RADAR: Reasoning–Ability and Difficulty-Aware Routing for Reasoning LLMs

Nigel Fernandez[*1], **Branislav Kveton**[2], **Ryan A. Rossi**[2], **Andrew S. Lan**[2], **Zichao Wang**[2]
[1]University of Massachusetts Amherst, [2]Adobe Research
{nigel,andrewlan}@cs.umass.edu, {kveton,ryrossi,jackwa}@adobe.com

Abstract

Reasoning language models have demonstrated remarkable performance on many challenging tasks in math, science, and coding. Choosing the right reasoning model for practical deployment involves a performance and cost tradeoff at two key levels: model size and reasoning budget, where larger models and higher reasoning budget lead to better performance but with increased cost and latency. In this work, we tackle this tradeoff from the angle of model configuration routing for different queries, and present RADAR (**R**easoning–**A**bility and **D**ifficulty-**A**ware **R**outing), a lightweight, interpretable, and scalable routing framework. Inspired by psychometrics, RADAR learns an *item response model* from model responses with different budgets to different queries, with *interpretable* parameters including *query difficulties* and *model-budget abilities*. RADAR then routes queries with higher difficulty to model-budget pairs with higher ability, and vice versa. We conduct extensive experiments on 8 widely used challenging reasoning benchmarks, demonstrating the superior performance of RADAR compared to state-of-the-art model routing methods. RADAR also exhibits *query generalization* capabilities, showing strong performance on out-of-distribution queries in all benchmarks. RADAR is also *scalable* and can efficiently integrate additional models by *dynamically selecting* a small set of evaluation queries to estimate their abilities.

1 Introduction

Recent advances in reasoning language models (RLMs), trained with reinforcement learning to produce chain-of-thought reasoning [42, 53, 12, 35], have achieved strong performance across math [28], science [38], coding [22], perception [27], and tool use [54]. Model developers release RLMs in multiple sizes and with configurable reasoning budgets, already numbering in the thousands [2]. Always choosing the largest, most expensive configuration is often wasteful: many queries can be answered by smaller models or with lower reasoning effort, while more complex queries require stronger configurations (see Figure 4). Over-thinking can even hurt performance [46, 13, 18, 43, 10] (see Appendix A for an extended related work). This performance-cost tradeoff presents a challenge for practitioners: given a diverse pool of (model, reasoning budget) configurations, how can we route each query to the "right" configuration that balances performance and cost?

We propose RADAR, a lightweight framework for Reasoning-Ability and Difficulty-Aware Routing. RADAR uses item response theory (IRT) [37, 26, 47, 5] to jointly model query difficulty and configuration ability, enabling interpretable, low-latency ($\sim$7 ms) routing before model execution. Our approach treats RLMs as black boxes, requires no fine-tuning, and supports plug-and-play integration of new configurations via adaptive calibration [48, 17]. By casting configuration selection as multi-objective optimization (MOO) [30, 8], RADAR chooses performance-cost tradeoffs towards

[*]Work done during an internship at Adobe Research.
[2]Hugging Face lists 2,710 RLMs as of September 17th, 2025.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MATH-AI.

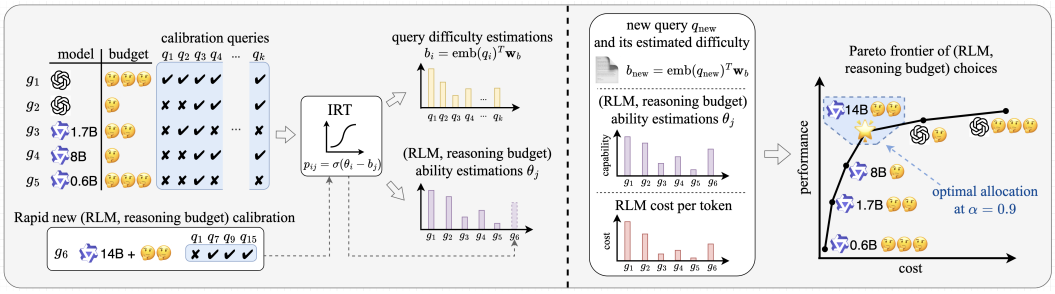


Figure 1: **Left**: RADAR jointly estimates *interpretable* query difficulties and RLM configuration abilities using IRT. New RLM configurations can be rapidly added by estimating their ability on a small subset of *dynamically selected* queries using adaptive testing. **Right**: RADAR formulates routing as multi-objective optimization and routes queries to sufficiently capable configurations, optimizing performance-cost tradeoffs towards the Pareto frontier.

the Pareto front, achieving superior routing across 8 challenging reasoning benchmarks. For example, on MATH-500 [16], it matches 90% of OpenAI o4-mini's high-budget performance at just 1.31% of its cost, while generalizing well to long-context QA [23], and newly added RLM configurations.

2 Methodology

Routing as Multi-objective Optimization. We discretize each RLM $m \in \mathcal{M}$ by its available reasoning budgets $u \in \mathcal{U}_m$ into configurations $g = (m, u) \in \mathcal{G} \subseteq M \times \mathcal{U}_m$. We present a novel view of model routing through the lens of MOO [30, 2, 32, 57]. Given a set of queries $\mathcal{Q} = \{q_1, \ldots, q_k\}$ and a set of candidate model configurations $\mathcal{G} = \{g_1, \ldots, g_n\}$, our goal is to assign each query $q_i \in \mathcal{Q}$ to the optimal configuration $g_j \in \mathcal{G}$ which maximizes performance and minimizes cost. For each query q, we define a two-dimensional a-priori [2] MOO with two objective functions, performance $p_q(g)$ (probability of correctness) and cost $c_q(g)$ (normalized token cost): $g^* = \arg\max_{g \in \mathcal{G}} f(p_q(g), c_q(g))$, where f is a scalarization function. This formulation enables different scalarization techniques to achieve different points on the performance-cost Pareto front. We explore linear scalarization [32] and Chebyshev scalarization [57]. The **linear scalarization** problem of our MOO with weight $w_1 \in [0, 1]$ is: $\text{LSP}_q^{w_1} = \arg\max_{g \in \mathcal{G}} \; w_1 p_q(g) - (1 - w_1) c_q(g)$, recovering the formulation presented in existing routing methods [20, 44, 59] but arrived at through the lens of an MOO. The **Chebyshev scalarization** problem of our MOO, capturing points on the non-convex parts of the Pareto front, is: $\text{CSP}_q^{w_1} = \arg\min_{g \in \mathcal{G}} \max\{w_1 |1 - p_q(g)|, \, (1 - w_1) c_q(g)\}$. In both scalarization techniques, the weight parameter w_1 controls the trade-off between performance and cost: a larger value of w_1 indicates a preference for performance over cost, favoring stronger model configurations more often, while a smaller value of w_1 indicates a preference for weaker but more cost-effective model configurations. Given a user-specified tradeoff profile with weight w_1 and query q, RADAR assigns configuration $g = \text{LSP}_q^{w_1}$ (or configuration $g = \text{CSP}_q^{w_1}$ depending on the chosen scalarization scheme) to maximize performance and minimize cost.

IRT-based Calibration. To parameterize $p_q(g)$, we adopt the 2-parameter logistic (2PL) IRT model [37, 26, 5] from psychometrics research used to model student responses to test items. Each query q has discrimination a and difficulty b obtained from linear projections of its embedding $\boldsymbol{e}$, where $a = \boldsymbol{w}_a^\top \boldsymbol{e}$ and $b = \boldsymbol{w}_b^\top \boldsymbol{e}$. Each configuration g has a scalar ability θ. The probability of g correctly answering q, a binary-valued outcome, is given by: $p(y = 1 \mid q, g) = \sigma\left(a(\theta - b)\right)$, where $\sigma(\cdot)$ is the sigmoid function. We train the model by minimizing the binary cross-entropy loss over an evaluation matrix of responses from all RLM configurations to all training queries.

Cost Prediction. Following prior routing work [20, 44], we adopt a heuristic-based approach to parameterize the cost prediction function: $c_q(g) = 1/|\mathcal{Q}| \sum_{q \in \mathcal{Q}} (n^{\text{rsn}}_{g(q)} + n^{\text{cmp}}_{g(q)}) \cdot t_g$, where t_g is the cost per token of the base RLM, and $n^{\text{rsn}}_{g(q)}$ and $n^{\text{cmp}}_{g(q)}$ are the total number of reasoning tokens and completion tokens generated, respectively. We min-max normalize all costs to the range $[0, 1]$.

Expanding the Pool of RLM Configurations Through Adaptive Testing. To add a new configuration g_i, inspired by psychometric testing [48], we estimate its ability on a *small* set of training queries $\mathcal{Q}^*$ by computing: $\hat{\theta}_i = \arg\max_\theta \prod_{q_j \in \mathcal{Q}^*} \left[\sigma(a_j(\theta - b_j))\right]^{y_{ij}} \left[1 - \sigma(a_j(\theta - b_j))\right]^{1 - y_{ij}}$. Inspired

Table 1: Routing performance on ID queries across benchmarks reported on the hypervolume metric (higher is better) and CPT (90%) metric (lower is better). RADAR outperforms baselines, denoting better performance-cost tradeoffs towards the Pareto frontier. See Table 5 and Table 6 for performance on OOD queries.

Benchmark	Random-Pair	RouterBench	IRT-Router	RADAR (ours)
GPQA-Diamond	0.555, 80.4%	0.687, 57.1%	0.694, 54.0%	**0.751**, **13.2**%
MMLU	0.691, 76.3%	0.859, 2.7%	0.860, **2.7**%	**0.872**, **2.7**%
MMLU-Redux	0.728, 75.1%	0.905, 2.6%	0.912, 2.8%	**0.923**, **2.4**%
MMLU-Pro	0.559, 83.6%	0.782, 5.2%	0.781, **3.8**%	**0.800**, 3.9%
LSAT	0.691, 80.1%	0.913, 2.0%	0.916, 1.9%	**0.919**, **1.8**%
AIME	0.516, 87.2%	0.768, 65.7%	**0.777**, 61.2%	0.776, **60.7**%
MATH-500	0.743, 76.2%	**0.953**, **1.3**%	0.942, 1.4%	0.945, **1.3**%
FRAMES	0.659, 77.9%	0.833, 43.5%	0.850, 31.5%	**0.876**, **13.1**%

by [17], queries are selected adaptively using Fisher information: $I(\theta, a_j, b_j) = a_j^2 \sigma(a_j(\theta - b_j))\big(1 - \sigma(a_j(\theta - b_j))\big)$. Set $\mathcal{Q}^*$ is updated as: $Q_i^*(t) = Q_i^*(t-1) \cup \left\{\arg\max_{q_j \in \mathcal{Q} \setminus Q_i^*(t-1)} I(\theta, a_j, b_j)\right\}$.

3 Experimental Evaluation

Benchmarks and Metrics. We evaluate on eight reasoning benchmarks, including **AIME** [28], **MATH** [16], **GPQA** [38], **LSAT** [49, 60], **MMLU** [15, 14], **MMLU Redux** [9], **MMLU Pro** [51], and **FRAMES** [23], spanning competition math, PhD-level science, law, and general knowledge with both multiple-choice and open-ended formats. In particular, **FRAMES** probes RADAR's generalization to long-context queries in a multi-doc QA setting; long-context evaluation is largely absent in prior routing work. We report **hypervolume** [8] (higher is better), which, in our setting, corresponds to the area under the performance-cost trade-off curve across weights w_1. We also use the cost–performance threshold (**CPT**) metric (lower is better), akin to call-performance thresholds [33]: relative to the best-performing configuration, o4-mini–high, $\mathrm{CPT}(x\%)$ is the minimum cost required to reach $x\%$ of that performance, normalized by the cost of o4-mini–high. For example, $\mathrm{CPT}(90\%) = 0.1$ means achieving 90% performance at 10% of the cost.

Baselines and Evaluation Setup. We compare RADAR to several recent, state-of-the-art model routing methods, including **RouterBench** [20, 4] and **IRT-Router** [44], and heuristic-based baselines, including **All-Large**, **All-Small**, **Oracle**, **Random-All** and **Random-Pair**. We adapt these methods for our experimental setting since they do not natively generalize to RLMs (see Appendix B.2). We conduct both in-distribution (ID) and out-of-distribution (OOD) evaluations. We route over 35 configurations comprising OpenAI **o4-mini** (budgets: low, medium, high) and **Qwen3** models (0.6B/1.7B/4B/8B) with budgets 0, 256, 512, 1k, 2k, 4k, 8k, 16k [53, 34]. In total, we collected 1.75 million binary responses over $50{,}139$ questions across all eight benchmarks (see Appendix B).

3.1 Results, Analysis and Discussion

RADAR outperforms state-of-the-art model routing methods. Table 1 reports the routing performance of all methods across 8 reasoning benchmarks evaluated in the ID setting on the hypervolume and CPT(90%) metrics, respectively. We see that RADAR outperforms all baselines on most benchmarks and performs comparably to the best existing baseline on the remaining benchmarks. RADAR outperforms IRT-Router [44], a concurrent IRT-based routing work, suggesting that our novel formulation of RLM routing as an MOO, as well as the use of solution techniques like Chebyshev scalarization, enable better recovery of the Pareto performance-cost frontier. On the challenging GPQA-Diamond benchmark, RADAR demonstrates an 8% performance boost over the second-best method on hypervolume. On the CPT metric, on MATH-500, RADAR matches 90% of the performance of o4-mini with a high reasoning budget at 1.31% of its cost.

RADAR exhibits strong query generalization capabilities. Table 5 and Table 6 report routing performance of all methods across 8 reasoning benchmarks evaluated in the OOD setting on the hypervolume and CPT (90%) metrics, respectively. We show the Pareto performance-cost tradeoff curves for all methods on OOD queries from FRAMES [23] in Figure 2. We see that RADAR exhibits strong generalization to OOD queries, outperforming existing state-of-the-art methods on most benchmarks. When generalizing to OOD queries with significantly higher difficulty (e.g., AIME) than those seen during training, RADAR tends to assign a model configuration with a slightly lower ability than optimal, resulting in a slight decrease in performance. This weakness can potentially be addressed by including a small number of representative queries during training.

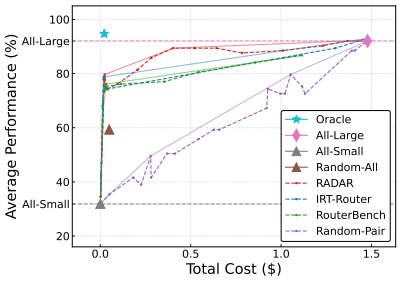


Figure 2: On FRAMES [23] OOD queries, RADAR can match 90% of the performance of OpenAI o4-mini with high reasoning effort at just 10% of its cost, with the next best method [44] requiring 30% of the cost.

Table 2: Routing performance across benchmarks reported on the hypervolume metric (higher is better), before (RADAR) and after (RADAR++) adding new RLM configurations from Qwen3-14B. RADAR++ quickly estimates the abilities of new configurations through adaptive testing for an improved routing performance.

Benchmark	ID		OOD	
	RADAR	RADAR++	RADAR	RADAR++
GPQA-Diamond	0.7513	0.7535	0.7466	0.7463
MMLU	0.8720	0.8731	0.8609	0.8698
MMLU-Redux	0.9230	0.9238	0.9072	0.9091
MMLU-Pro	0.7995	0.8021	0.7858	0.7951
LSAT	0.9188	0.9233	0.9146	0.9255
AIME	0.7760	0.7828	0.7566	0.7566
MATH-500	0.9449	0.9461	0.9368	0.9368
FRAMES	0.8762	0.8830	0.8865	0.8931

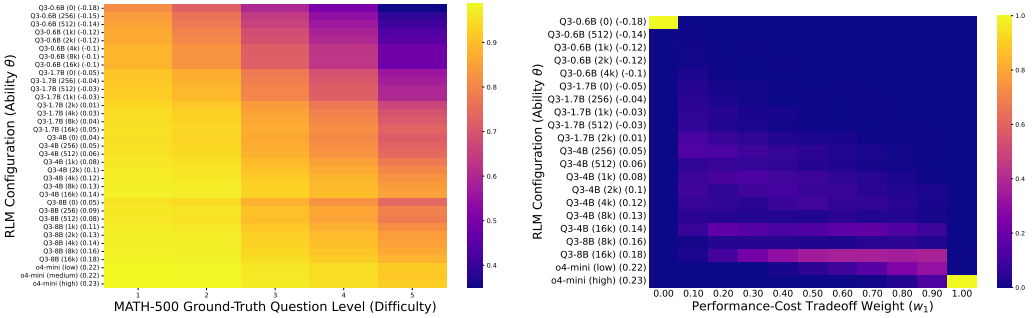


Figure 3: RADAR estimates *interpretable* query difficulties and RLM configuration abilities. **Left:** Mean predicted correctness probability of configurations on queries with 5 different ground-truth difficulty levels in MATH-500. As difficulties increase, configurations with higher abilities are predicted to perform better. **Right:** Fraction of routing calls on MATH-500 queries spread across RLM configurations when varying the performance-cost tradeoff weight. A lower (higher) weight leverages a higher fraction of Qwen3 (o4-mini) configurations, prioritizing cost (performance).

Ablation and latency study. Table 3 shows that Chebyshev scalarization outperforms linear scalarization on OOD queries, with linear being marginally better on ID queries. Table 4 shows that RADAR achieves a similar performance to using the entire training set with just 20% of subsampled training queries. RADAR works in real-time adding *minimal latency* with details included in Appendix D.

Model scalability and generalization evaluation. We evaluate the scalability of RADAR to new RLMs by adding 8 new model configurations from the Qwen3 14B RLM. Table 2 shows the result; using adaptive testing, RADAR accurately estimates the abilities of these new configurations by dynamically selecting just 5k training queries (12% of the training set) for evaluation, resulting in improved routing performance. Figure 5 shows how routing shifts to new RLM configurations.

RADAR estimates interpretable query difficulties and RLM configuration abilities. On ID queries from MATH-500 [16], annotated with one of five levels of increasing difficulty, the Pearson correlation coefficient between estimated query difficulties and ground-truth levels is 0.509. In Figure 3 (left), we see that with an increase in query level, configurations with higher abilities are predicted to have higher correctness probabilities, leading to a mean correctness prediction accuracy of 84.42%. Figure 3 (right) shows the fraction of routing calls across configurations as the performance-cost weight is varied, with cost-effective Qwen3 models (performant o4-mini) preferred at lower (higher) weights.

4 Conclusion

We introduced RADAR, a reasoning–ability and difficulty-aware routing framework that (1) formalizes adaptive reasoning as an MOO, and (2) leverages IRT to adaptively assign queries to RLM model–budget configurations. RADAR achieves strong cost–performance tradeoffs, generalizes well to OOD queries, offers interpretability by exposing query difficulty and model abilities, and supports plug-and-play integration of new RLM configurations through adaptive calibration.

Acknowledgements

The authors would like to thank Tong Sun and Jaewook Lee for helpful discussions. We also thank the anonymous reviewers for their helpful comments.

References

[1] P. Aggarwal and S. Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

[2] J. Branke, K. Deb, K. Miettinen, and R. Slowinski. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer Science & Business Media, 2008.

[3] L. Chen, M. Zaharia, and J. Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

[4] Z. Chen, Z. Wei, Y. Bai, X. Xiong, and J. Wu. Tagrouter: Learning route to llms through tags for open-domain text generation tasks. *arXiv preprint arXiv:2506.12473*, 2025.

[5] C. DeMars. *Item response theory*. Oxford University Press, 2010.

[6] D. Ding, A. Mallick, C. Wang, R. Sim, S. Mukherjee, V. Ruhle, L. V. Lakshmanan, and A. H. Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*, 2024.

[7] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

[8] M. T. Emmerich and A. H. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, 17(3):585–609, 2018.

[9] A. P. Gema, J. O. J. Leang, G. Hong, A. Devoto, A. C. M. Mancino, R. Saxena, X. He, Y. Zhao, X. Du, M. R. G. Madani, et al. Are we done with mmlu? *arXiv preprint arXiv:2406.04127*, 2024.

[10] S. S. Ghosal, S. Chakraborty, A. Reddy, Y. Lu, M. Wang, D. Manocha, F. Huang, M. Ghavamzadeh, and A. S. Bedi. Does thinking more always help? understanding test-time scaling in reasoning models. *arXiv preprint arXiv:2506.04210*, 2025.

[11] M. Gor, H. Daumé III, T. Zhou, and J. Boyd-Graber. Do great minds think alike? investigating human-ai complementarity in question answering with caimira. *arXiv preprint arXiv:2410.06524*, 2024.

[12] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[13] M. Hassid, G. Synnaeve, Y. Adi, and R. Schwartz. Don't overthink it. preferring shorter thinking chains for improved llm reasoning, 2025.

[14] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[15] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[16] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[17] V. Hofmann, D. Heineman, I. Magnusson, K. Lo, J. Dodge, M. Sap, P. W. Koh, C. Wang, H. Hajishirzi, and N. A. Smith. Fluid language model benchmarking. In *Second Conference on Language Modeling*, 2025.

[18] J. Hong, T. Zhen, K. Chen, J. Liu, W. Zhu, J. Huo, Y. Gao, D. Wang, H. Wan, X. Yang, B. Wang, and F. Meng. Reconsidering overthinking: Penalizing internal and external redundancy in cot reasoning, 2025.

[19] B. Hou, Y. Zhang, J. Ji, Y. Liu, K. Qian, J. Andreas, and S. Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*, 2025.

[20] Q. J. Hu, J. Bieker, X. Li, N. Jiang, B. Keigwin, G. Ranganath, K. Keutzer, and S. K. Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024.

[21] S. Huang, H. Wang, W. Zhong, Z. Su, J. Feng, B. Cao, and Y. R. Fung. Adactrl: Towards adaptive and controllable reasoning via difficulty-aware budgeting. *arXiv preprint arXiv:2505.18822*, 2025.

[22] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

[23] S. Krishna, K. Krishna, A. Mohananey, S. Schwarcz, A. Stambler, S. Upadhyay, and M. Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. *arXiv preprint arXiv:2409.12941*, 2024.

[24] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention, 2023.

[25] C. Lee, A. M. Rush, and K. Vafa. Critical thinking: Which kinds of complexity govern optimal reasoning length? *arXiv preprint arXiv:2504.01935*, 2025.

[26] F. M. Lord. *Applications of item response theory to practical testing problems*. Routledge, 2012.

[27] P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, and J. Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.

[28] MAA. MAA Invitational Competitions; Mathematical Association of America — maa.org. `https://maa.org/maa-invitational-competitions/`, 2024. [Accessed 03-09-2025].

[29] G. Meng, Q. Zeng, J. P. Lalor, and H. Yu. A psychology-based unified dynamic framework for curriculum learning, 2024.

[30] K. Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.

[31] N. Muennighoff, Z. Yang, W. Shi, X. L. Li, L. Fei-Fei, H. Hajishirzi, L. Zettlemoyer, P. Liang, E. Candès, and T. Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

[32] T. Murata and H. Ishibuchi. MOGA: Multi-objective genetic algorithms. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, pages 289–294, 1995.

[33] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

[34] OpenAI. Introducing openai o3 and o4-mini. `https://openai.com/index/introducing-o3-and-o4-mini/`, Apr. 2025. Accessed: 2025-09-24.

[35] OpenAI and et al. Openai o1 system card, 2024.

[36] F. M. Polo, L. Weber, L. Choshen, Y. Sun, G. Xu, and M. Yurochkin. tinybenchmarks: evaluating llms with fewer examples. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[37] G. Rasch. Studies in mathematical psychology: I. probabilistic models for some intelligence and attainment tests. 1960.

[38] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[39] P. Rodriguez, J. Barrow, A. M. Hoyle, J. P. Lalor, R. Jia, and J. Boyd-Graber. Evaluation examples are not equally informative: How should that change NLP leaderboards? In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4486–4503, Online, Aug. 2021. Association for Computational Linguistics.

[40] M. Šakota, M. Peyrard, and R. West. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 606–615, 2024.

[41] A. Scarlatos, N. Fernandez, C. Ormerod, S. Lottridge, and A. Lan. Smart: Simulated students aligned with item response theory for question difficulty prediction. *arXiv preprint arXiv:2507.05129*, 2025.

[42] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[43] P. Shojaee, I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, and M. Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity, 2025.

[44] W. Song, Z. Huang, C. Cheng, W. Gao, B. Xu, G. Zhao, F. Wang, and R. Wu. Irt-router: Effective and interpretable multi-llm routing via item response theory. *arXiv preprint arXiv:2506.01048*, 2025.

[45] J. Su and C. Cardie. Thinking fast and right: Balancing accuracy and reasoning length with adaptive rewards. *arXiv preprint arXiv:2505.18298*, 2025.

[46] J. Su, J. Healey, P. Nakov, and C. Cardie. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *arXiv preprint arXiv:2505.00127*, 2025.

[47] W. J. van der Linden and R. K. Hambleton, editors. *Handbook of Modern Item Response Theory*. Springer, New York, NY, 1997.

[48] H. Wainer, N. J. Dorans, R. Flaugher, B. F. Green, and R. J. Mislevy. *Computerized adaptive testing: A primer*. Routledge, 2000.

[49] S. Wang, Z. Liu, W. Zhong, M. Zhou, Z. Wei, Z. Chen, and N. Duan. From lsat: The progress and challenges of complex reasoning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022.

[50] X. Wang, Y. Huang, Y. Wang, X. Luo, K. Guo, Y. Zhou, and X. Zhang. Adareasoner: Adaptive reasoning enables more flexible thinking. *arXiv preprint arXiv:2505.17312*, 2025.

[51] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

[52] Y. Xu, H. Dong, L. Wang, D. Sahoo, J. Li, and C. Xiong. Scalable chain of thoughts via elastic reasoning. *arXiv preprint arXiv:2505.05315*, 2025.

[53] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[54] S. Yao, N. Shinn, P. Razavi, and K. R. Narasimhan. {τ}-bench: A benchmark for \underline{T}ool-\underline{A}gent-\underline{U}ser interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025.

[55] Z. Yu, T. Xu, D. Jin, K. A. Sankararaman, Y. He, W. Zhou, Z. Zeng, E. Helenowski, C. Zhu, S. Wang, et al. Think smarter not harder: Adaptive reasoning with inference aware optimization. *arXiv preprint arXiv:2501.17974*, 2025.

[56] L. Yue, Y. Du, Y. Wang, W. Gao, F. Yao, L. Wang, Y. Liu, Z. Xu, Q. Liu, S. Di, et al. Don't overthink it: A survey of efficient r1-style large reasoning models. *arXiv preprint arXiv:2508.02120*, 2025.

[57] R. Zhang and D. Golovin. Random hypervolume scalarizations for provable multi-objective black box optimization. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[58] X. Zhang, Z. Huang, E. O. Taga, C. Joe-Wong, S. Oymak, and J. Chen. Efficient contextual llm cascades through budget-constrained policy learning. *Advances in Neural Information Processing Systems*, 37:91691–91722, 2024.

[59] Y. Zhang, H. Li, J. Chen, H. Zhang, P. Ye, L. Bai, and S. Hu. Beyond gpt-5: Making llms cheaper and better via performance-efficiency optimized routing, 2025.

[60] W. Zhong, S. Wang, D. Tang, Z. Xu, D. Guo, J. Wang, J. Yin, M. Zhou, and N. Duan. Ar-lsat: Investigating analytical reasoning of text, 2021.

[61] V. Zouhar, P. Cui, and M. Sachan. How to select datapoints for efficient human evaluation of nlg models? *arXiv preprint arXiv:2501.18251*, 2025.

A Extended Related Work

Efficient Reasoning. A rapidly growing literature seeks to make reasoning models themselves more efficient; see [56] for a broader overview of this direction. Methods such as L1 [1] and S1 [31] provide *length control*, enabling reasoning models to trade off accuracy and cost by constraining chain-of-thought length. Others prune or adapt the reasoning process by dynamically shortening or extending reasoning [19, 52, 50]; adaptively controlling inference steps [21]; and analyzing when additional reasoning is beneficial or wasteful [45, 46, 55, 10]. Theoretical perspectives further study optimal reasoning length [25]. These works aim to make a single model more efficient. They also require access to model weights, which usually do not apply for closed-source or black-box settings. Our approach is complementary: RADAR treats any such efficient reasoning model as an additional candidate in its pool of (model, reasoning effort) configurations. This means advances in adaptive or efficient reasoning can be seamlessly integrated into our framework, while RADAR contributes orthogonally by providing per-query routing, interpretability through IRT, and Pareto-optimal cost–performance control.

Routing for Foundation Models. Recent work studies cost–quality routing across multiple LLMs [3, 58, 6, 33, 20, 40, 4, 44]. Most methods focus on *model selection* with black-box predictors or cascades [3, 6, 33, 40, 4], though TREACLE additionally co-selects prompt types under budget constraints [58]. We instead study *adaptive reasoning* and cast this problem as routing over *model–budget configurations*, where the budget controls thinking-token effort, making reasoning cost an explicit decision dimension in addition to the model itself. Routers also differ in *when* they commit: cascaded approaches may re-query a model [3, 58], while others choose once per query [6]. Our router makes a single assignment before generation, avoiding mid-turn switching (and KV-cache recomputation) or multiple re-querying while still retaining favorable cost–quality trade-offs. Finally, we emphasize *interpretability and control*. Unlike opaque regressors [3, 6, 33], we use an IRT parameterization to expose query difficulty and configuration ability. Compared to a concurrent work that also utilizes IRT-based model for routing [44], our work (1) generalizes

the routing problem from model only to *model configurations*, which enables broader applicability e.g. for reasoning models at various budget levels, and (2) generalizes and formalizes the routing problem as a multi-objective optimization (MOO) problem, which enables a wide range of techniques for solving it, such as Chebyshev scalarization that we explored in this work which offers a more favorable tradeoff compared to linear scalarization.

Item Response Theory in Machine Learning. Originally designed for assessment and other educational applications, Item Response Theory (IRT) has emerged as a versatile tool for understanding and improving foundation models. It has been applied to evaluation and benchmarking such as jointly estimating model ability and item difficulty to build adaptive or efficient test suites [39, 61, 17, 36]; to training and curriculum design, where IRT-based difficulty estimates guide data selection for faster and more effective learning [29, 41]; and to the diagnostics and bias analysis, exposing strengths and weaknesses of models relative to humans or ideological leanings [11]. Most relevant to our setting, IRT has recently been explored for multi-model routing, where it parameterizes query difficulty and model ability to guide cost–performance trade-offs with interpretability [44]. Our work extends this line by applying IRT not only to model selection, but also to adaptive reasoning configurations (model × effort), contributing to the continuing exploration of IRT for foundation models.

B Additional Experimental Details

B.1 Hardware

For all open-source models, we use vLLM [24] to host the model. All experiments involving open-source models are run on NVIDIA A100 80GB GPUs.

B.2 Baselines

For **RouterBench** [20], we adopt its k-nearest neighbors (kNN) parameterization which performs best [4]. We adapt a concurrent IRT-based model routing work, **IRT-Router** [44], by using the same 2PL IRT model parameterization and query embedder as RADAR, thereby improving its embedder to handle long-context queries for fairness. For both model routing methods, **RouterBench** and **IRT-Router**, we adapt them to RLMs by using all RLMs at their respective fixed maximum budgets, and use their performance-cost formulation similar to linear scalarization (see Equation 2). In addition, we include simple heuristic-based baselines: **All-Large** (o4-mini at high budget) and **All-Small** (Qwen3 0.6B at zero budget) as approximate upper and lower bounds on performance and cost, respectively. The **Oracle** router, provided with model configuration performance on test queries, serves as an idealized approximate upper bound of the performance-cost tradeoff by picking the cheapest best-performing configuration. **Random-All** serves as a diversity baseline selecting a configuration at random to answer each query. **Random-Pair** selects the largest configuration (o4-mini at high budget) with probability w_1, and the smallest configuration (Qwen3 0.6B at zero budget) with probability $1 - w_1$, where w_1 is the user-defined performance-cost tradeoff weight.

B.3 IRT implementation details in RADAR

We employ a two-parameter logistic (2PL) IRT model implemented as a custom PyTorch model class. Input queries are first processed into fixed embeddings by a frozen-weight `Qwen/Qwen3-Embedding-8B`; the dimension $d_q = 4096$ for both the query embedding and and the learnable weights $\boldsymbol{w}_a, \boldsymbol{w}_b$. Training runs for 100 epochs with learning rate 5×10^{-4}, batch size 32 for both training and evaluation, gradient clipping at norm 1.0, and gradient accumulation of 1 step.

B.4 Metrics

Our formulation of adaptive reasoning as an MOO naturally lends the use of the **hypervolume** indicator metric [8], which measures the size of the dominated space recovered by the MOO solution method, with a higher value indicating performance close to the Pareto front. In our two-dimensional routing MOO, hypervolume intuitively measures the area under the performance-cost tradeoff curve recovered by the routing method for various values of tradeoff weights w_1. An advantage of

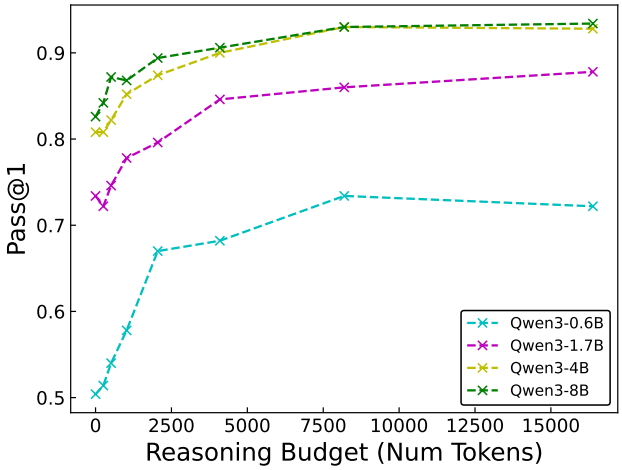


Figure 4: Our pilot study on MATH-500 [16] shows a performance differential over (RLM, reasoning budget) configurations, with the smallest RLM already solving over 50% of the queries with minimal reasoning.

hypervolume over similar area-based metrics defined in existing routing work (e.g. AIQ in [20]) is its generalizibility to measuring performance performance of a multi-dimensional routing MOO. In future work, additional dimensions such as latency, bias, and carbon emissions can be added to the routing MOO. We also formulate a **cost-performance threshold (CPT)** metric, similar to the call-performance threshold metrics in [33], a useful metric for real-world applications quantifying the cost required to reach a specified performance level. Given a performance threshold $x\%$, CPT($x\%$) measures the minimum cost required to achieve $x\%$ of the performance of the largest configuration (OpenAI o4-mini with high reasoning budget). We normalize this cost to $[0, 1]$ by dividing by the cost of running the largest configuration. Therefore, a CPT(90%) of 0.1 implies that the routing method can match 90% of the performance of o4-mini high at 10% of its cost.

C Additional Results

C.1 Pilot Study

Our pilot study on MATH-500 [16], shown in Figure 4, shows a performance differential over (RLM, reasoning budget) configurations, with the smallest RLM already solving over 50% of the queries with minimal reasoning.

C.2 Performance-Cost Pareto Curves

We show Pareto performance-cost tradeoff curves for all methods on ID queries across all benchmarks in Figure 6, and on OOD queries across all benchmarks in Figure 7.

C.3 Model Scalability and Generalization Evaluation of RADAR

We evaluate the scalability of RADAR to new RLMs by adding 8 new model configurations from the Qwen3 14B RLM. Using adaptive testing, RADAR accurately estimates the abilities of these new configuration by dynamically selecting just 5k training queries (12% of training set) for evaluation, resulting in improved routing performance. Figure 5 shows how routing shifts to new RLM configurations.

C.4 Ablation Study

See Table 3 for an ablation study which shows Chebyshev scalarization outperforms linear scalarization on OOD queries due to its ability to explore both convex and concave points on the Pareto front.

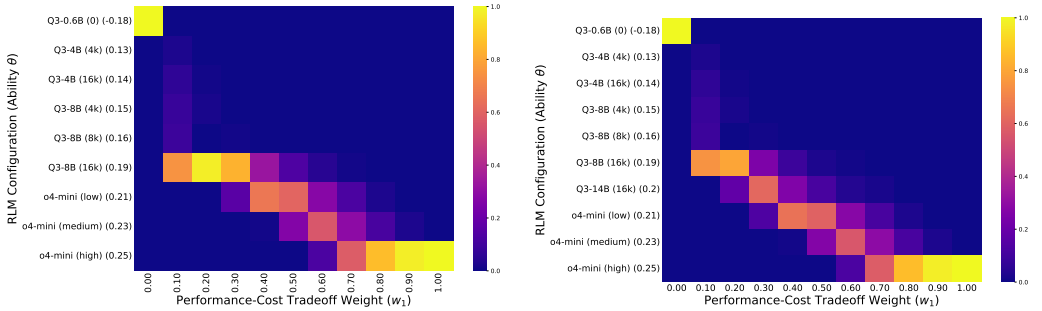


Figure 5: Fraction of routing calls on OOD queries from FRAMES spread across RLM configurations when varying the performance-cost tradeoff weight before (left) and after (right) adding new RLM configuration from Qwen3-14B. RADAR rapidly estimates the ability of Qwen3-14B at 16K reasoning budget to leverage it for improved performance.

Table 3: Ablation study showing Chebyshev scalarization outperforms linear scalarization on OOD queries due to its ability to explore both convex and concave points on the Pareto front.

Benchmark	Hypervolume (higher is better)		CPT(90%) (lower is better)	
	RADAR (LS)	RADAR (CS)	RADAR (LS)	RADAR (CS)
GPQA-Diamond	0.7280	0.7466	29.94%	17.60%
MMLU	0.8580	0.8609	2.50%	2.63%
MMLU-Redux	0.9049	0.9072	2.25%	2.54%
MMLU-Pro	0.7812	0.7858	3.81%	3.54%
LSAT	0.9165	0.9146	2.00%	2.15%
AIME	0.7464	0.7566	56.23%	55.30%
MATH-500	0.9331	0.9368	1.41%	1.55%
FRAMES	0.8656	0.8865	21.56%	9.99%

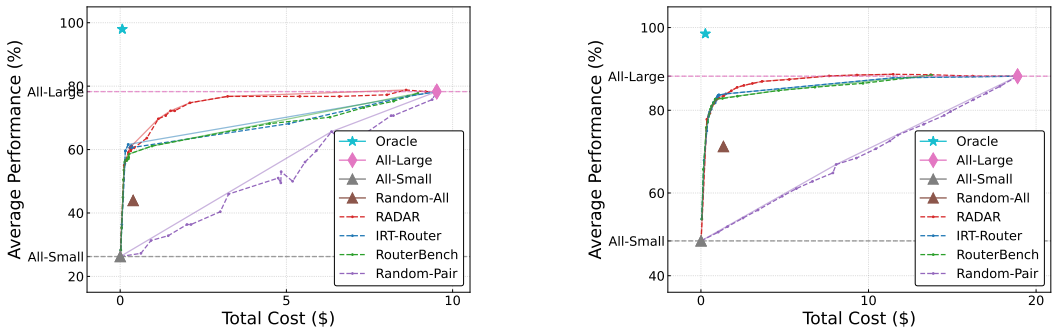
See Table 4 for an ablation study on the size of the training matrix of RADAR. Using just 20% of subsampled training queries, RADAR achieves a similar performance to using the entire training set.

C.5 Results on OOD Queries

Table 5 and Table 6 report routing performance of all methods across 8 reasoning benchmarks evaluated in the OOD setting on the hypervolume and CPT (90%) metrics, respectively. RADAR exhibits strong query generalization capabilities.

Table 4: Ablation study on the size of the training matrix of RADAR. Using just 20% of subsampled training queries, RADAR achieves a similar performance to using the entire training set.

Benchmark	Hypervolume (higher is better)		CPT(90%) (lower is better)	
	RADAR (20%)	RADAR	RADAR (20%)	RADAR
GPQA-Diamond	0.7526	0.7513	16.29	13.21
MMLU	0.8726	0.8720	2.67	2.69
MMLU-Redux	0.9207	0.9230	2.69	2.42
MMLU-Pro	0.7990	0.7995	3.59	3.89
LSAT	0.9175	0.9188	1.95	1.82
AIME	0.7832	0.7760	57.85	60.69
MATH-500	0.9450	0.9449	1.15	1.31
FRAMES	0.8940	0.8762	10.70	13.11



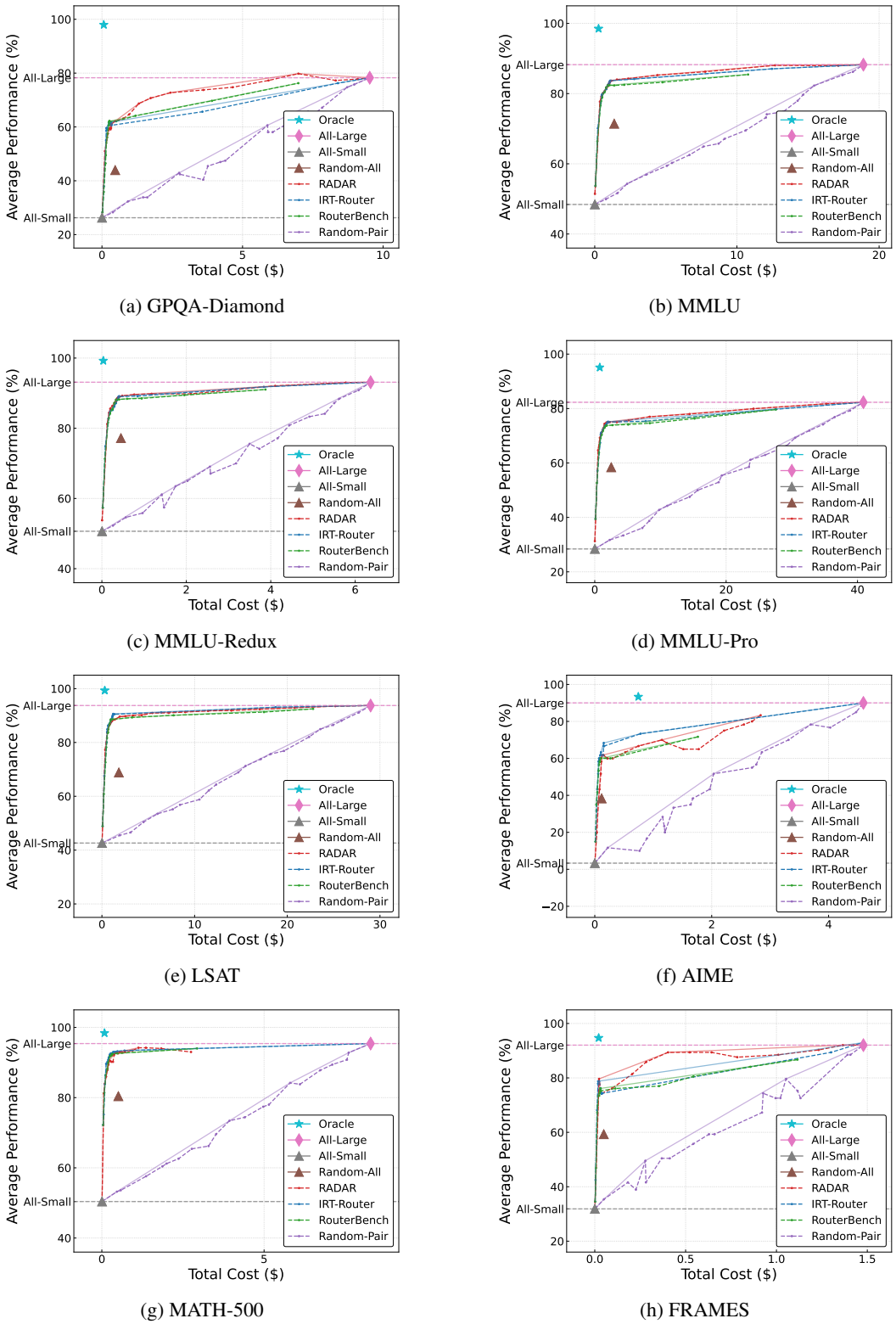
(a) GPQA-Diamond (b) MMLU

(c) MMLU-Redux (d) MMLU-Pro

(e) LSAT (f) AIME

(g) MATH-500 (h) FRAMES

Figure 6: We show the Pareto performance-cost tradeoff curves for all methods on ID queries across benchmarks. RADAR outperforms baselines, denoting better performance-cost tradeoffs towards the Pareto frontier.



(a) GPQA-Diamond (b) MMLU

(c) MMLU-Redux (d) MMLU-Pro

(e) LSAT (f) AIME

(g) MATH-500 (h) FRAMES

Figure 7: We show the Pareto performance-cost tradeoff curves for all methods on OOD queries across benchmarks. RADAR outperforms baselines, denoting better performance-cost tradeoffs towards the Pareto frontier.

Table 5: Routing performance on OOD queries across benchmarks reported on the hypervolume metric (higher is better). RADAR outperforms baselines denoting better performance-cost tradeoffs towards the Pareto frontier.

Benchmark	Random-Pair	RouterBench	IRT-Router	RADAR (ours)
GPQA-Diamond	0.5369	0.7047	0.6938	**0.7466**
MMLU	0.6934	0.8398	0.8550	**0.8609**
MMLU-Redux	0.7298	0.8948	0.9050	**0.9072**
MMLU-Pro	0.5686	0.7703	0.7800	**0.7858**
LSAT	0.6887	0.9046	**0.9175**	0.9146
AIME	0.5283	0.6890	**0.7915**	0.7566
MATH-500	0.7493	0.9326	**0.9385**	0.9368
FRAMES	0.6624	0.8230	0.8548	**0.8865**

Table 6: Routing performance on OOD queries across benchmarks reported on the CPT (90%) metric (lower is better). CPT (90%) denotes the fraction of cost of running OpenAI o4-mini with high reasoning effort to match 90% of its performance.

Benchmark	Random-Pair	RouterBench	IRT-Router	RADAR (ours)
GPQA-Diamond	82.54%	44.18%	54.19%	**17.6**%
MMLU	74.53%	2.94%	**2.61**%	2.63%
MMLU-Redux	74.61%	2.90%	2.71%	**2.54**%
MMLU-Pro	82.65%	7.67%	4.02%	**3.54**%
LSAT	80.07%	2.27%	**1.96**%	2.15%
AIME	84.88%	–	**55.19**%	55.30
MATH-500	74.94%	1.4%	**1.29**%	1.55%
FRAMES	78.61%	48.52%	29.49%	**9.99**%

D Latency Analysis of RADAR

We measure the latency of RADAR and compare it to the latency of the smallest RLM configuration (Qwen3-0.6B with 0 reasoning budget) used to generate answers to queries. The average per query routing latency overhead of RADAR over three runs of 500 queries from MATH-500 [16] is 6.89 ± 0.53 milliseconds. Compared to the time taken for the smallest RLM configuration to answer the query, which is 869.56 ± 1.1 milliseconds, RADAR adds negligible overhead.

E Dataset Description

The 9 benchmarks are: 1) **AIME** [28]: A benchmark of competition math problems from American Invitational Mathematics Examination (AIME), which determines qualification for the United States Mathematical Olympiad, 2) **MATH** [16]: A benchmark of math problems drawn from various math competitions, 3) **GPQA** [38]: A benchmark of PhD-level science multiple-choice questions (MCQs) written by domain experts, 4) **LSAT** [49, 60]: A benchmark of MCQs from the three tasks of the Law School Admission Test (LSAT), including analytical reasoning, logical reasoning and reading comprehension, 5) **MMLU** [15, 14]: A benchmark of MCQs from various branches of knowledge covering diverse domains, 6) **MMLU Redux** [9]: A subset of MMLU with manually corrected MCQs to remove errors from the original benchmark, 7) **MMLU Pro** [51]: An enhanced MMLU benchmark with a focus on reasoning questions with increased answer options from 4 to 10, 8) **DROP** [7]: A benchmark of reading comprehension questions requiring discrete reasoning over the question's associated paragraph, and 9) **FRAMES** [23]: A benchmark of long-context reasoning-based questions associated with multiple wikipedia articles. Table 7 shows the statistics of each dataset.

E.1 Preprocessing details

Across datasets, we standardize formatting; compute prompt token counts with the `Qwen/Qwen3-0.6B` tokenizer (no padding, truncation, or added special tokens); and discard items exceeding a configured token budget. To prevent leakage, we compute a content-based item key and apply deduplication when specified for a given dataset, with some datasets deferring duplicate

Table 7: Dataset statistics of prompt tokens across reasoning benchmarks used.

Dataset	Samples	Mean Tokens	Min Tokens	Max Tokens
AIME	1,035	143.00	30	3,312
MATH	8,000	86.16	24	806
GPQA	448	250.27	82	2,812
LSAT	2,025	263.63	174	570
MMLU	13,937	150.50	66	1,040
MMLU Redux	5,298	135.50	68	1,000
MMLU Pro	12,032	237.51	70	1,700
FRAMES	561	16,272.19	690	31,954

handling to later analysis. Where applicable, we normalize available metadata and extract missing numeric answers. All datasets are mapped into a unified prompt–response format; detailed prompt templates are provided in Appendix E.2.

AIME. We preprocess AIME by standardizing sources and prompts, then filtering and deduplicating. Training data span years 1983–2023,[3] while test data consist of the union of unique items from AIME 2024[4] and 2025[5] to reduce evaluation variance. Evaluating on AIME 2024 and AIME 2025 separately resulted in high evaluation variance even after averaging over multiple runs. Examples with prompt length exceeding the maximum token budget are discarded. For AIME 2025, only the problem text and numeric answer are retained; missing fields such as solution or difficulty are set to "NA."

MATH. We construct the training split by combining the seven subject configurations of the MATH dataset[6] (7,500 problems total) and use the fixed 500-problem test set.[7] Examples exceeding the maximum prompt length are removed. When an explicit numeric answer is missing, it is extracted from the provided solution. Metadata such as subject/type and level are normalized, and any available `unique_id` is preserved. We shuffle the data with a fixed seed.

GPQA. We preprocess GPQA[8] by combining the main and diamond subsets and verifying that diamond IDs are contained within the main set. Each example is reformatted into a multiple-choice format with options A–D, and the correct answer is recorded as a letter. Answer options are randomly permuted with a fixed seed. Items exceeding the maximum prompt length are filtered out. Deduplication is performed using a content-based key, and evaluation is conducted with the diamond subset as the test set.

LSAT. We preprocess LSAT by standardizing items from the official AR-LSAT release,[9] spanning reading comprehension, logical reasoning, and analytical reasoning. Each example is reformatted into a multiple-choice prompt with options A–E. Items exceeding the token budget are discarded. We preserve the original section and split labels, and record the gold answer both as an index and as a letter. Data are shuffled deterministically with a fixed seed.

MMLU. We use only the official test split of MMLU.[10] Each example is converted into a standardized multiple-choice prompt (options A–D), retaining both the textual correct answer and its letter index. Items exceeding the token threshold are discarded. Subject metadata are preserved, and the dataset is shuffled with a fixed seed.

[3] https://github.com/rllm-org/rllm/blob/deepscaler/deepscaler/data/train/aime.json
[4] https://github.com/rllm-org/rllm/blob/deepscaler/deepscaler/data/test/aime.json
[5] https://huggingface.co/datasets/yentinglin/aime_2025
[6] https://huggingface.co/datasets/HuggingFaceH4/MATH/viewer
[7] https://huggingface.co/datasets/HuggingFaceH4/MATH-500
[8] https://huggingface.co/datasets/Idavidrein/gpqa
[9] https://github.com/zhongwanjun/AR-LSAT/tree/main/complete_lsat_data
[10] https://huggingface.co/datasets/cais/mmlu

MMLU Pro. We use the public test split of MMLU Pro.[11] Each example is constructed into a multiple-choice prompt with options A–J, and the gold answer is stored as a letter. Prompts exceeding a 32k token budget are discarded. The dataset is shuffled deterministically with a fixed seed.

MMLU Redux. We preprocess MMLU Redux[12] by aggregating all subject configurations and discarding items flagged with metadata `error_type` $\neq$ `ok`. Each example is normalized and converted into a multiple-choice prompt (options A–D), with both the correct answer letter and text recorded. Items longer than the token budget are removed, and the dataset is shuffled deterministically with a fixed seed.

FRAMES. We preprocess FRAMES[13] by retrieving and cleaning the corresponding Wikipedia pages for each example. Cleaning removes site chrome, images, hyperlinks, citation markers, and irrelevant sections, while preserving tables and converting text to Markdown. Examples are then converted into a document QA style format. Items exceeding the token budget are filtered out, and unique article texts are cached to avoid re-downloading.

E.2 QA Prompts

The prompts below are applied to all RLM configurations.

For AIME and MATH, we use the following prompt:

```
{question}
Please reason step by step, and put your final answer within \boxed{}.
```

For GPQA, LSAT, MMLU, and MMLU Redux, we use the following prompt:

```
Answer the following multiple choice question.

{question}

A) {option_A}
B) {option_B}
C) {option_C}
D) {option_D}

Please reason step by step, and put your final answer option within \boxed{}.
Only put the letter in the box, e.g. \boxed{A}. There is only one correct
answer.
```

For MMLU Pro, we use the following prompt:

```
Answer the following multiple choice question.

{question}

{options}

Please reason step by step, and put your final answer option within \boxed{}.
Only put the letter in the box, e.g. \boxed{A}. There is only one correct
answer.
```

For FRAMES, we assemble the prompt programmatically that includes the context of all documents relevant to the question:

```
prompt = f"""You are asked to read {len(docs)} Wikipedia article extracts
and answer a question. Please reason step by step, and put your final
answer within \\boxed{}."""
```

[11] https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro
[12] https://huggingface.co/datasets/edinburgh-dawg/mmlu-redux-2.0
[13] https://huggingface.co/datasets/google/frames-benchmark

```
for i, doc in enumerate(docs):
    prompt += f"\n\n# Wikipedia article {i+1}:\n{doc}"
prompt += f"\n\n# Question: {question}"
prompt += """\n\nPlease reason step by step, and put your final answer within
    ↪ \\boxed{}."""
```

F Future Work

Several promising avenues for future work exist. First, we would like to extend RADAR beyond text to multi-modal reasoning settings. Second, incorporating additional configurations beyond the reasoning budget, such as retrieval, tool usage, and decoding algorithms, may yield fine-grained routing decisions for a wider range of applications, such as ultra-long context QA and deep research. Third, exploring RADAR in other constraint scenarios, such as when there is a total budget constraint on a batch of queries. Together, these directions highlight the broader potential of RADAR as a principled, interpretable foundation for adaptive reasoning in an ever-evolving RLM ecosystem.