yang et al., 2024), LLaMA-3.1-8B (Dubey et al., 2024), Mistral-7B (Jiang et al., 2023), Gemma-2-27B (Gemma Team et al., 2024), Mixtral-8x22B (Jiang et al., 2024), and LLaMA-3.1-70B (Dubey et al., 2024). The Small Pool contains only the three 7B-tier models. Routing cost follows Router-R1: the sum over external model calls of price-tier coefficient times output tokens. Expanded setup details are in Appendix A.5.

5 Experimental Results and Analysis

5.1 Main Results: Accuracy and Cost on the Full Pool

The aggregate EM improvement and external-output-token cost reduction are statistically significant under the five-seed paired-bootstrap analysis; detailed results and the evaluation protocol are provided in Appendix A.8.

Table 1 reports EM and F1 on seven benchmarks under the heterogeneous Full Pool. Granule-R1 attains the best average EM (0.450 vs. 0.416 for Router-R1-Qwen, a $+3.3$-point absolute gain) and the best average F1 (0.521 vs. 0.498), and is the top method on every single dataset—covering both in-domain (NQ, HpQA) and five out-of-domain settings—so the gain is not a benchmark-specific artifact. The gap over single-model and prompting baselines (Direct, CoT, SFT, RAG, Search-R1) reflects access to a heterogeneous pool with learned per-call routing rather than a fixed evidence-acquisition pattern. The gap over query-level routers (Prompt LLM, Largest LLM, RouterDC, GraphRouter), which still treat the original query as the atomic decision unit, and the further gap over Router-R1-Qwen, which performs multi-round whole-query routing, isolate the contribution of *routing granularity*: only Granule-R1 exposes the internal structure of a compositional question *before* model selection, so model specialization can be applied at the subquestion level.

Accuracy gains do not come from spending more. On the four cost-evaluated datasets (Figure 3), Granule-R1 raises average EM from 0.390 to 0.423 over Router-R1-Qwen *while* reducing average routing cost from 134.6 to 110.6 (-17.8%), with cost reductions on each individual dataset. Cheaper query-level routers occupy a strictly lower-accuracy region; on the high-accuracy frontier, Granule-R1 is the unique method that is both more accurate and cheaper than the strongest multi-round router. Our cost comparisons exclude router inference for all methods; the compact 3B controller's measured overhead is detailed in Appendix A.11.

Table 1: Main results on seven QA benchmarks under the Full Pool. The upper block reports exact match (EM), and the lower block reports F1 scores for shared baselines. Dagger marks in-domain evaluation. Best results are bolded.

Exact Match (EM)								
Method	**NQ†**	**TriviaQA**	**PopQA**	**HpQA†**	**2Wiki**	**MuSiQue**	**Bamboogle**	**Avg.**
Qwen2.5-3B-Instruct as policy model								
Direct	0.092	0.260	0.122	0.140	0.266	0.026	0.040	0.135
CoT	0.126	0.358	0.160	0.168	0.208	0.046	0.224	0.184
SFT	0.212	0.400	0.160	0.198	0.256	0.052	0.112	0.199
RAG	0.298	0.540	0.366	0.216	0.146	0.078	0.224	0.267
Search-R1	0.328	0.510	0.324	0.236	0.278	0.090	0.272	0.291
Prompt LLM	0.300	0.580	0.340	0.268	0.262	0.108	0.448	0.329
Largest LLM	0.296	0.578	0.354	0.278	0.274	0.104	0.480	0.338
RouterDC	0.278	0.592	0.282	0.244	0.218	0.080	0.504	0.314
GraphRouter	0.276	0.586	0.280	0.234	0.180	0.076	0.448	0.297
Router-R1-Qwen	0.388	0.706	0.384	0.352	0.434	0.138	0.512	0.416
Granule-R1	**0.432**	**0.752**	**0.428**	**0.384**	**0.454**	**0.158**	**0.544**	**0.450**
F1-score								
Method	**NQ†**	**TriviaQA**	**PopQA**	**HpQA†**	**2Wiki**	**MuSiQue**	**Bamboogle**	**Avg.**
Qwen2.5-3B-Instruct as policy model								
Direct	0.162	0.341	0.154	0.215	0.304	0.081	0.112	0.196
CoT	0.218	0.431	0.185	0.260	0.251	0.106	0.332	0.255
SFT	0.289	0.460	0.207	0.281	0.291	0.121	0.173	0.260
RAG	0.414	0.622	0.452	0.307	0.187	0.134	0.303	0.346
Search-R1	0.407	0.575	0.383	0.328	0.317	0.145	0.387	0.363
Prompt LLM	0.437	0.694	0.409	0.407	0.393	0.205	0.579	0.446
Largest LLM	0.431	0.695	0.423	0.416	0.397	0.199	0.608	0.453
RouterDC	0.410	0.694	0.328	0.381	0.293	0.165	0.623	0.413
GraphRouter	0.412	0.690	0.324	0.366	0.258	0.154	0.546	0.393
Router-R1-Qwen	0.484	0.772	0.447	0.449	0.487	0.212	0.635	0.498
Granule-R1	**0.507**	**0.806**	**0.460**	**0.462**	**0.515**	**0.238**	**0.662**	**0.521**

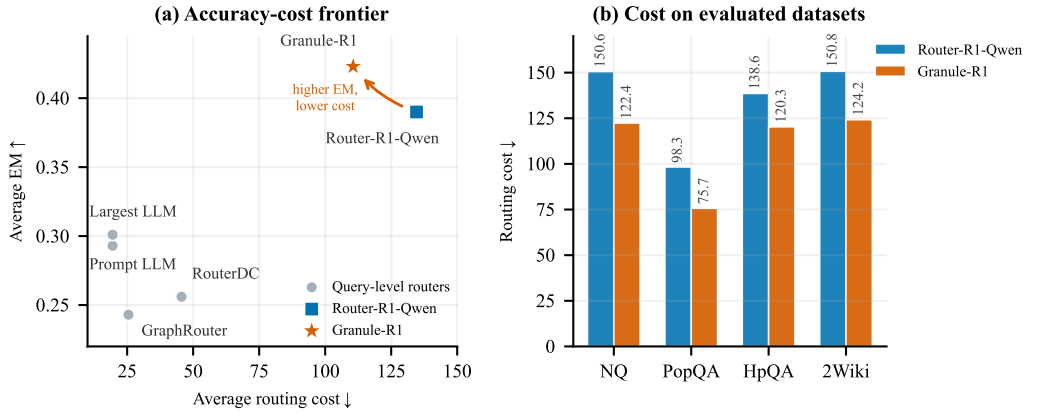

Figure 3: Accuracy–cost visualization on the four cost-evaluated datasets under the Full Pool. (a) Average EM versus average routing cost: Granule-R1 moves the high-accuracy frontier upward and left relative to Router-R1-Qwen. (b) Dataset-wise routing cost for the two strongest multi-round methods: Granule-R1 reduces cost on all four datasets while improving average EM.

5.2 GRANULARITY ANALYSIS: SMALL-POOL GRANULE-R1 VS. MID-TIER WHOLE-QUERY METHODS

To isolate the granularity mechanism from raw model capacity, we restrict Granule-R1 to a *Small Pool* containing only Qwen2.5-7B, LLaMA-3.1-8B, and Mistral-7B—no 27B/70B model is accessible. Table 2 contains the result.

Table 2: Granularity analysis on the four cost-evaluated datasets. The top block shows whole-query methods with access to a 27B model. The middle block restricts whole-query methods to 7B-tier models. The bottom row is Granule-R1 restricted to the same 7B-tier Small Pool.

Method	**Pool**	**NQ**†		**PopQA**		**HpQA**†		**2Wiki**		**Avg. EM**	**Avg. Cost**
		EM	Cost	EM	Cost	EM	Cost	EM	Cost	↑	↓
Whole-query inference with access to a 27B model											
Direct prompting (Gemma-2-27B)	{27B}	0.282	42.7	0.290	22.0	0.244	37.8	0.190	44.6	0.252	36.8
Prompt LLM (selects from $\mathcal{P}_{27}$)	$\mathcal{P}_{27}$	0.296	28.4	0.318	19.3	0.262	25.6	0.234	26.2	0.278	24.9
RouterDC	$\mathcal{P}_{27}$	0.284	41.5	0.295	21.6	0.252	38.8	0.218	31.4	0.262	33.3
GraphRouter	$\mathcal{P}_{27}$	0.286	32.2	0.291	20.8	0.245	28.6	0.210	27.5	0.258	27.3
Router-R1-Qwen	$\mathcal{P}_{27}$	0.310	78.5	0.306	41.6	0.302	64.2	0.286	56.4	0.301	60.2
Whole-query inference restricted to 7B-tier models (Small Pool)											
Direct prompting (Qwen2.5-7B)	{7B}	0.138	24.2	0.130	18.6	0.152	24.9	0.166	28.7	0.147	24.1
Direct prompting (LLaMA-3.1-8B)	{8B}	0.242	29.5	0.208	14.3	0.198	24.1	0.130	21.3	0.195	22.3
Prompt LLM (selects from Small)	Small	0.218	25.4	0.196	16.1	0.184	23.7	0.170	24.0	0.192	22.3
Router-R1-Qwen	Small	0.244	32.8	0.224	16.5	0.194	28.4	0.180	27.6	0.211	26.3
Subquestion-level routing restricted to 7B-tier models (this work)											
Granule-R1 (Small)	**Small**	**0.318**	**30.6**	**0.306**	**17.4**	**0.302**	**28.7**	**0.286**	**31.1**	**0.303**	**27.0**

Granularity, not capacity, is the decisive lever. Two contrasts pin down the claim. *Same pool, different granularity:* restricting Router-R1-Qwen to the same Small Pool yields only 0.211 average EM, so Granule-R1's 0.303 is a $+9.2$-point improvement at nearly the same cost (27.0 vs. 26.3)—a gap that cannot come from model capacity since both methods see the same three 7B models. *Different pool, same granularity:* Granule-R1 (Small) also outperforms all static and query-level methods equipped with a 27B model (Direct Gemma-2-27B 0.252, Prompt LLM 0.278, RouterDC 0.262, GraphRouter 0.258). Moreover, it surpasses the strongest 27B-access multi-round router by 0.02 EM points (0.303 vs. 0.301), *at less than half its cost* (27.0 vs. 60.2, a 55.1% reduction).The mechanism is local simplification: the router exposes locally simple subquestions to small models and performs the integration step itself, so a multi-hop question that exceeds a 7B model as a single unit is solved by composing factual lookups and short comparisons—aligning model calls with the actual difficulty of each subproblem.

5.3 ABLATION STUDY

Table 3 ablates the core training-time design choices on the four cost-evaluated datasets, reporting also *Cost / Correct* as a unified efficiency metric. *(i) Decomposition action* provides expressivity that whole-query multi-round routing cannot recover: removing a_{dec} drops EM to 0.388 and *raises* cost to 136.4, as the policy compensates with more expensive whole-query calls. *(ii) Decomposition reward* makes the action productive; without it the policy emits sporadic, ill-structured plans (EM 0.374, cost 152.2). *(iii) Efficiency reward* is the cost lever, not an accuracy regularizer: removing it lifts EM to 0.428 but cost balloons to 177.4, the worst Cost/Correct 414.5—the policy buys accuracy by overusing expensive models. *(iv) Outcome gating* prevents reward hacking; removing it crashes EM to 0.272 at cost 82.5—the policy learns cheapness before correctness. *(v) Action-only scoring* is essential under multi-LLM rollouts; scoring environment-supplied tokens drops EM to 0.349, confirming the gradient must train router decisions, not other models' text.

5.4 BEHAVIORAL ANALYSIS

Granularity follows question structure. The policy invokes DECOMPOSE on 12.7% of TriviaQA and 18.4% of NQ inputs, compared with 73.2% on 2Wiki and 81.6% on Bamboogle. This pattern is con-

Table 3: Ablation on the four cost-evaluated datasets, Full Pool. "Cost / Correct" is average routing cost per correct answer (lower is better).

Method	Avg. EM↑	Avg. Cost↓	Cost / Correct↓
Granule-R1 (full)	**0.423**	**110.6**	**261.4**
w/o decomposition action	0.388	136.4	351.5
w/o decomposition reward	0.374	152.2	407.1
w/o action-only scoring	0.349	128.3	367.6
w/o efficiency reward	0.428	177.4	414.5
w/o outcome-gated composition	0.272	82.5	303.3

sistent with learned sensitivity to compositional demands, without difficulty or decomposition labels. A question that is not decomposed may be answered directly or routed as a whole; decomposition frequency therefore does not measure self-answering. The policy learns when the potential benefit of a structured plan warrants additional execution.

Most subquestion calls use smaller candidates. Among decomposed trajectories, 71.3% of external subquestion calls use 7B-tier models, 18.5% use the 27B-tier, and 10.2% use the 70B-tier. These are call shares, excluding router-only commitments, rather than fractions of queries. The concentration on smaller models is consistent with making individual routing units easier through decomposition. The $+9.2$-point Small-Pool EM gap over Router-R1-Qwen provides complementary evidence: the gain persists when both routers have access to the same small candidates (Table 2).

Plan width and execution adapt within the one-shot policy. Average recorded width is 1.04 on NQ, 2.18 on HotpotQA, and 2.71 on MuSiQue. The NQ mean includes plan usage beyond explicit decomposition, whose minimum width is two. Only 2.3% of MuSiQue episodes reach $B_{\max} = 4$, suggesting that the policy does not routinely exhaust the width budget. Once a plan is formed, the router can still change the selected model, retry after an uninformative response, or commit a subanswer from its own knowledge. The latter incurs no external-call price. Appendix A.4 illustrates self-resolution and conditional escalation; Appendix A.9 evaluates the resulting performance–overhead trade-off.

Conservative decomposition still leaves avoidable steps. The paired inference audit identifies redundant plans when both variants are correct but suppressing decomposition is cheaper, and harmful plans when decomposition changes a correct answer into an incorrect one. The failure audit also distinguishes model selection, incorrect commitments, and aggregation errors. These diagnostics delimit what the learned routing behavior establishes: selective decomposition does not guarantee necessary subquestions or verified intermediate answers. Detailed frequencies and audit protocols are in Appendix A.12.

6 Conclusion

Granule-R1 makes routing granularity a learned part of multi-model question answering. Through joint correctness and efficiency optimization, a single policy learns whether to answer directly, route a whole query, or construct and execute an ordered subquestion plan. Persistent subquestion state and explicit commitments connect these decisions to local allocation credit and final-answer quality, without oracle decomposition or allocation labels. Across seven QA benchmarks, the original evaluation improves average EM by 3.3 points over Router-R1-Qwen and reduces the external-output-token cost proxy by 17.8% on four datasets. The same-pool small-model comparison further supports the value of learning the routing unit.

The supplementary experiments show that the trained one-shot policy achieves a favorable performance–overhead trade-off relative to the evaluated recursive variant, while execution remains adaptive within each plan. MATH-500 extends the evidence to mathematical QA. These findings motivate granularity-aware coordination, with current limitations in intermediate-answer verification, full serving-cost accounting, and non-monotonic replanning. Extending the learned policy to broader agent tasks and evaluating its robustness across changing model pools remain important directions.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and polishing, literature search and the organization of related work, and the collection and organization of experimental data. The authors take full responsibility for the final manuscript, including the accuracy of the references, experimental results, and conclusions.

REFERENCES

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023. URL https://arxiv.org/abs/2305.05176.

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*, 2024.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024. URL https://arxiv.org/abs/2404.14618.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. Successive prompting for decomposing complex questions. *arXiv preprint arXiv:2212.04092*, 2022. URL https://arxiv.org/abs/2212.04092.

Abhimanyu Dubey et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. *arXiv preprint arXiv:2410.03834*, 2024. URL https://arxiv.org/abs/2410.03834.

Gemma Team et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024. URL https://arxiv.org/abs/2408.00118.

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

Qian Hu, Jacob Bieker, Xi Li, Nan Jiang, Ben Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shyam Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024. URL https://arxiv.org/abs/2403.12031.

Zhongzhan Huang, Guoming Ling, Vincent S. Liang, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. *arXiv preprint arXiv:2503.10657*, 2025. URL https://arxiv.org/abs/2503.10657.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. URL https://arxiv.org/abs/2310.06825.

Albert Q. Jiang et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. URL https://arxiv.org/abs/2401.04088.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025. URL https://arxiv.org/abs/2503.09516.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Zifeng Wang, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient llm inference. *arXiv preprint arXiv:2502.08773*, 2025. URL https://arxiv.org/abs/2502.08773.

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*, 2017.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations*, 2023.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*, 2023. URL https://arxiv.org/abs/2309.06180.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. Llmrouterbench: A massive benchmark and unified framework for llm routing. *arXiv preprint arXiv:2601.07206*, 2026. URL https://arxiv.org/abs/2601.07206.

Yuhang Liu, Ruijie Wang, Yunlong Chu, Bing Hao, Yumeng Lin, Shengzhong Liu, and Minglai Shao. Routegot: Node-adaptive routing for cost-efficient graph of thoughts reasoning. *arXiv preprint arXiv:2603.05818*, 2026. URL https://arxiv.org/abs/2603.05818.

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*, 2023.

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 with preference data. *arXiv preprint arXiv:2406.18665*, 2024. URL https://arxiv.org/abs/2406.18665.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023.

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*. ACM, 2026. doi: 10.1145/3774904.3793038. URL https://doi.org/10.1145/3774904.3793038.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024. URL https://arxiv.org/abs/2409.19256.

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

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022. URL https://arxiv.org/abs/2212.03533.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Jian Wu, Linyi Yang, Yuliang Ji, Wenhao Huang, Börje F. Karlsson, and Manabu Okumura. Gendec: A robust generative question-decomposition method for multi-hop reasoning. *arXiv preprint arXiv:2402.11166*, 2024. URL https://arxiv.org/abs/2402.11166.

Jiaqi Xue, Qian Lou, Jiarong Xing, and Heng Huang. R2-router: A new paradigm for llm routing with reasoning. *arXiv preprint arXiv:2602.02823*, 2026. URL https://arxiv.org/abs/2602.02823.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. *arXiv preprint arXiv:2506.09033*, 2025a. URL https://arxiv.org/abs/2506.09033.

Pan Zhang et al. Route to reason: Adaptive routing for llm and reasoning strategy selection. *arXiv preprint arXiv:2505.19435*, 2025b. URL https://arxiv.org/abs/2505.19435.

Shu Zhao, Tan Yu, Anbang Xu, Japinder Singh, Aaditya Shukla, and Rama Akkiraju. Parallelsearch: Train your llms to decompose query and search sub-queries in parallel with reinforcement learning. *arXiv preprint arXiv:2508.09303*, 2025. URL https://arxiv.org/abs/2508.09303.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023.

A APPENDIX

APPENDIX CONTENTS

- Additional Related Work . . . . . p. 13
- Methodology Derivations . . . . . p. 14
  - Action Grammar and State Transitions . . . . . p. 14
  - Reward Components and Hyperparameters . . . . . p. 17
  - Action-Segment PPO and Token-Level Implementation . . . . . p. 20
  - Capability-Cost Profile Construction . . . . . p. 21
- Training Prompts . . . . . p. 21
- Routing Case Studies . . . . . p. 25
- Experiment Implementation and Reproducibility Details . . . . . p. 26
- Baseline Provenance and Comparison Protocols . . . . . p. 28
- Ablation Supplement . . . . . p. 28
- Five-Seed Statistical Evaluation . . . . . p. 29
- One-Shot versus Recursive Decomposition . . . . . p. 30
- Supplementary Mathematical QA Evaluation . . . . . p. 31
- Training and Controller Overhead . . . . . p. 31
- Failure, Decomposition, and Informativeness Audits . . . . . p. 32
- Optional CCP Pool-Shift Case Study . . . . . p. 33
- Price Mapping and Scope of the Cost Proxy . . . . . p. 34

A.1 ADDITIONAL RELATED WORK

Query-level LLM routing. A large body of work studies routing at the level of the original user query. Early methods such as FrugalGPT use cascades to reduce inference cost while preserving or even improving quality (Chen et al., 2023). Hybrid LLM routes a query to either a small or a large model based on predicted difficulty and a target quality level (Ding et al., 2024). RouteLLM learns strong-versus-weak model routing from preference data (Ong et al., 2024), while RouterDC and GraphRouter improve query-to-model matching using contrastive learning and graph-based modeling, respectively (Chen et al., 2024; Feng et al., 2024). More recent work extends this line toward dynamic model pools and richer routing variables, e.g., UniRoute for unseen test-time models and R2-Router for jointly selecting a model and an output-length budget (Jitkrittum et al., 2025; Xue et al., 2026). Despite their differences, these methods largely treat the original query as the atomic routing unit.

Multi-round and reasoning-aware routing. Recent work moves beyond one-shot selection by making routing itself sequential and reasoning-aware. Router-R1 formulates multi-LLM routing and aggregation as a sequential decision process that interleaves internal thinking and external model invocation (Zhang et al., 2025a). Route-to-Reason further studies the joint selection of both the language model and the reasoning strategy under budget constraints (Zhang et al., 2025b). These methods are closely related to our setting because they demonstrate that routing can be embedded inside a multi-step reasoning process rather than performed only once before generation. Our work differs in that it explicitly models whether the question should be decomposed as a first-class routing decision, rather than leaving decomposition to emerge implicitly inside a generic multi-round reasoning trajectory.

Question decomposition for reasoning, retrieval, and QA. Another line of work shows that decomposing a complex question into simpler subproblems can substantially improve compositional reasoning. Least-to-Most prompting solves complex problems by breaking them into easier subproblems (Zhou et al., 2023). Decomposed Prompting delegates simpler sub-tasks to specialized prompting modules (Khot et al., 2023), while Self-Ask explicitly generates follow-up questions before answering the original one (Press et al., 2023). Successive Prompting and GenDec further explore

decomposition for complex and multi-hop question answering (Dua et al., 2022; Wu et al., 2024). In retrieval-augmented settings, question decomposition has also been used to improve evidence coverage for multi-hop QA, and recent RL-based search agents such as Search-R1 and ParallelSearch learn to issue or decompose search queries during reasoning (Jin et al., 2025; Zhao et al., 2025). However, these methods primarily use decomposition as a reasoning or retrieval aid; they do not study cost-aware subquestion routing over a heterogeneous pool of LLMs.

Subtask-level routing and structured multi-model collaboration. Closest to our work are recent approaches that route at a finer granularity than the full query. R2-Reasoner introduces a two-module architecture with a task decomposer and a subtask allocator, enabling heterogeneous models to collaborate on decomposed subtasks (Shao et al., 2026). RouteGoT performs node-adaptive routing within graph-of-thought reasoning under explicit budget control (Liu et al., 2026). These works validate the value of fine-grained allocation, but they are typically tied to a specific execution scaffold such as a fixed decomposition stage or graph-structured reasoning. By contrast, our method targets question answering directly and learns an adaptive *decompose-or-route* policy over natural-language questions, allowing the router to skip decomposition when it is unnecessary and to exploit subquestion-level routing only when it improves the accuracy–cost trade-off.

Benchmarks and evaluation for routing. The routing community has also invested in standardized evaluation. RouterBench introduced one of the first systematic benchmarks for multi-LLM routing (Hu et al., 2024). RouterEval significantly expanded the scale of evaluation by collecting performance records over thousands of LLMs (Huang et al., 2025). More recently, LLMRouterBench re-evaluated representative routing methods under a unified framework and found that many routers remain surprisingly close to simple baselines, with a substantial gap to the oracle still remaining (Li et al., 2026). These results suggest that further progress may require not only better query-to-model matching, but also alternative formulations of routing itself. Our work explores one such alternative by making routing granularity adaptive: the router first decides whether to decompose a question, and then routes the resulting subquestions to cost-appropriate models.

A.2 Methodology Derivations

This section provides closed-form expressions for every reward component, the full action-segment PPO objective with token-level masking and GAE, the action grammar with the complete transition function, and the Capability-Cost Profile construction referenced in Section 3.

Trajectory and regularized objective. A decision segment and the resulting environment interaction form one variable-duration macro-action. Let $T_y \leq T$ be the realized horizon, with router action a_t, observation o_t, and final answer y_a:

$$y = (s_1, a_1, o_1, \ldots, s_{T_y}, a_{T_y}, o_{T_y}, y_a), \tag{7}$$

The trajectory distribution P_θ in Eq. (6) includes both router generation and external-response sampling. The undiscounted setting $\gamma_{\text{ppo}} = 1$ does not introduce duration-dependent discounting. External observations are part of the state history but are not router actions.

A.2.1 Action Grammar and State Transitions

Tag-level grammar and per-turn decision flow. The four executable action types are DECOMPOSE (a_{dec}), SEARCH (a_{srch}), COMMIT (a_{sub}), and ANSWER (a_{ans}), with the arguments specified below. Every router turn begins with a private `<think>` block in which the policy reasons about the current state and *then* commits to exactly one *executable* action. The think-first ordering is strict: the router never emits `<decompose>`, `<search>`, `<subanswer>`, or `<answer>` without an immediately preceding `<think>`. After an executable action that triggers external interaction, the environment appends a *passive* block (`<information>` after a `<search>`, `<decomposition_state>` after every `<decompose>`, `<search>`, or `<subanswer>`); these passive blocks are part of the transcript H_t but do not appear in the policy likelihood (Appendix A.2.3).

Algorithm 2 gives the per-decision dispatch, including invalid actions, direct answering, whole-query routing, explicit decomposition, and budget exhaustion. The router *thinks before deciding* whether to decompose, *thinks again* before choosing which model to invoke, and *thinks once more* before

committing each subanswer or producing the final answer. The five tag families and their concrete formats are listed below; a complete trajectory using them all appears in Appendix A.4, Case 3.

- DECOMPOSE — Router-emitted, *after* a <think> block in which the policy has judged the question compositional enough to warrant decomposition. Allowed at most once per episode and only while the plan is empty. The body is a newline-separated bulleted list of two to $B_{\max}$ self-contained subquestions, in the order they should be solved.

```
<think>
The question requires the director's nationality, which decomposes
↪ into
(i) identifying the director, then (ii) recovering their
↪ nationality.
</think>
<decompose>
- Who directed the film "Amelie"?
- What is the nationality of this director?
</decompose>
```

- SEARCH — Router-emitted, *after* a <think> block in which the policy has selected the candidate $L \in \mathcal{P}$ that best matches the active subquestion's capability–cost requirement. Routes a message to L for the first unresolved subquestion. The bracket prefix [SubQk] must match that entry's index; the model identifier L must be one of the descriptors in $\mathrm{Desc}(\mathcal{P})$.

```
<think>
SubQ1 is a foreign-language-title fact lookup; route to a 70B
↪ model
to maximize informativeness on this proper noun.
</think>
<search>[SubQ1] LLaMA-3.1-70B-Instruct: Who directed the film
↪ Amelie?</search>
```

- <information> — *Environment-emitted, not a router action.* The verbatim response returned by the invoked model L is wrapped in this tag and appended to the evidence buffer e_k (Algorithm 1); cost C_t is updated correspondingly.

```
<information>[SubQ1] The film "Amelie" (2001) was directed by
Jean-Pierre Jeunet.</information>
```

- COMMIT — Router-emitted. Commits a concise $\hat{y}_k$ for the active subquestion and marks that entry DONE. The resolving model $L_k^\star$ is the last external model invoked before this commit. When the router commits without any external call, no resolving model price is evaluated and the self-resolution branch of Eq. (13) supplies unit efficiency credit.

```
<subanswer>[SubQ1] Jean-Pierre Jeunet</subanswer>
```

- <decomposition_state> — *Environment-emitted, not a router action.* After every DECOMPOSE, COMMIT, and SEARCH, the environment renders the current D_t into a textual status block so that the router can condition future decisions on plan progress. TODO and DONE reflect z_k; resolved subquestions display their committed $\hat{y}_k$.

```
<decomposition_state>
[SubQ1][DONE] Who directed the film "Amelie"? => Jean-Pierre
↪ Jeunet
[SubQ2][TODO] What is the nationality of this director?
</decomposition_state>
```

- ANSWER — Router-emitted, terminates the episode. Must be emitted exactly once per trajectory; emitting it while any $z_k = \mathrm{TODO}$ remains triggers $I_{\mathrm{early}} = 1$ (Eq. (11)).

```
<answer> French </answer>
```

The crucial distinction is between *router-emitted* executable tags (<decompose>, <search>, <subanswer>, <answer>, plus the private <think> span) and *environment-emitted* passive blocks (<information>, <decomposition_state>). Both appear in the rolled-out transcript H_t and condition future router decisions, but only the router-emitted ones contribute to the action likelihood $\log \pi_\theta(a_t|s_t)$ in Eq. (19)—this is what the action-only mask in Appendix A.2.3 enforces.

Execution procedure. The environment updates its records after observing the actual model response, which may be stochastic. Algorithm 2 validates and dispatches one router decision; Algorithm 1 handles ordered Search and Commit. A working copy ensures that an invalid first whole-query Search does not leave a partially initialized singleton behind. Call logs retain all returned responses in order, including repeats. Each external charge is the selected model's fixed output-price coefficient times its recorded output-token count, matching the evaluation metric in Appendix A.14.

Algorithm 2 Complete environment dispatch for one router decision

Require: Current state (x, H, D, C), router output, remaining turn budget
if the budget is exhausted: terminate; use an empty answer if none was emitted.
Record the router output in H and consume one decision turn.
Parse one Think block followed by exactly one recognized executable action.
if parsing fails or required fields are missing or empty: reject and return.
if the action is Decompose **then**
    if a plan exists or its proposed width is outside two to $B_{\max}$: reject and return.
    Initialize the ordered subquestions as TODO, with empty answers and call logs.
    Append the plan-status observation to H.
else if the action is Search or Commit **then**
    if Search names a model outside the candidate pool: reject and return.
    Run Algorithm 1 on the validated action and current state.
else ▷ Answer
    if any TODO entry remains: record early termination for reward computation.
    Terminate the episode with the emitted final answer.
end if

Here, "reject and return" appends corrective feedback and returns control to the router; it leaves the plan and cost unchanged and does not itself terminate the episode.

One-shot decomposition guard. DECOMPOSE is allowed at most once per episode and only while the plan is empty. Any subsequent DECOMPOSE attempt is treated as a grammar violation. This restriction stabilizes credit assignment: every committed subanswer is attributable to a fixed plan rather than to a sequence of plans being revised, and the policy's adaptivity is concentrated in (i) whether to decompose at all, (ii) how many subquestions to generate, and (iii) which model to invoke for each one—rather than in recursive replanning.

Corrective observations. Any of the following triggers a corrective observation $o_t = \mathrm{Reject}(a_t)$ which is appended to H_t but does not advance D_t or C_t: malformed or nested tags, an unknown model name in $L \notin \mathcal{P}$, repeated DECOMPOSE, an index that does not target the first TODO on SEARCH or COMMIT, or a missing required field. The episode does *not* terminate; the router must emit another action. Repeated rejects within one episode count toward the format penalty (Appendix A.2.2). ANSWER is the explicit terminating action; reaching the turn budget also ends a rollout. A rollout with no final answer has an empty predicted answer for outcome scoring. If ANSWER is emitted while some $z_k = \mathrm{TODO}$ remains in D_t, the episode terminates with the early-termination indicator $I_{\mathrm{early}} = 1$ used in the decomposition reward (Eq. (11)).

State space and modes. Table 4 makes explicit how the three operating modes referenced in Section 3 share a single state space.

Table 4: The three operating modes share one state space; the policy is not given a mode label.

Mode	First action sequence	Plan width	External invocation
Direct answer	ANSWER	$B = 0$	none
Whole-query routing	SEARCH$\rightarrow \cdots \rightarrow$ COMMIT$\rightarrow$ ANSWER	$B = 1$ (auto)	≥ 1
Decomposition routing	DECOMPOSE$\rightarrow$ SEARCH/COMMIT$^{B} \rightarrow$ ANSWER	$B \geq 2$	≥ 0

A.2.2 Reward Components and Hyperparameters

This section specifies the reward components referenced in Section 3 (Eqs. (8) and (4)), the hyperparameter table, and the informativeness detector specification.

Outcome reward. Let $\mathcal{Y}^\star$ be the gold answer set, $c_{\text{ans}} = \mathbf{1}[\text{EM}(y_a, \mathcal{Y}^\star) = 1]$, and $f = \text{F}_1(y_a, \mathcal{Y}^\star)$. The outcome reward is

$$R_{\text{out}} = c_{\text{ans}} + \gamma(1 - c_{\text{ans}})\,\phi(f), \qquad \phi(f) = \min\left\{1,\, \max\left(0, \frac{f-\delta}{1-\delta}\right)\right\}, \tag{8}$$

where $\gamma = 0.2$ caps the F1 shaping bonus and $\delta = 0.4$ sets a deadband below which partial answers receive no shaping. With these values, $R_{\text{out}} \in [0, 0.2] \cup \{1\}$: the policy never gets credit between 0.2 and 1 for an inexact answer, so EM dominates.

Format reward (graded penalty). $R_{\text{fmt}} \le 0$ penalizes structural defects in the trajectory. Concretely,

$$R_{\text{fmt}} = \begin{cases} 0 & \text{fully well-formed,} \\ -0.2 & \text{minor defect (e.g., empty/garbled query body} \\ & \text{inside a valid <search>),} \\ -0.5 & \text{invalid model identifier in an otherwise valid SEARCH,} \\ -1.0 & \text{severe defect: unmatched/nested tags,} \\ & \text{multiple <answer>, repeated DECOMPOSE,} \\ & \text{premature termination, persistent active-index violation.} \end{cases} \tag{9}$$

If several defect categories apply, the most severe category determines R_{fmt}. The effective format term used in Eq. (4) is

$$\bar{R}_{\text{fmt}} = \begin{cases} \max\{R_{\text{fmt}}, -0.2\}, & c_{\text{ans}} = 1, \\ R_{\text{fmt}}, & c_{\text{ans}} = 0. \end{cases} \tag{10}$$

Thus correct answers incur at most 0.2 in format penalty before weighting.

Decomposition reward. R_{dec} rewards syntactically valid and outcome-coupled explicit plans. Let $I_{\text{dec}} = 1$ only when a legal explicit decomposition has created a plan ($B \ge 2$); otherwise $I_{\text{dec}} = 0$, so direct-answer and implicit singleton modes receive no explicit-decomposition reward. Then

$$R_{\text{dec}} = I_{\text{dec}}\,\text{clip}\Big(\lambda_{\text{str}} I_{\text{str}} + \lambda_{\text{sub}} N_{\text{sub}} + \lambda_{\text{done}} I_{\text{done}} + \lambda_{\text{util}} U - \lambda_{\text{early}} I_{\text{early}},\; r_{\min},\; r_{\max}\Big), \tag{11}$$

with components

- $I_{\text{str}} \in \{0, 1\}$: indicator that all executable actions in the trajectory respect the grammar and active-index guard;
- $N_{\text{sub}} = \sum_{k=1}^{B} \mathbf{1}[z_k = \text{DONE}]$: number of legally committed subanswers within the current plan;
- $I_{\text{done}} \in \{0, 1\}$: indicator that $B > 0$ and all B subquestions are resolved before ANSWER;
- $I_{\text{early}} \in \{0, 1\}$: indicator that ANSWER was emitted while some $z_k = \text{TODO}$;
- U: plan utility, defined as

$$U = \begin{cases} c_{\text{ans}} I_{\text{done}}\, \dfrac{1}{B} \displaystyle\sum_{k=1}^{B} \iota_k \mathbf{1}[z_k = \text{DONE}], & B > 0, \\ 0, & B = 0. \end{cases} \tag{12}$$

  which can be nonzero only when the final answer is correct, the nonempty plan is complete, and at least one externally supported commitment has positive informativeness. Router-only commitments are excluded from this external-evidence term; they can still receive structural and final-outcome credit. This proxy does not establish that every committed subanswer is correct or necessary.

The clip bounds $[r_{\min}, r_{\max}] = [-0.4, 0.4]$ and $\lambda_{\text{dec}} = 0.5$ imply $\lambda_{\text{dec}} R_{\text{dec}} \in [-0.2, 0.2]$. This bounds the structural component relative to the unit exact-match term, but does not prove that every correct trajectory outranks every incorrect trajectory under the complete reward.

Per-subquestion efficiency reward. Let e_k be the ordered external-call log associated with q_k. For any subquestion with $|e_k| > 0$, $L_k^\star$ is its last invoked model and ι_k is the informativeness indicator of that response. When $|e_k| = 0$, set $\iota_k = 0$ as an external-evidence convention. A router-only commitment incurs no external price and receives unit efficiency credit by convention, as illustrated in Appendix A.4. This reward convention does not verify its answer or assign a price to an absent model. The contribution is

$$\eta_k = \begin{cases} \iota_k \, p_{\min}/p_{L_k^\star}, & z_k = \text{DONE and } |e_k| > 0, \\ 1, & z_k = \text{DONE and } |e_k| = 0, \\ 0, & \text{otherwise}, \end{cases} \qquad p_{\min} = \min_{L \in \mathcal{P}} p_L, \tag{13}$$

and the episode-level efficiency reward averages over the active plan:

$$R_{\text{eff}} = \begin{cases} \frac{1}{B} \sum_{k=1}^{B} \eta_k & B > 0, \\ 0 & B = 0. \end{cases} \tag{14}$$

By construction $\eta_k \in [0, 1]$ and $R_{\text{eff}} \in [0, 1]$. Unit credit arises from either a router-only commitment or an informative commitment using a cheapest-tier model. Multiplication by c_{ans} in Eq. (4) ensures cheap-but-wrong trajectories receive no efficiency credit.

Failure-aware adjustment (full form). For each subquestion q_k, let $\iota_{k,1}$ be the informativeness of the *first* invoked model on q_k. Let $\mathcal{K}_{\text{ext}}$ contain only indices of committed subquestions with at least one external invocation. First/last model prices are evaluated only on this set. For each k, let $\mathcal{R}_k$ be the maximal consecutive runs of uninformative responses from the same model within e_k. An informative response or a model switch ends such a run. We define

$$E_{\text{esc}} = \sum_{k \in \mathcal{K}_{\text{ext}}} \mathbf{1}\left[\iota_{k,1} = 0 \ \wedge \ p_{L_k^\star} > p_{L_{k,1}} \ \wedge \ z_k = \text{DONE} \ \wedge \ \iota_k = 1\right], \tag{15}$$

$$E_{\text{rep}} = \sum_{k=1}^{B} \sum_{r \in \mathcal{R}_k} \max\{0, |r| - 1\}, \tag{16}$$

where $L_{k,1}$ is the first model invoked on q_k and $|r|$ is the number of calls in failure run r. A later return to the same model starts a new run; failed calls from different runs are not merged. The adjustment is

$$R_{\text{adj}} = c_{\text{ans}} \, \lambda_{\text{esc}} \, E_{\text{esc}} - \lambda_{\text{rep}} \, E_{\text{rep}}. \tag{17}$$

The first term rewards *conditional escalation*—an initial cheap attempt that genuinely failed, followed by a more expensive informative response—while the second term penalizes consecutive uninformative calls to the same model on one subquestion. The first failed call in each consecutive run is unpenalized, leaving room for an initial attempt with a cheap model. This repetition term alone does not penalize alternating failed calls to different models; the turn budget still bounds the trajectory.

Composite reward and reporting cost. The full trajectory-level reward is Eq. (4) (boxed in the main text): $r_\phi(x, y) = R_{\text{out}} + \lambda_{\text{fmt}} \bar{R}_{\text{fmt}} + \lambda_{\text{dec}} R_{\text{dec}} + c_{\text{ans}} \lambda_{\text{eff}} R_{\text{eff}} + R_{\text{adj}}$. For evaluation we report the unmodified routing cost of (Zhang et al., 2025a):

$$C_{\text{eval}}(y) = \sum_{j \in \mathcal{I}(y)} p_{L_j} \, T_j^{\text{out}}, \tag{18}$$

where $\mathcal{I}(y)$ indexes external invocations with recorded output tokens, T_j^{out} is their output token count, and p_L is the fixed price in USD per million output tokens. Thus C_{eval} is reported in micro-USD-equivalent proxy units; its dollar value is $C_{\text{eval}}/10^6$. Appendix A.14 lists the price mapping and exclusions. C_{eval} does not appear in r_ϕ; the cost-side training signal is entirely carried by R_{eff} and R_{adj}.

Hyperparameters. Table 5 lists the main reward configuration. We selected weights through staged manual calibration during pilot training and then froze them across datasets and model pools. The component ablations support their intended roles; they are not a systematic coefficient sensitivity sweep. The separate planning comparison uses the explicitly reported $T = 8$ budget.

Table 5: Reward-component hyperparameters. The structural term is bounded and efficiency credit is gated by exact correctness.

Symbol	Value	Role
γ	0.2	F1 shaping cap in R_{out}
δ	0.4	F1 deadband threshold in R_{out}
λ_{fmt}	1.0	Format penalty weight
λ_{dec}	0.5	Decomposition reward weight
λ_{eff}	0.3	Efficiency reward weight (gated by c_{ans})
λ_{str}	0.1	Structural validity coefficient inside R_{dec}
λ_{sub}	0.05	Per-commit credit inside R_{dec}
λ_{done}	0.1	Plan completion bonus inside R_{dec}
λ_{util}	0.2	Plan-utility weight inside R_{dec}
λ_{early}	0.3	Early-termination penalty inside R_{dec}
$r_{\min}, r_{\max}$	$-0.4, 0.4$	Clip bounds on R_{dec}
λ_{esc}	0.05	Conditional-escalation bonus weight
λ_{rep}	0.1	Repeated-failure penalty weight
$B_{\max}$	4	Maximum plan width
T	6	Maximum interaction turns

Informativeness detector. The indicator $\iota_k \in \{0, 1\}$ is produced by a fixed deterministic detector that takes the raw text o_L returned by the resolving model and returns 0 (uninformative) if any of the following patterns match (case-insensitive, after light whitespace normalization):

- Empty or whitespace-only response, or response shorter than 4 characters.
- Refusal patterns: "*i cannot*", "*i can't*", "*cannot answer*", "*unable to*", "*i won't*".
- Inability verbalizations: "*unknown*", "*not enough information*", "*insufficient context*", "*i don't know*", "*no information*", "*cannot determine*".
- Apology-only responses ("*sorry, ...*") with no factual content following the apology.

Otherwise $\iota_k = 1$. The detector requires no oracle decomposition or allocation labels and is shared by Eqs. (12), (13), and (17). It is an informativeness heuristic, not a factuality verifier: incorrect content may pass this filter. The commitment-level audit and its conditional denominator are reported in Appendix A.12.

How the reward discourages unnecessary decomposition. An empty or refusal response receives no informativeness-based credit, repeated failed calls are penalized, and KL regularization constrains deviation from the reference policy. However, Eq. (14) contains no explicit width penalty, and a redundant but informative subquestion can still receive credit. These mechanisms are incentives, not a guarantee of minimal plans or correct local answers. Appendix A.12 reports the observed frequency of redundant and harmful decomposition.

Why a ratio rather than absolute cost. We use the price ratio $p_{\min}/p_{L_k^\star} \in [0, 1]$ in Eq. (13) rather than the absolute monetary cost $p_{L_k^\star} T_j^{\text{out}}$. An absolute-cost shaping term scales linearly with response length; a single verbose answer can swamp the gradient and destabilize PPO updates across heterogeneous tiers (output token counts span roughly 10–1000 across our pool). The ratio formulation isolates the allocation choice from output verbosity. The reported routing cost $C_{\text{eval}}(y)$ in Eq. (18) is preserved unmodified as the evaluation metric, so the training signal and the evaluation metric are aligned in *direction* (cheaper resolving models are preferred by both) without requiring identical functional form.

Why outcome gating. Multiplying R_{eff} and the escalation bonus by c_{ans} ensures that no efficiency credit is granted for cheap-but-wrong trajectories. Without this gate, an incorrect trajectory can still earn positive efficiency credit from cheap informative external calls. A direct answer with $B = 0$ has $R_{\text{eff}} = 0$ under Eq. (14); it does not maximize this term. The ablation in Section 5.3 ("w/o outcome-gated composition") shows exactly this collapse: cost falls to 82.5 but EM drops to 0.272.

A.2.3 ACTION-SEGMENT PPO AND TOKEN-LEVEL IMPLEMENTATION

This section gives the full PPO objective with token-level masking, the value loss, and the GAE advantage estimator referenced in Section 3.

Action-likelihood at the segment level. Let $a_t = (y_{m_t}, \ldots, y_{n_t})$ be the router-generated decision segment at step t (private THINK tokens plus the executable tag). Define the segment indicator $\mathbf{u}_i = \mathbf{1}[\text{token } y_i \text{ is part of a router-generated segment}]$. The masked segment log-probability is

$$\log \pi_\theta(a_t|s_t) = \sum_{i \in [m_t, n_t]} \mathbf{u}_i \log \pi_\theta(y_i|s_t, y_{m_t:i-1}), \tag{19}$$

Within a router segment $\mathbf{u}_i = 1$; the mask matters when assembling the full interleaved trajectory loss. Prior environment observations are contained in s_t and condition subsequent decisions, but are not themselves scored as router-generated actions.

Segment-level PPO objective.

$$\rho_t(\theta) = \exp[\log \pi_\theta(a_t|s_t) - \log \pi_{\theta_{\text{old}}}(a_t|s_t)], \tag{20}$$

With the importance ratio $\rho_t(\theta)$ from Eq. (20) and the advantage estimate $\hat{A}_t$ defined below, the clipped surrogate is

$$\mathcal{L}_\pi(\theta) = -\mathbb{E}_t\Big[\min\Big(\rho_t(\theta)\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big] + \beta\, \mathbb{E}_t\, \text{KL}(\pi_\theta(\cdot|s_t) \,\|\, \pi_{\text{ref}}(\cdot|s_t))\,, \tag{21}$$

The expectation is taken only over router *decision* states s_t, never over indices that fall inside an environment observation.

Value loss and return-to-go. The value head V_ψ is trained on the same router decision states:

$$\mathcal{L}_V(\psi) = \mathbb{E}_t\Big[\big(V_\psi(s_t) - \hat{R}_t\big)^2\Big], \quad \hat{R}_t = \sum_{u=t}^{T_y} \gamma_{\text{ppo}}^{u-t}\, r_u, \tag{22}$$

where r_u is the reward attributable to decision step u and $\gamma_{\text{ppo}} = 1$ throughout (rewards are delivered at episode end, so undiscounted return-to-go suffices). The total training loss is $\mathcal{L} = \mathcal{L}_\pi + c_v\, \mathcal{L}_V$ with $c_v = 0.5$.

Advantage estimation (GAE). We use Generalized Advantage Estimation over decision states:

$$\hat{A}_t = \sum_{l=0}^{T_y - t} (\gamma_{\text{ppo}}\lambda_{\text{gae}})^l\, \delta_{t+l}, \qquad \delta_u = r_u + \gamma_{\text{ppo}} V_\psi(s_{u+1}) - V_\psi(s_u), \tag{23}$$

with $\lambda_{\text{gae}} = 0.95$ and $V_\psi(s_{T_y+1}) = 0$ at true episode termination. Value estimates used in constructing returns and advantages are held fixed during an update. Advantages are standardized per minibatch in this decision-level description.

Why action-only scoring matters. Conditional on the emitted action and prior state, the external-response distribution does not depend on router parameters. Its own conditional log-likelihood therefore has exactly zero derivative with respect to θ. Scoring an external response under the router distribution instead would introduce an unrelated, generally biased imitation objective. Observations remain in the context, so earlier router actions can still receive credit through their effect on later states and rewards. The action-only-scoring ablation in Table 3 yields EM 0.349.

Segment and token clipping are distinct. The chain-rule identity for segment log-probability does not make clipping a whole segment ratio equivalent to clipping each token ratio. For a router token with prefix h_i, let $r_i(\theta) = \pi_\theta(y_i \mid h_i)/\pi_{\theta_{\text{old}}}(y_i \mid h_i)$. Then $\rho_t = \prod_{i=m_t}^{n_t} r_i$, whereas a token-clipped surrogate averages $\min\{r_i\hat{A}_i, \text{clip}(r_i, 1-\epsilon, 1+\epsilon)\hat{A}_i\}$ over router tokens. These are different objectives. Eq. (21) states the decision-segment formulation, not an algebraic equivalence to token-wise clipping.

Table 6: PPO hyperparameters.

Hyperparameter	Value
Actor learning rate	1×10^{-6}
Critic learning rate	1×10^{-5}
KL coefficient β	1×10^{-3}
PPO clip ratio ϵ	0.2
GAE λ_{gae}	0.95
Discount γ_{ppo}	1.0
Value loss weight c_v	0.5
Maximum interaction turns T	6
Plan width bound $B_{\max}$	4
Rollouts per update	256 trajectories
Minibatch size	32 trajectories
PPO epochs per update	2

Training hyperparameters. Table 6 lists the main decision-level training configuration stated in the manuscript; the matched recursive-planning comparison uses $T = 8$ (Appendix A.9).

A.2.4 Capability-Cost Profile Construction

The CCP is an optional inference-time hint distilled from past invocation logs. Deployment profiling can additionally accumulate completed calls from a continuing request stream. It introduces no learnable parameters, no constraint on the action space, and does not appear in any training objective.

Logged tuples. During training the environment passively records, for each external invocation, the tuple (q, L, o_L, c_L, ι): the routed (sub)question q, the invoked model L, the response o_L, the call cost c_L, and the informativeness indicator ι from Appendix A.2.2. No gold labels are used.

Subquestion category clustering. At distillation time we project each logged q into a coarse category c via either a fixed lightweight classifier (e.g., a 5–10 way taxonomy: *single-fact lookup, biographical, date/time, numerical comparison, multi-entity relation*, etc.) or k-means on sentence embeddings. The choice of classifier does not affect the training procedure since the CCP is built post hoc.

Smoothed capability-cost score. For each (model, category) pair we compute a Beta-smoothed informativeness rate weighted by the inverse price ratio:

$$\widehat{S}(L, c) = \frac{\alpha + \sum_i \mathbf{1}[q_i \in c,\, L_i = L]\, \iota_i}{2\alpha + \sum_i \mathbf{1}[q_i \in c,\, L_i = L]} \cdot \frac{p_{\min}}{p_L}, \tag{24}$$

with $\alpha = 1$ (uninformative prior). The first factor is the Beta-posterior mean of the informativeness rate of L on category c; the second factor expresses cost-effectiveness in the same units as η_k (Eq. (13)). $\widehat{S} \in [0, 1]$.

Verbalization. For each category c we list the top-k models (positive side, "empirically cost-effective on c") and the bottom-k models ("lower empirical capability–cost score on c") by $\widehat{S}(L, c)$, with $k = 2$ in our experiments. The lists are concatenated into a short natural-language hint that is prepended to each model's descriptor at evaluation time. Because the hint is purely additive context, the same trained checkpoint can be evaluated with or without CCP; all main results use CCP off. The separate pool-shift case study is in Appendix A.13. A low composite score alone does not imply repeated failure, since the score also penalizes price. The informativeness rate is not a correctness estimate, and sparse invocation history limits interpretation of a candidate's profile.

A.3 Training Prompts

This section presents the prompt used to elicit the granularity-aware routing behavior from the 3B router policy π_θ at both training and inference time. The same template `PROMPT_TEMPLATE_QWEN`

(also used as `PROMPT_TEMPLATE_LLAMA` in the codebase) is shared between training rollouts and evaluation; only the `{question}` slot at the bottom is filled per instance. The prompt encodes (i) the action grammar of Appendix A.2.1, (ii) a strict linear execution plan that matches the one-shot decomposition guard, (iii) two worked examples (a decomposition trajectory and a direct-answer trajectory) that anchor the format, and (iv) the candidate-model descriptors $\mathrm{Desc}(\mathcal{P})$ that the router uses to make capability–cost trade-offs.

Full prompt template.

```
Answer the given question.
## ROLE
You are a granularity-aware QA router. For each question, decide whether
↪  to
(a) answer directly,
(b) route the whole question to a single model, or
(c) decompose the question into ordered subquestions and route each
↪  subquestion
    to a model.
You may interleave private before every other action.
1. **Think**: Use <think> ... </think> to reason about the next step.
2. **Decompose**: Use <decompose> ... </decompose> at most once, with 2
↪  or 3
   ordered sub-questions.
3. **Search**: Use <search> ... </search> to query an external LLM and
↪  receive
   one matching <information> ... </information>.
4. **Subanswer**: Use <subanswer> ... </subanswer> only in decompose mode
↪  to
   mark the current TODO as solved. The content inside <subanswer> must
   ↪  be a
   short extracted answer span, not a long explanatory sentence.
5. **Answer**: Use <answer> ... </answer> only for the final answer to
↪  the
   original question. The content inside <answer> must be a short
   ↪  extracted
   answer span, not a long explanatory sentence.

## STRICT RULES
You must follow every rule below exactly. If a later action would violate
↪  one
of these rules, rewrite the action before responding.

Linear execution plan means:
- the two sub-questions must be solved in a single chosen order: SubQ1,
↪  then
  SubQ2
- each later sub-question depends only on the original question or on
↪  earlier
  solved sub-questions, never on a future unanswered sub-question
- while SubQ1 is still TODO, do not search or answer SubQ2
- if the problem cannot be cleanly solved in this sequential order, do
↪  NOT use
  <decompose>; use direct reasoning, <search>, and <answer> instead

## Example (decomposition trajectory)
<think>
The question requires the director's nationality, which decomposes into
(i) identifying the director, then (ii) recovering their nationality.
</think>
<decompose>
- Who directed the film "Amelie"?
- What is the nationality of this director?
</decompose>

The system then shows:
```

```
<decomposition_state>
[SubQ1][TODO] Who directed the film "Amelie"?
[SubQ2][TODO] What is the nationality of this director?
</decomposition_state>

<think>
SubQ1 is a foreign-language-title fact lookup; route to a 70B model
to maximize informativeness on this proper noun.
</think>
<search>[SubQ1] LLaMA-3.1-70B-Instruct: Who directed the film
↪  Amelie?</search>

<information>[SubQ1] The film "Amelie" (2001) was directed by Jean-Pierre
Jeunet.</information>

<think>
The information directly identifies the director; commit it as SubQ1's
↪  answer.
</think>
<subanswer>[SubQ1] Jean-Pierre Jeunet</subanswer>

The system then shows:
<decomposition_state>
[SubQ1][DONE] Who directed the film "Amelie"? => Jean-Pierre Jeunet
[SubQ2][TODO] What is the nationality of this director?
</decomposition_state>

<think>
Jean-Pierre Jeunet is a well-known French director; this is recoverable
↪  from
my own parametric knowledge without an external search.
</think>
<subanswer>[SubQ2] French</subanswer>

The system then shows:
<decomposition_state>
[SubQ1][DONE] Who directed the film "Amelie"? => Jean-Pierre Jeunet
[SubQ2][DONE] What is the nationality of this director? => French
</decomposition_state>

<think>
Both subquestions are resolved; the director's nationality is French.
</think>
<answer> French </answer>

## Another Example (simple question, no decomposition needed)
Question: What is the capital of France?

<think>
This is a single well-known fact; answer directly without decomposition
↪  or
external search.
</think>
<answer> Paris </answer>

#### LLM Descriptions

Qwen2.5-7B-Instruct:
Strong general-purpose model for language, reasoning, coding, and
↪  multilingual
tasks. Good default choice for concise factual queries.

LLaMA-3.1-8B-Instruct:
Efficient general-purpose multilingual model. Useful for lower-cost
↪  factual
lookup and straightforward reasoning.
```

```
LLaMA-3.1-70B-Instruct:
Best choice for the hardest reasoning, ambiguous questions, and
↪  high-precision
general knowledge.

Mistral-7B-Instruct:
Lightweight instruction model. Useful for simple retrieval and short-form
reasoning.

Mixtral-8x22B-Instruct:
High-capacity model with strong multilingual, long-context, and reasoning
performance. Use when you need more depth than 7B or 8B models.

Gemma-2-27B-Instruct:
Strong mid-sized model for question answering, summarization, and
↪  reasoning.
Good fallback when you want quality without always using 70B.

If no further external knowledge is needed, directly provide the final
↪  answer
inside <answer> ... </answer>.
The final answer should be the minimal extracted answer span whenever
↪  possible,
not a full sentence. For example: <answer> Beijing </answer>.
Do not output placeholder text like "<answer> and </answer>" by itself.

## Now answer the following question:
Question: {question}
```

How the prompt encodes the methodology. Several design choices in the prompt directly correspond to formal constructs in Section 3:

- The "`Always start with <think> ...`" instruction enforces the think-first ordering described in Appendix A.2.1: every router-emitted executable action is preceded by a private `<think>` span. Format violations are penalized by R_{fmt} (Eq. (9)).
- "Use `<decompose>` at most once, with 2 or 3 ordered sub-questions" implements the one-shot adaptive decomposition guard with the soft width recommendation $B \in \{2, 3\}$, well below the hard cap $B_{\max} = 4$ in Table 5.
- The "Linear execution plan" rules implement Algorithm 1: subquestions are resolved in generation order, and sub-question dependencies are forward-only.
- "If the problem cannot be cleanly solved in this sequential order, do NOT use `<decompose>`" explicitly steers the policy toward whole-query routing or direct answering when the question is not cleanly decomposable. Together with R_{dec} and R_{adj}, this discourages gratuitous decomposition; the behavioral statistics (12.7–81.6% decomposition rate, Section 5.4) show the policy learns to gate `<decompose>` on input difficulty.
- "The content inside `<subanswer>` / `<answer>` must be a short extracted answer span" aligns the policy's output format with the EM metric used in R_{out} (Eq. (8)): long explanatory sentences would tank EM and the gate c_{ans} would zero out the efficiency reward R_{eff}.
- The two worked examples (Amélie, Paris) illustrate two of the three operating modes in Table 4: decomposition routing with router self-integration on SubQ2 and direct answering. Whole-query routing emerges naturally when the policy emits a single `<search>` without a preceding `<decompose>`; the environment then instantiates the singleton plan $q_1 = x$ (Section 3).
- The six LLM descriptions form $\text{Desc}(\mathcal{P})$. They are intentionally written as short capability–cost notes rather than benchmark scores so the router learns capability boundaries from interaction (the R_{eff} and R_{adj} shaping in Appendix A.2.2) rather than from supervised labels. At inference time, the optional Capability-Cost Profile hint (Appendix A.2.4) may be appended to each descriptor when explicitly enabled; all main experiments keep CCP off.

Reward-side parsing. The reward computer is given the gold answer set $\mathcal{Y}^\star$ and the trajectory y; it parses the executable tags, applies the grammar guards (Appendix A.2.1), and computes Eqs. (8)–(17). The informativeness detector (Appendix A.2.2) is applied directly to each `<information>` block returned by an invoked model.

Reproducibility. The verbatim prompt above is the constant `PROMPT_TEMPLATE_QWEN` in the released codebase; `PROMPT_TEMPLATE_LLAMA` is set to the same value, so all runs reported in Section 5 share this prompt. The candidate-model descriptors are likewise constant across training and evaluation; only the `{question}` slot varies per instance. License and release details follow Appendix A.5.

A.4 Routing Case Studies

This section presents qualitative trajectories that illustrate the three operating modes (Table 4) and the failure-aware adjustment in Appendix A.2.2. All trajectories are taken from the evaluation logs on out-of-domain benchmarks; gold answers are shown for verification.

Case 1: Direct answer on a simple factual question. On a simple single-fact input covered by its parametric knowledge, the router can answer without invoking any external model. The percentages in Section 5.4 measure decomposition frequency, not direct-answer frequency.

> **Question.** "What is the capital of France?"
> **Trajectory.** `<think>` Single fact within router's parametric knowledge. `</think>` `<answer>` Paris `</answer>`
> **Plan width.** $B = 0$. **External calls.** 0. **Cost.** 0. **Correct.** ✓

Case 2: Whole-query routing to a 7B model. For a single-hop knowledge question outside the router's parametric coverage, the router skips DECOMPOSE and routes the whole query to a cheap model.

> **Question.** "Who directed the 1995 film *Heat*?"
> **Trajectory.** `<think>` Single fact, factual lookup; route whole query to a 7B model. `</think>` `<search>` [SubQ1] Qwen2.5-7B: Who directed the 1995 film *Heat*? `</search>` *(observation: "Michael Mann")* `<subanswer>` [SubQ1] Michael Mann `</subanswer>` `<answer>` Michael Mann `</answer>`
> **Plan width.** $B = 1$ (auto singleton). **External calls.** 1, all 7B-tier. **Correct.** ✓

Case 3: Decomposition with router self-integration (a real trajectory). Decomposition rates are 73.2% on 2Wiki and 81.6% on Bamboogle; these rates do not measure the frequency of router self-resolution. In this example, the router decomposes a compositional question, routes one subquestion externally for a fact lookup, and resolves the second subquestion from its own parametric knowledge—demonstrating the trajectory-aware integration noted in Section 1.

> **Question.** "What is the nationality of the director of the film *Amélie*?"

Action and observation excerpt (private reasoning spans omitted):

```
<decompose>
- Who directed the film "Amelie"?
- What is the nationality of this director?
</decompose>

<decomposition_state>
[SubQ1][TODO] Who directed the film "Amelie"?
[SubQ2][TODO] What is the nationality of this director?
</decomposition_state>

<search>[SubQ1] LLaMA-3.1-70B-Instruct: Who directed the film
↪  Amelie?</search>
<information>[SubQ1] The film "Amelie" (2001) was directed by Jean-Pierre
↪  Jeunet.
```

```
</information>

<subanswer>[SubQ1] Jean-Pierre Jeunet</subanswer>

<decomposition_state>
[SubQ1][DONE] Who directed the film "Amelie"? => Jean-Pierre Jeunet
[SubQ2][TODO] What is the nationality of this director?
</decomposition_state>

<subanswer>[SubQ2] French</subanswer>

<decomposition_state>
[SubQ1][DONE] Who directed the film "Amelie"? => Jean-Pierre Jeunet
[SubQ2][DONE] What is the nationality of this director? => French
</decomposition_state>

<answer> French </answer>
```

Plan width. $B = 2$. **External calls.** 1 ($L_1^\star =$ LLaMA-3.1-70B resolves SubQ1; SubQ2 is committed without external invocation, $L_2^\star = \varnothing$). **Correct.** ✓
Why illustrative. (i) The router does *not* mechanically search every subquestion—once the director's identity is in the transcript, the nationality is recoverable from the router's own parametric knowledge, so SubQ2 is committed directly. This is precisely the *trajectory-aware integration* highlighted in Section 1, where the router closes the loop instead of handing aggregation to another model. (ii) For SubQ1, the router selected the 70B-tier model to maximize informativeness on a fact lookup involving a foreign-language title; the 10.2% 70B-tier allocation in Section 5.4 is composed of trajectories of this kind. (iii) SubQ2 incurs zero external-call cost and receives $\eta_2 = 1$ from the explicit self-resolution branch of Eq. (13). No price ratio is evaluated for the absent external model. This credit is averaged over the plan and gated by exact final correctness.

Case 4: Conditional escalation rescues a failed cheap attempt. This trajectory illustrates the E_{esc} term in Eq. (17): an initial 7B call returns an uninformative response, and the router escalates to a 27B model that produces an informative subanswer.

Question. (MuSiQue, multi-hop with rare entity) "..."
Trajectory excerpt. `<search>` [SubQ2] Mistral-7B: ... `</search>` *("I don't know.")* $\Rightarrow \iota_{2,1} = 0$ `<search>` [SubQ2] Gemma-2-27B: ... `</search>` *(informative response)* $\Rightarrow \iota_2 = 1$ `<subanswer>` [SubQ2] ... `</subanswer>`
Reward annotation. E_{esc} contributes $+1$ (cheap-first uninformative, expensive-second informative, $z_2 = \text{DONE}$, $\iota_2 = 1$); its reward contribution is also gated by final-answer correctness. The first failed call incurs no E_{rep} penalty—only repeat invocations on the *same* model after an uninformative response do.
Why illustrative. This shows conditional escalation from 7B to 27B. A single trajectory does not establish the aggregate frequency of escalation to any model tier.

A.5 Experiment Implementation and Reproducibility Details

This section expands the condensed Section 4 with the full descriptions of benchmarks, baselines, candidate pools, and cost measurement.

Datasets and metrics. We evaluate Granule-R1 on seven question-answering benchmarks spanning two regimes. The *general QA* suite consists of Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023), in which most questions are answerable from a single fact or short reasoning chain. The *multi-hop QA* suite consists of HotpotQA (HpQA) (Yang et al., 2018), 2WikiMultiHopQA (2Wiki) (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Bamb) (Press et al., 2023), which require composing multiple facts or comparing entities. Together they exercise both the simple-input regime, where the router should answer directly or use a single model call, and the compositional regime, where decomposition becomes profitable. We report Exact Match (EM) and token-level F1, evaluating on 500 randomly sampled instances per dataset; Bamboogle uses its full 125-instance test set. Training mixes NQ and HotpotQA only, so NQ and HpQA serve as in-domain evaluations and the remaining five benchmarks

(TriviaQA, PopQA, 2Wiki, MuSiQue, Bamboogle) are out-of-domain transfer tests for the same trained checkpoint.

Baselines. We compare Granule-R1 against three categories of baselines under the same Full Pool, training data, and evaluation splits.

- **Single-model and prompting baselines** include *Direct prompting*, which queries the policy LLM with the raw question; *Chain-of-Thought (CoT) prompting* (Wei et al., 2022), which elicits intermediate reasoning before the answer; *supervised fine-tuning (SFT)* on the training mixture; *Retrieval-Augmented Generation (RAG)* (Lewis et al., 2020) using the Wikipedia-18 dump as the corpus and E5 (Wang et al., 2022) as the dense retriever; and *Search-R1* (Jin et al., 2025), which trains the policy to interleave reasoning with search-engine calls via reinforcement learning. These baselines capture the within-model frontier without external model selection.
- **Query-level routers** include *Prompt LLM*, which prompts the base model to select a candidate from $\mathcal{P}$ given each query; *Largest LLM*, which deterministically selects the largest available model; *RouterDC* (Chen et al., 2024), which trains a query-to-model classifier with dual contrastive learning; and *GraphRouter* (Feng et al., 2024), which models task–query–LLM interactions on a heterogeneous graph. All four route the original query as the atomic decision unit.
- **Multi-round routers** are represented by *Router-R1* (Zhang et al., 2025a), which interleaves internal reasoning with external model invocations through reinforcement learning but treats the original query as the atomic routing unit (no decomposition action). This is the strongest comparable router and the primary head-to-head baseline for Granule-R1.

The Qwen2.5-3B-Instruct initialization is shared where a policy LLM is used. Reported Router-R1 results, independent verification, and newly evaluated fixed-checkpoint comparisons are distinguished in Appendix A.6; different baseline classes do not all share the same training procedure or evidence source. The main tables retain their original single-run protocol, with the final five-seed analysis reported separately in Appendix A.8.

Candidate pools. For the main results in Section 5.1, all methods access a heterogeneous pool of six models spanning a wide capability–cost spectrum: Qwen2.5-7B-Instruct (Yang et al., 2024), LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Mistral-7B-Instruct (Jiang et al., 2023), Gemma-2-27B-Instruct (Gemma Team et al., 2024), Mixtral-8x22B-Instruct (Jiang et al., 2024), and LLaMA-3.1-70B-Instruct (Dubey et al., 2024); we refer to this as *Full Pool*. For the granularity analysis in Section 5.2, we additionally evaluate Granule-R1 with a restricted *Small Pool* containing only the three 7B-tier models (Qwen2.5-7B, LLaMA-3.1-8B, Mistral-7B), excluding all 27B/70B-class models from access. Crucially, the same trained checkpoint is used in both pool settings; only the descriptor file fed to the router at inference time changes, exploiting the descriptor-based extensibility of the framework (Section 3). This isolates the effect of pool composition from any training-time bias toward specific candidate models.

Cost measurement. We adopt the external-output-token proxy in Eq. (18), using the same fixed pricing vector as Router-R1 for controlled comparison. The coefficient p_L is in USD per million output tokens, so a reported value of 110.6 corresponds to USD 0.0001106 per query for the metered external outputs. This is not a full serving bill and is not invariant to price changes. The numerical mapping, conversion, and exclusions are in Appendix A.14; training and controller overhead are reported separately in Appendix A.11. The ratio-based R_{eff} and failure-aware R_{adj} supply the stated cost-side training signal, while C_{eval} is the evaluation metric.

Single unified configuration. **Granule-R1 is trained as a single, unified configuration**: we do not introduce additional cost-control hyperparameters at inference time. The composite reward in Eq. (4) is used at training time only; at evaluation, the policy emits one trajectory per input and we record both EM/F1 and the resulting routing cost. There is no inference-time knob to retune for different cost budgets—the entire accuracy–cost trade-off in Figure 3 of the main text is realized by a single trained policy.

Cost is reduced even as accuracy improves. Table 7 reports the routing cost incurred to achieve the EM scores above on the four cost-evaluated datasets (NQ, PopQA, HpQA, 2Wiki). Although Granule-R1 has access to the same Full Pool as every other router, it spends substantially less than the strongest whole-query router: the average cost falls from 134.6 (Router-R1-Qwen) to 110.6 (Granule-R1), a $\mathbf{17.8\%}$ reduction. This is achieved while improving accuracy on all four datasets simultaneously. The combined accuracy gain and cost reduction shows that Granule-R1 does not merely trade one for the other—it strictly Pareto-dominates the prior multi-round router, without requiring any cost-control hyperparameter at evaluation.

Table 7: Routing cost on the four cost-evaluated datasets, evaluated under the Full Pool. Granule-R1 simultaneously improves accuracy and reduces cost relative to the strongest whole-query routing baseline.

Method	**NQ**† Cost↓	**PopQA** Cost↓	**HpQA**† Cost↓	**2Wiki** Cost↓	**Avg. EM↑**	**Avg. Cost↓**
Prompt LLM	20.0	17.6	20.1	20.2	0.293	19.5
Largest LLM	20.2	17.6	20.1	20.1	0.301	19.5
GraphRouter	29.6	19.2	24.7	28.6	0.243	25.5
RouterDC	57.5	21.8	62.3	40.7	0.256	45.6
Router-R1-Qwen	150.6	98.3	138.6	150.8	0.390	134.6
Granule-R1	**122.4**	**75.7**	**120.3**	**124.2**	**0.423**	**110.6**

The default software stack uses Python 3.9, PyTorch 2.4.0, vLLM 0.6.3, FlashAttention 2, and veRL. Training data is constructed by mixing 7K examples from NQ with 7K examples from HotpotQA, while evaluation covers seven QA benchmarks.

A.6 Baseline Provenance and Comparison Protocols

The Router-R1-Qwen entries in the original main-result and small-pool comparisons are the published values from Zhang et al. (2025a). Independent verification with the authors' public repository and released evaluation configuration yielded average EM 0.414 ± 0.003, compared with the published 0.416, with a maximum dataset-level deviation of 0.012. These verification runs are distinct from the five-seed paired analysis below, which evaluates one fixed Router-R1-Qwen checkpoint on exactly the same examples as each Granule-R1 checkpoint.

Router-R1-Qwen and Granule-R1 use the same 7K NQ + 7K HotpotQA training mixture, Qwen2.5-3B initialization, candidate pool, and evaluation splits. RouterDC uses its public checkpoint with matching training data and pool. GraphRouter is independently trained using per-candidate EM supervision on the same mixture, the official two-layer heterogeneous GNN, and the official hyperparameter configuration; checkpoint selection uses the shared validation set. Dense supervision for 14K queries and six candidates requires 84K candidate evaluations. This is an offline data-construction expense, distinct from online router inference.

Removing the decomposition action while retaining the other Granule-R1 training settings is the direct mechanism ablation. Comparisons against systems with different evidence sources or training procedures do not isolate the effect of structured decomposition by themselves. In particular, the Search-R1-MultiTool comparison in Appendix A.10 is a zero-shot backend adaptation, not a generic planner retrained under fully matched PPO conditions.

A.7 Ablation Supplement

Motivation and comparison scope. R2-Reasoner (Shao et al., 2026) follows a predefined decomposition-first pipeline, in which question decomposition precedes subtask-level model allocation and sequential execution. Granule-R1 instead adaptively determines whether and how to decompose a question, making the comparison relevant to assessing adaptive granularity management against a fixed decomposition-first workflow. Because the evaluated systems differ in controller architecture, training configuration, and model pool, the present evaluation does not constitute a fully matched baseline comparison, and we cannot directly adopt the comparative results in the main

experiment. We therefore report the R2 transfer experiment in the appendix as a supplementary evaluation that complements our ablation study.

R2-Reasoner uses separate Qwen2.5-7B-Instruct-based models for task decomposition and subtask allocation, whereas Granule-R1 uses a single Qwen2.5-3B-Instruct policy. Its execution pipeline additionally separates subtask execution, answer review, and final synthesis. We preserve this modular structure when evaluating the released checkpoints. The differences in controller capacity, architecture, and training data make this a comparison of transfer configurations.

Transfer implementation. We reconstruct each CSQA controller by sequentially merging its released SFT and RL adapters into Qwen2.5-7B-Instruct. All expert models are accessed through APIs. The adapted candidate pool contains Qwen2.5-7B-Instruct, Llama-3.1-8B-Instruct, Mistral NeMo, Gemma-2-27B-Instruct, and Llama-3.1-70B-Instruct.

Evaluation protocol. The evaluation uses 500 questions from each benchmark except Bamboogle, for which all 125 questions are used. We reuse the existing frozen NQ and HotpotQA evaluation subsets and independently sample 500 examples from each of TriviaQA, PopQA, 2WikiMultiHopQA, and MuSiQue using seed 42. The newly loaded datasets use the FlashRAG dataset revision `bcafb8dd07d453be3cbeeeb3f78be1841bddf92c`. Inference receives only the question, without retrieved documents or benchmark-provided context. We use inference seed 42 and evaluate normalized EM and token-level F1 over the available answer aliases. All 3,125 questions complete inference after retrying API failures.

Table 8 places these transfer results alongside the Granule-R1 results reported in Table 1.

Table 8: R2-Reasoner zero-shot transfer to seven QA benchmarks. Scores are percentages. Granule-R1 values are reproduced from Table 1;

	Granule-R1 (reference)		R2-Reasoner (transfer)	
Dataset	EM	F1	EM	F1
NQ	43.20	50.70	25.80	35.68
TriviaQA	75.20	80.60	58.80	66.93
PopQA	42.80	46.00	20.00	26.67
HotpotQA	38.40	46.20	20.00	30.35
2WikiMultiHopQA	45.40	51.50	20.20	25.85
MuSiQue	15.80	23.80	5.40	12.31
Bamboogle	54.40	66.20	24.00	37.18

Interpretation. The transferred R2-Reasoner configuration obtains lower EM and F1 than the Granule-R1 reference on all seven benchmarks, despite using larger controller models. This observation shows that larger controllers and a separate decomposition-first pipeline do not automatically yield stronger performance in the evaluated transfer setting. The comparison complements our internal ablations and input-dependent decomposition statistics, which support the value of learning granularity selection together with model allocation.

A.8 Five-Seed Statistical Evaluation

The original main tables report single-run point estimates on 500 examples per dataset, except Bamboogle, which uses its full 125-question test set. For the supplementary evaluation, we train Granule-R1 with five independent PPO seeds and compare every checkpoint with the same fixed Router-R1-Qwen checkpoint. Evaluation is expanded to 2,000 held-out examples per dataset wherever the split permits; the original 500-example sets are subsets of the expanded sets, and Bamboogle retains its full test set. Decoding remains greedy (`do_sample=False`), so decoding seeds are not independent experimental runs.

Let Δ denote Granule-R1 minus Router-R1-Qwen. The reported SD is the spread of Δ across Granule-R1 training seeds. The 95% confidence interval for the mean Δ uses 10,000 hierarchical paired-bootstrap resamples: training seeds are resampled at the top level, with the same resampled seed indices applied to all datasets; example indices are then resampled within each dataset, pairing

both methods on the same questions. Aggregates are unweighted macro-averages over datasets. This preserves training-seed dependence across datasets while avoiding implicit reweighting by evaluation-set size. The intervals describe uncertainty conditional on one fixed baseline checkpoint, not variation across Router-R1 training runs.

Table 9: Final five-seed evaluation. Differences are Granule-R1 minus Router-R1-Qwen. The five individually listed datasets each use 2,000 examples. Four-set aggregates use NQ, PopQA, HotpotQA, and 2Wiki; seven-set aggregates include all QA benchmarks. Cost is the external-output-token proxy in micro-USD-equivalent units.

Comparison	Five-seed mean ± SD	95% paired-bootstrap CI
NQ EM	$+0.038 \pm 0.010$	$[+0.020, +0.056]$
PopQA EM	$+0.038 \pm 0.007$	$[+0.021, +0.055]$
HotpotQA EM	$+0.027 \pm 0.009$	$[+0.014, +0.040]$
2Wiki EM	$+0.017 \pm 0.006$	$[+0.005, +0.029]$
MuSiQue EM	$+0.019 \pm 0.006$	$[+0.007, +0.031]$
Four-set average EM	$+0.030 \pm 0.006$	$[+0.022, +0.038]$
Seven-set average EM	$+0.029 \pm 0.006$	$[+0.016, +0.042]$
Four-set average cost	-22.6 ± 4.0	$[-26.4, -18.8]$

The aggregate EM and cost intervals exclude zero, supporting statistical significance under this paired-bootstrap protocol. Dataset-level intervals are reported separately; small absolute differences should not be treated as equally important merely because their intervals exclude zero. Five training seeds remain a modest basis for characterizing training variability. The original 17.8% cost reduction and the five-seed absolute reduction of 22.6 units refer to different evaluation protocols; the latter must not be converted to a percentage using the original-protocol baseline mean.

A.9 One-Shot versus Recursive Decomposition

Joint optimization of correctness and efficiency trains the policy to balance answer quality against external-call expenditure through its decomposition and allocation decisions. In the evaluated QA setting, the trained one-shot policy obtains a more favorable trade-off than the recursive variant under matched training conditions. A single explicit plan fixes subquestion identities and commitment order, while evidence acquisition and model selection adapt during execution. Recursive planning additionally revises the unresolved plan after execution has begun. The following training comparison and inference-time probe provide distinct evidence about the resulting policy behavior.

Matched training comparison. Both policies are compared under the reported matched training setting with $T = 8$. The recursive variant permits $R_{\max} = 2$ additional decomposition actions after the initial plan. Plan completion is measured among decomposed trajectories; format violations are measured per episode.

Table 10: One-shot and recursive policies trained under the same eight-turn budget. External calls measure invocation overhead, not dollar cost.

Policy	Plan completion	Format violation	External calls/episode	Avg. EM
One-shot ($T = 8$)	0.914	0.044	1.9	0.421
Recursive ($R_{\max} = 2$, $T = 8$)	0.746	0.118	3.0	0.404

Under this training comparison, recursive planning raises external calls from 1.9 to 3.0, reduces plan completion, and decreases EM by 1.7 percentage points. Revising or superseding plans also complicates assignment of final-outcome credit to earlier decisions. These results show the performance–overhead advantage of the trained one-shot policy in this setting; they do not rule out stronger recursive policies using different representations or objectives.

Diagnostic inference-time replanning. A separate experiment keeps the submitted one-shot-trained checkpoint fixed and relaxes the environment to allow replanning after two uninformative

responses, including an escalation attempt. Completed subanswers are preserved, and only the unresolved suffix may change. All variants use $T = 8$.

Table 11: Inference-time probe using the same one-shot-trained checkpoint. Cost is the external-output-token proxy. This is not a comparison of independently trained planning policies.

Inference policy	Avg. EM	Avg. cost
One-shot	0.423	110.6
One additional replan ($R_{\max} = 1$)	0.424	128.2
Two additional replans ($R_{\max} = 2$)	0.421	140.5

One additional replan gives a 0.001 EM increase at greater cost, while two replans give a 0.002 EM decrease relative to the one-shot reference. No material accuracy improvement is observed for the added expenditure in this probe. Since the checkpoint was not trained to replan, this evidence is a diagnostic of its accuracy–cost trade-off, not proof that one-shot planning is intrinsically superior. Long-horizon tasks and non-monotonic dependencies may benefit from recursive replanning beyond the current QA formulation.

A.10 Supplementary Mathematical QA Evaluation

We evaluate MATH-500 as an additional mathematical QA setting using the same candidate pool, routing budget, verifier, and cost-accounting convention across the compared systems. Search-R1-MultiTool retains the released Search-R1 checkpoint and multi-turn tool-use loop, but replaces its retrieval backend with the candidate LLM pool, so each `<search>` becomes a parameterized model-tool call. This is a zero-shot backend adaptation without matched retraining on the new tool interface.

Table 12: MATH-500 under a matched candidate pool and routing budget. Accuracy is mathematical answer correctness under the shared verifier. Cost is the external-output-token proxy.

Method	Accuracy	Avg. cost
Router-R1-Qwen	51.0%	276.7
Granule-R1	57.0%	194.6
Search-R1-MultiTool	40.8%	292.3

Granule-R1 improves accuracy by 6.0 percentage points over Router-R1-Qwen and by 16.2 points over Search-R1-MultiTool, with lower proxy cost in both comparisons. The backend adaptation is an additional tool-use comparison, but does not fully isolate the structured formulation from a comparably retrained generic planner. MATH-500 extends the evidence to mathematical QA; it does not establish generalization to coding, data analysis, or arbitrary agent workflows.

A.11 Training and Controller Overhead

Shared cost accounting. Following Router-R1, all methods in our cost comparisons use the same candidate-model pricing vector and external-output-token accounting convention. Router inference is excluded from this metric for both Granule-R1 and the baselines. Thus, the reported 17.8% reduction measures external-output charges under the original evaluation protocol. The price mapping and USD conversion are given in Appendix A.14. To complement this comparison, we separately profile controller training and inference under matched settings below.

Table 13: Training profiles on one A100-80GB over 225 PPO steps. Wall-clock totals include rollout/reward time. External-service hardware consumption is not separately measured.

Method	Wall-clock / GPU-hours	Rollout/reward	Share
Router-R1-Qwen	10.21 h / 10.21	6.02 h	58.9%
Granule-R1	11.37 h / 11.37	8.49 h	74.6%

Training overhead. Under the same hardware and 225-step PPO schedule, Granule-R1 requires 11.37 GPU-hours compared with 10.21 for Router-R1-Qwen, an additional 1.16 GPU-hours (11.4%). Rollout/reward processing accounts for 8.49 hours (74.6%) and 6.02 hours (58.9%), respectively. These measurements describe training time on the profiled GPU; the external services used during rollouts are not included as additional measured GPU-hours.

Table 14: Controller inference on one RTX 4090 using FP16 vLLM and batch size 10. Router latency excludes external-model latency. Router tokens include decomposition, intermediate reasoning, and router-resolved content.

Method	Dataset	External calls/query	Router tokens/query	Router latency/query
Router-R1-Qwen	NQ	1.06	139.19	1.54 s
Router-R1-Qwen	HotpotQA	1.12	178.95	1.85 s
Granule-R1	NQ	2.31	392.57	2.72 s
Granule-R1	HotpotQA	2.47	416.85	3.57 s

Inference overhead and self-resolution. The matched measurements in Table 14 include all router-generated decomposition, reasoning, and directly resolved content. Relative to Router-R1-Qwen, Granule-R1 generates 253.38 additional router tokens per NQ query and 237.90 per HotpotQA query, with controller-only latency increases of 1.18 and 1.72 seconds, respectively. These latencies exclude external-model service time and therefore do not establish an end-to-end latency comparison. Self-resolved subquestions incur no external-call price, while their computation is included in these controller measurements. The router independently resolves approximately 10–12% of single-hop queries and 6–8% of multi-hop queries; these percentages use queries, rather than committed subanswers, as the denominator.

Controller footprint and deployment interpretation. Our controller has 3B parameters, compared with 70B for a large expert in the candidate pool, and is profiled on a single GPU. Its smaller model footprint can reduce reliance on large-expert serving when decomposition enables collaboration among smaller models. However, parameter count alone does not determine the controller's share of serving cost: generated tokens, context length, hardware utilization, batching, and external-model execution also matter. The profiles quantify controller overhead but do not assign it a per-query dollar price or establish that it is negligible. The measured monetary saving is therefore the reduction in external-output charges; full deployment cost additionally requires accounting for input tokens, controller execution, verification, and the serving factors listed in Appendix A.14.

Offline supervision for lightweight routers. RouterDC and GraphRouter have lighter online routing components. If their dense query–model supervision is constructed from scratch under the matched 14K-query, six-model setup, it entails 84K candidate-model evaluations. This is the size of the required supervision matrix, not a measured dollar cost. Reusing public data or checkpoints can amortize that offline work. It should therefore be distinguished from both online router computation and the external-output charges reported in our cost comparisons.

A.12 Failure, Decomposition, and Informativeness Audits

Incorrect-trajectory audit. We sample 200 incorrect trajectories across the seven QA benchmarks. GPT-5.5 assigns preliminary categories under a predefined failure taxonomy, after which every complete trajectory and label is manually verified. Disagreements are resolved through manual review. Each trajectory is assigned only its earliest causal error. A model-selection error requires replaying the same subquestion and finding another candidate that supplies the needed evidence; if no candidate succeeds, the failure is assigned to the model pool.

Avoidable decomposition. We pair ordinary inference with inference in which DECOMPOSE is disabled. Decomposition is redundant when both variants are correct and the variant without decomposition is cheaper; it is harmful when decomposition changes a correct result into an incorrect one. These are inference counterfactuals, not comparisons between separately retrained policies.

Table 15: Earliest causal error among 200 sampled incorrect trajectories.

Failure category	Count	Share
Incorrect or incomplete decomposition	49	24.5%
Inappropriate model selection	45	22.5%
Incorrect subanswer commitment	38	19.0%
Premature direct answer / insufficient routing	28	14.0%
Final aggregation failure	20	10.0%
Candidate-model pool failure	12	6.0%
Format violation / turn-budget exhaustion	8	4.0%

Table 16: Avoidable decomposition. All columns use all dataset inputs as the denominator, except the final column, which conditions on decomposed inputs. Percentages are rounded.

Dataset	Decomposed	Redundant	Harmful	Avoidable	Within decomposed
TriviaQA	12.7%	2.4%	0.8%	3.2%	25.2%
NQ	18.4%	3.7%	1.2%	4.9%	26.6%

Approximately one quarter of the decomposition decisions on these two predominantly single-hop datasets remain avoidable under this counterfactual definition. The learned decision rule therefore reduces, but does not eliminate, unnecessary decomposition.

False-positive informativeness. The final expanded audit selects 100 final-answer-correct, reward-eligible trajectories from each of the seven benchmarks and manually examines every committed subanswer.

Table 17: Final informativeness audit, conditional on correct final answers and reward eligibility.

Audit statistic	Value
Sampled trajectories	700
Committed subanswers examined	1,155
Incorrect commitments nevertheless marked informative	21
Erroneous-informative commitments / examined commitments	1.82%
Trajectories sampled per benchmark	100

The ratio $21/1155 = 1.82\%$ uses committed subanswers as its denominator. It is not an unconditional error rate over all external calls, nor the fraction of trajectories containing an error. These observations show that final-outcome gating does not make the informativeness heuristic a factuality verifier: incorrect intermediate commitments can coexist with a correct final answer. Calibrated verification is a possible extension but would introduce additional inference cost and potentially stochastic reward noise.

A.13 Optional CCP Pool-Shift Case Study

CCP is disabled in all main experiments. In a separate deployment case study, three of the six training-seen candidate models are replaced with unseen models. The same shifted pool is evaluated with and without CCP, using the same learned router checkpoint. Comparing either shifted-pool result directly with the original pool does not isolate CCP because the candidates also change.

Within this shifted pool, enabling CCP reduces cost by approximately 10.8% and end-to-end latency by 20.3%, with a 0.005 EM decrease. This supports an efficiency trade-off in the particular evaluated setting. It does not establish general robustness across unseen pools, pricing schemes, or data-distribution shifts, and does not imply that CCP is necessary to introduce unseen candidates without retraining. Broader evaluation remains future work.

Table 18: Exploratory CCP experiment under one candidate-pool replacement. Cost is the external-output-token proxy; latency is end-to-end and is distinct from controller-only latency in Table 14.

Pool	CCP	Avg. EM	Avg. cost	Latency
Six seen models	Off	0.423	110.6	18.3 s
Three seen + three unseen	Off	0.436	115.2	19.2 s
Three seen + three unseen	On	0.431	102.8	15.3 s

A.14 Price Mapping and Scope of the Cost Proxy

The controlled comparisons use the fixed Together-style price vector below, matching the accounting convention of Router-R1. These are the coefficients used for the reported results, not a claim about current API prices.

Table 19: Fixed candidate-model coefficients p_L, in USD per million output tokens.

Candidate model	p_L
LLaMA-3.1-8B-Instruct	0.18
Mistral-7B-Instruct	0.20
Qwen2.5-7B-Instruct	0.30
Gemma-2-27B-Instruct	0.80
LLaMA-3.1-70B-Instruct	0.88
Mixtral-8x22B-Instruct	1.20

With the reported proxy C_{eval} from Eq. (18), the corresponding metered external-output cost in USD is

$$C^{\$}_{\text{ext}}(y) = 10^{-6} C_{\text{eval}}(y), \qquad C^{\$}_{\text{ext}}(y; p') = 10^{-6} \sum_{L \in \mathcal{P}} p'_L N^{\text{out}}_L(y), \tag{25}$$

where $N^{\text{out}}_L(y)$ sums recorded output tokens from all calls to model L in the trajectory. Thus an average reported value of 110.6 corresponds to USD 0.0001106 per query under this fixed vector. Self-resolved subquestions contribute no external-call charge.

The proxy excludes input-token charges, router computation, charges for failed calls without recorded outputs, verification, context-growth charges, caching discounts, queueing, and other serving overhead. Repeated calls with returned outputs are counted, including uninformative ones. It measures metered external output, not full deployment cost or latency. Training and controller profiles are reported separately in Appendix A.11.