ProMoRouter: Joint Prompt and Model Routing for Cost-Efficient LLM Inference

Anonymous ACL submission

Abstract

Routing has emerged as a key strategy for deploying large language models (LLMs) at scale by assigning each query to the most appropriate model to balance output quality against inference cost. Existing routers, however, optimize only model selection while treating the prompting strategy as fixed, despite it substantially affecting both quality and cost. We address this gap by formulating routing as a joint prompt–model selection problem and introduce **ProMoRouter**, a graph-based router that scores candidate prompt–model actions over a heterogeneous interaction graph connecting queries, tasks, prompts, and models. ProMoRouter achieves the best reward among learned routers in the balanced and cost-sensitive settings, especially under tight cost budgets. Our results confirm that the prompting strategy is an integral routing decision rather than a fixed experimental condition. The repository is available at: `https://anonymous.4open.science/r/ProMoRouter-3FC8/`

1 Introduction

Large language model (LLM) routing balances inference cost against output quality: it sends each query to the most suitable model in a pool, reserving expensive models for hard queries and lightweight ones for easy queries. Learned routers can now match strong models at a fraction of the cost (Chen et al., 2024; Feng et al., 2025). Yet they reduce routing to one choice—*which model to use*—and apply a fixed prompting strategy to every query. This is costly, because how a query is presented shapes both quality and cost, independently of the model chosen (Wei et al., 2022). A chain-of-thought prompt can improve accuracy on a complex query, but generates far more tokens; a direct prompt is cheap, yet often too thin for multi-step reasoning (Wang et al., 2023; Zhou et al., 2023). Nor is the best strategy universal. It depends on the query—simple arithmetic needs no intermediate reasoning, whereas a multi-step word problem does—and on the model, since a stronger model may not need the guidance a weaker one requires (Fu et al., 2023; Diao et al., 2024). Prompt and model selection are therefore coupled, and a router that fixes the prompt must either waste tokens on easy queries or sacrifice accuracy on hard ones (Ong et al., 2025; Hu et al., 2024).

This coupling reshapes the routing problem: the action space becomes a joint space of prompt–model pairs, and the router must choose within it. We formalize this as *prompt-aware routing* and introduce **ProMoRouter**, a router that jointly selects how to prompt and which model to use for each query. Naive solutions fall short: running every prompt and keeping the best multiplies cost, while treating pairs as independent forgoes shared structure across actions and queries. Similar models can benefit from similar prompts, and queries with similar reasoning demands can exhibit similar routing behavior. ProMoRouter exploits this structure. It represents each prompt–model pair as an edge in a heterogeneous graph of queries, tasks, prompts, and models, and an edge-aware graph neural network scores each edge by its expected deployment reward $R(q, a) = P(q, a) - \lambda C(q, a)$. Message passing then generalizes across queries, prompts, and models that behave alike, rather than learning each pair in isolation.

We evaluate ProMoRouter across three deployment settings that vary the performance–cost weight λ: performance-first, balanced, and cost-sensitive. ProMoRouter achieves the highest reward in the balanced and cost-sensitive settings, improving deployment reward by up to 41.7% over GraphRouter ($R = 0.411$ vs. 0.290) while reducing inference cost by 16% ($C = 0.107$ vs. 0.127). The gains peak under tight budgets, where routing easy queries to a direct prompt buys the most efficiency without compromising quality (Feng et al., 2025; Chen et al., 2024). When cost is only weakly

penalized, the best fixed prompt–model pair stays strongest, since a single high-quality action is hard to beat adaptively.

Our contributions are as follows:

- We reformulate LLM routing as a decision over a joint prompt–model action space, showing the prompting strategy is a routing decision that interacts with model choice, not a fixed condition.
- We introduce **ProMoRouter**, which casts prompt-aware routing as edge scoring over a heterogeneous graph of queries, tasks, prompts, and models, so the router learns shared structure across pairs rather than treating them as independent.
- We show that ProMoRouter achieves the best reward among learned routers in the balanced and cost-sensitive settings, with the largest gains under tight cost budgets.

2 Related Work

LLM Routing under Cost Constraints. Cost-aware routing systems such as FrugalGPT (Chen et al., 2024) and RouteLLM (Ong et al., 2025) dynamically allocate queries to cheaper or stronger models depending on predicted difficulty. These methods show that adaptive routing can outperform static deployments, but treat prompting as fixed.

Graph-Based Routers. Beyond threshold or classifier-based routing, graph-based methods model structured relationships among queries, tasks, and candidate models. GraphRouter (Feng et al., 2025) introduced graph neural networks for learning routing policies over heterogeneous interaction graphs, showing that relational inductive biases can improve model selection decisions.

Prompting strategies for reasoning. Prompt design can substantially affect model performance, especially on reasoning tasks. Chain-of-thought prompting (Wei et al., 2022), self-consistency (Wang et al., 2023), least-to-most decomposition (Zhou et al., 2023), and self-reflection methods (Madaan et al., 2023) improve performance through intermediate reasoning, decomposition, or verification. These results suggest that inference quality depends not only on the model, but also on how the query is prompted.

Algorithm 1: ProMoRouter Edge-GNN training and inference

Input : Training log $\mathcal{D} = \{(q_i, t_i, p_i, m_i, R_i)\}$ with $R_i = P(q_i, a_i) - \lambda C(q_i, a_i)$;
node embeddings Xq, Xt, Xp, Xm; prompt set $\mathcal{P}$;
model set $\mathcal{M}$;
model cost table $\kappa(m)$; deployment weight λ;
ranking weight β
Output : Trained router parameters θ

Graph construction
Construct a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node types (t, q, p, m)
foreach $(q_i, t_i, p_i, m_i, R_i) \in \mathcal{D}$ **do**
  Add structural edges (q_i, t_i), (q_i, p) for $p \in \mathcal{P}$, and (q_i, m) for $m \in \mathcal{M}$
  Add observed action edge (q_i, p_i, m_i) with target reward R_i

Training
repeat
  Compute node representations $h_v = \mathrm{GNN}_\theta(\mathcal{G})$ for $v \in \{q, p, m\}$
  foreach *query* q_i *in mini-batch* **do**
    foreach $a = (p, m) \in \mathcal{P} \times \mathcal{M}$ **do**
      $\phi_{q_i a} = [\, C(q_i, a);\ \mathbf{e}_{pm};\ \bar{R}_{pm} \,]$
      $\phi_{q_i a} = [\, C(q_i, a; \kappa(m));\ \mathbf{e}_{pm};\ \bar{R}_{pm} \,]$
      $s_\theta(q_i, a) = f_\theta(h_{q_i}, h_p, h_m, \phi_{q_i a})$
    $a_i^* = \arg\max_{a \in \mathcal{A}_{q_i}} R_\lambda(q_i, a)$
    $\mathcal{L}_{\mathrm{reg}} = \sum_{a \in \mathcal{A}_{q_i}} \big(s_\theta(q_i, a) - R_\lambda(q_i, a)\big)^2$
    $\mathcal{L}_{\mathrm{rank}} = -\log \dfrac{\exp(s_\theta(q_i, a_i^*))}{\sum_{a' \in \mathcal{A}_{q_i}} \exp(s_\theta(q_i, a'))}$
    Update θ using $\mathcal{L}_{\mathrm{reg}} + \beta \mathcal{L}_{\mathrm{rank}}$
until *early stopping on validation reward*

Inference
Given a test query q, add q to $\mathcal{G}$ with structural edges to t, $\mathcal{P}$, and $\mathcal{M}$
Compute $h_q, h_p, h_m = \mathrm{GNN}_\theta(\mathcal{G})$
return $a^*(q) = \arg\max_{(p,m) \in \mathcal{P} \times \mathcal{M}} s_\theta(q, (p, m))$

Positioning. Existing routers optimise *which model* to call; prompting strategy is treated as a fixed experimental condition rather than a routing variable. ProMoRouter addresses this gap with a heterogeneous graph-based router that represents tasks, queries, prompts, and models as distinct node types, and learns to select prompt–model action edges under a cost-aware deployment reward.

3 Methodology

Problem Definition Given an input query q, the router selects an action $a = (p, m)$, where $p \in \mathcal{P}$ is a prompting strategy and $m \in \mathcal{M}$ is a candidate language model. Unlike prior routers that optimize model selection alone, our action space jointly captures *how* to prompt and *which* model to use. Each action yields a response with a task-specific performance score $P(q, a) \in [0, 1]$ and a normalized inference cost $C(q, a) \in [0, 1]$, computed from

model-level token prices $\kappa(m)$ and the observed token usage. We optimize the deployment reward

$$R(q, a) = P(q, a) - \lambda\, C(q, a), \qquad (1)$$

where $\lambda \in [0, 1]$ controls the performance–cost trade-off: small λ prioritizes quality; large λ penalizes cost. We study three settings—performance-first ($\lambda = 0.1$), balanced ($\lambda = 0.5$), and cost-sensitive ($\lambda = 0.9$)—and train a dedicated router for each to select, per query, the action that maximizes $R(q, a)$.

3.1 ProMoRouter

ProMoRouter is a graph-based router that models tasks, queries, prompts, and models as nodes in a heterogeneous interaction graph (t, q, p, m). The architecture proceeds in three stages: graph construction, message passing over the heterogeneous graph, and edge-aware scoring of candidate prompt–model actions (Algorithm 1).

Graph construction. Task, query, prompt, and model nodes are initialized from text embeddings: queries from their input text, and tasks, prompts, and models from short semantic descriptions. *Structural edges* link each query to its task, all prompts, and all models, encoding the available action space. *Action edges*, drawn from the training logs, link each query to the prompt–model pairs it was executed with, carrying the reward signal the router learns from.

Message passing. We instantiate the encoder as a graph convolutional network (Kipf and Welling, 2017): each node aggregates transformed messages from its neighbors across edge types and updates its representation accordingly. After L layers (Section 4.1), each query embedding reflects its parent task and observed action history, while prompt and model embeddings reflect the queries they have served, giving representations $\{h_q, h_p, h_m\}$ that generalize across queries with similar routing behavior.

Edge scoring. For each query q and candidate action $a = (p, m)$, the router predicts a reward via an edge-aware scoring function:

$$s_\theta(q, a) = f_\theta(h_q,\, h_p,\, h_m,\, \phi_{qa})\,, \qquad (2)$$

where ϕ_{qa} is an action-level feature vector formed by concatenating the normalized table-based cost $C(q, a)$, a learned prompt–model edge embedding $\mathbf{e}_{pm}$ capturing interaction effects beyond individual node embeddings, and the mean training-log reward $\bar{R}_{pm}$ as a data-driven prior. We pass $[\,h_q;\, h_p;\, h_m;\, \phi_{qa}\,]$ through a two-layer MLP to produce the scalar score $s_\theta(q, a)$. The router selects the highest-scoring candidate:

$$a^*(q) = \arg\max_{(p,m)\,\in\,\mathcal{P}\times\mathcal{M}} s_\theta(q, a). \qquad (3)$$

ProMoRouter thus builds on graph-based routing principles from(Feng et al., 2025) while extending the decision space from model selection to joint prompt–model action selection.

Metric	Mistral-7B	Qwen2.5-7B	Llama3.1-8B
Δ Exact Match	+0.061	+0.080	-0.006
Token Cost Ratio	8.32×	8.82×	7.37×

Table 1: RQ1 on GSM8K: Decomposition improves Exact Match for most small models but incurs a roughly 8× token cost compared to monolithic prompting.

3.2 Training and Inference

We train ProMoRouter offline from interaction logs containing $R(q, a) = P(q, a) - \lambda C(q, a)$ for multiple prompt–model actions under a fixed setting λ. The objective combines two complementary terms (Algorithm 1): a regression term $\mathcal{L}_{\text{reg}}$ that fits predicted scores to observed rewards, calibrating their magnitudes onto the true reward scale; and a list-wise ranking term $\mathcal{L}_{\text{rank}}$, a softmax cross-entropy over the candidate set $\mathcal{A}_q$ that concentrates probability on the highest-reward action a^*. The full objective is $\mathcal{L} = \mathcal{L}_{\text{reg}} + \beta\, \mathcal{L}_{\text{rank}}$, with β tuned on validation: regression calibrates absolute estimates, while ranking directly optimizes the top-action selection used at inference (Eq. 3). At inference, the encoder computes node representations, the scorer evaluates all $|\mathcal{P}| \times |\mathcal{M}|$ candidate actions, and the router returns $a^*(q)$.

4 Experiments

We address two research questions. **RQ1:** How does prompting strategy shape the performance–cost trade-off? **RQ2:** Does joint prompt–model routing outperform model-only routing?

4.1 Setup

Interaction Log Construction. We build an offline interaction log from four benchmarks (GSM8K (Cobbe et al., 2021), HotpotQA (Yang

Method	Performance-first (λ = 0.1)			Balanced (λ = 0.5)			Cost-sensitive (λ = 0.9)		
	P	C	R	P	C	R	P	C	R
Largest LLM	0.462	0.756	0.386	0.462	0.756	0.084	0.462	0.756	-0.219
Smallest LLM	0.265	0.034	0.261	0.265	0.034	0.248	0.265	0.034	0.234
Best Fixed Model	0.545	0.569	0.488	0.398	0.129	0.334	0.374	0.076	0.306
Best Fixed Pair	**0.610**	0.569	**0.553**	0.473	0.114	0.416	0.473	0.114	0.370
FrugalGPT-direct	0.476	0.516	0.377	0.215	0.056	0.079	0.214	0.054	-0.027
RouteLLM-direct	0.481	0.576	0.375	0.216	0.056	0.080	0.216	0.056	-0.029
GraphRouter (model-only, qnorm)	0.550	0.420	0.508	0.412	0.123	0.351	0.405	0.127	0.290
ProMoRouter (qnorm)	0.561	0.435	0.517	**0.586**	0.218	**0.477**	**0.508**	**0.107**	**0.411**
Oracle Model-Only	0.671	0.350	0.636	0.621	0.178	0.532	0.582	0.116	0.477
Oracle	0.811	0.198	0.792	0.799	0.140	0.729	0.770	0.101	0.680

Table 2: Main results under query-normalized cost across three deployment settings (P = performance, C = cost, R = reward). ProMoRouter leads in balanced and cost-sensitive settings; the best fixed pair leads performance-first.

et al., 2018), SQuAD (Rajpurkar et al., 2016), and Alpaca-style instruction-following prompts (Taori et al., 2023)), running four prompting strategies—Direct, Chain-of-Thought, Decomposition, Self-Check—across nine open-weight LLMs (7B–72B). Each execution records the query, prompt, model, task score, and token usage. After filtering malformed outputs, the log holds 34.6k interactions and $|\mathcal{P}| \times |\mathcal{M}| = 36$ candidate actions per query.

Protocol and Metrics. We split queries query-level into non-overlapping train/validation/test sets, train one router per setting $\lambda \in \{0.1, 0.5, 0.9\}$, and select checkpoints by validation reward with early stopping. We report test results. Performance P is exact match (GSM8K) or token-level F1 (HotpotQA, SQuAD, Alpaca); cost C is query-normalized token usage across all 36 actions; reward is $R = P - \lambda C$.

Baselines. We compare against four static baselines—Largest LLM, Smallest LLM, Best Fixed Model, and Best Fixed Prompt–Model Pair—testing whether a single globally optimal static choice matches adaptive routing. The adaptive baseline is **GraphRouter** (Feng et al., 2025), a model-only router under fixed prompting. We also report **FrugalGPT-direct** and **RouteLLM-direct** under a shared direct prompt, plus oracle model-only and full oracle upper bounds.

4.2 Main Results

RQ1: Reasoning-intensive prompts improve quality at a steep token cost. Table 1 compares direct prompting with decomposition on GSM8K for three compact models. Decomposition improves exact-match accuracy for Qwen2.5-7B (+0.080) and Mistral-7B (+0.061), while slightly reducing accuracy for Llama3.1-8B. These gains come at a substantial token cost: average usage rises from roughly 100–130 tokens to 940–980 tokens, corresponding to a 7–9$\times$ increase. This shows that prompting strategy substantially alters the quality–cost trade-off.

RQ2: Joint prompt–model routing improves deployment reward across settings. Table 2 reports held-out test results under three deployment settings. In the balanced setting ($\lambda = 0.5$), ProMoRouter obtains the highest reward ($R = 0.477$), above GraphRouter ($R = 0.351$) and the Best Fixed Pair ($R = 0.416$). These gains amplify under the cost-sensitive setting ($\lambda = 0.9$), where ProMoRouter improves the reward from 0.290 to 0.411 relative to GraphRouter while reducing cost by 16% ($C = 0.107$ vs. 0.127). Under the performance-first setting ($\lambda = 0.1$), the Best Fixed Pair remains strongest ($R = 0.553$ vs. 0.517). We attribute this to the fact that when cost pressure is absent, a globally strong static policy acts as a near-optimal upper bound, whereas adaptive routing must navigate estimation noise across the full action space $\mathcal{P} \times \mathcal{M}$. Joint routing is most beneficial under active cost constraints.

Prompt versus model routing. To isolate each routing axis, Table 3 compares fixed versus adaptive configurations under the balanced setting. Notably, adaptive prompt selection alone ($R = 0.431$, $C = 0.134$) outperforms adaptive model selection alone ($R = 0.416$, $C = 0.127$), indicating that prompt diversity is a primary driver of the performance–cost trade-off. Full joint routing (ProMoRouter) leverages both axes to achieve the highest overall reward ($R = 0.477$) and lowest cost ($C = 0.119$).

5 Limitations

Our study relies on offline interaction logs from a fixed set of tasks, prompting strategies, and open-weight LLMs, so we cannot assess live deployment, distribution shift, or unseen model families. We also train a separate router per deployment setting and do not test transfer to new models or prompts, so generalizing across unseen actions remains future work. The action space grows with $|\mathcal{P}| \times |\mathcal{M}|$, so scoring every prompt–model pair becomes more expensive as the pools expand, and large action spaces may need candidate pruning. Query-normalized cost reduces but does not remove token-usage variance across prompt types, so the cost estimates stay approximate. Finally, ProMoRouter adds offline training cost and a per-query selection step; the latter generates no additional tokens, so it does not affect the token-based cost we report, though we leave latency profiling to future work.

Risks. ProMoRouter is evaluated offline on existing benchmark data and is not deployed with end users. A potential risk is that routing systems may be used to reduce inference cost without adequate monitoring of reliability, bias, or task-specific failure modes. We therefore view prompt–model routing as a deployment aid that should be combined with standard evaluation and safety checks for the selected models.

AI Usage Disclosure

Generative AI tools were used for limited language refinement and editing assistance. All scientific content, methodology, experiments, and conclusions were developed and verified by the authors.

Ethics Statement

Our experiments use public benchmark-style datasets and generated model interaction logs for evaluating routing decisions. We do not intentionally collect or use personally identifying information. Since model outputs may occasionally contain undesirable or biased content, our work should be understood as evaluating routing behavior rather than endorsing any generated content.

References

Lingjiao Chen, Matei Zaharia, and James Zou. 2024. Frugalgpt: How to use large language models while reducing cost and improving performance. *Trans. Mach. Learn. Res.*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. 2024. Active prompting with chain-of-thought for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 1330–1350.

Tao Feng, Yanzhen Shen, and Jiaxuan You. 2025. Graphrouter: A graph-based router for LLM selections. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, pages 10421–10430.

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

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. 2025. Routellm: Learning to route llms from preference data. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in*

Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, pages 2383–2392.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. https://crfm.stanford.edu/2023/03/13/alpaca.html.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Appendix A.1: Ablation on Routing Configuration

To isolate the contribution of joint prompt–model routing, we compare three routing configurations on GSM8K under the balanced setting ($\lambda = 0.5$): (i) **Fixed Prompt + Adaptive Model**: the prompting strategy is held constant while the router selects the model; (ii) **Adaptive Prompt + Fixed Model**: the router selects the prompting strategy while the model is held constant; (iii) **Full Joint Routing** (ProMoRouter): both prompt and model are selected jointly.

Table 3 reports reward R, performance P, and cost C for this task-specific diagnostic setting.

Full joint routing yields the highest reward and performance while also incurring the lowest inference cost in this diagnostic setting. This suggests that adaptive prompting and adaptive model selection provide complementary gains rather than redundant ones.

Table 3: Ablation on routing configuration for GSM8K under the balanced setting ($\lambda = 0.5$). Higher R and P are better; lower C is better.

Configuration	R	P	C
Fixed Prompt + Adaptive Model	0.416	0.389	0.127
Adaptive Prompt + Fixed Model	0.431	0.401	0.134
Full Joint Routing (ProMoRouter)	**0.477**	**0.443**	**0.119**

Appendix A.2: Analysis of the Performance-First Setting

In the performance-first setting ($\lambda = 0.1$), the reward is dominated by the performance term P, and token cost is only weakly penalized. Under these conditions, the **Best Fixed Pair** achieves higher reward than ProMoRouter ($R = 0.553$ vs. 0.517; Table 2). This outcome is plausible because a globally strong prompt–model configuration can act as a near-optimal static policy when cost pressure is largely irrelevant.

By contrast, adaptive routing must estimate fine-grained reward differences across the full prompt–model action space $\mathcal{P} \times \mathcal{M}$ and may occasionally assign a query to a slightly weaker action. The marginal value of joint prompt–model routing is therefore largest when cost pressure is active, as in the balanced and cost-sensitive settings, where efficiency opportunities cannot be fully captured by a single fixed policy.

Prompt versus model routing. The diagnostic ablation in Appendix A.1 further decomposes the contribution of each routing axis on GSM8K. Adaptive prompt selection alone yields higher reward than adaptive model selection alone under the balanced setting, indicating that prompt diversity contributes directly to the performance–cost trade-off. Full joint routing benefits from both axes simultaneously and achieves the best overall reward in this setting.