

Adaptive Multi LLM Routing Under Cost Latency Quality Constraints

Saksham Rai

UC San Diego

DOI:10.37648/ijrst.v15i04.008

¹ Received: 14/10/2025; Accepted: 22/11/2025; Published: 29/11/2025

Abstract

Large language model services are increasingly offering heterogeneous portfolios of models, rather than a single universal endpoint. Each model inhabits a different point in a space of cost, latency, and quality, varying with prompt type, output length, system load and provider conditions. This paper formulates adaptive multi-LLM routing as a constrained contextual decision problem. A router estimates request-level quality, monetary cost, and tail latency for candidate models, selects the least costly feasible action, and escalates when confidence or verification signals are insufficient. The proposed architecture unifies an offline performance matrix, calibrated predictors, constrained policy, and online monitoring, and treats abstention, cascades and test-time sampling as first-class actions. Previous systems demonstrate massive savings, but production deployment involves explicit service-level constraints, drift-handling, and risk-sensitive evaluation beyond average accuracy. A practical evaluation protocol based on Pareto frontiers, constraint-violation rates, calibration and tail latency is presented and applied in an offline replay of GSM8K and MMLU with a weak and a strong model. At a quality floor of 90% of the strong model's accuracy, a simple calibrated router reduces cost by 45% on GSM8K and 69% on MMLU, but thresholds tuned to just meet the floor violate it in about half of the evaluation splits; a conservative threshold rule lowers the violation rate to at most 1% while retaining savings of 29% and 62%, respectively. The analysis suggests that robust routing gains are derived less from a universally superior router, than from accurate uncertainty estimates, diverse candidate models, and disciplined feedback loops.

Keywords: *large language models; model routing; constrained optimization; latency; inference cost; quality assurance; contextual bandits*

1. Introduction

No single large language model is universally best in coding, retrieval, reasoning, summarization, multilingual generation, and conversational tasks. The best performing model can also incur the highest cost or have the lowest throughput, and a smaller specialist model can match or exceed the largest model on a given query. This necessitates an operational choice of what model, configuration, or cascade to use for each query. FrugalGPT showed that learned cascades can preserve or enhance quality while sharply reducing API costs by choosing the right model at each step, demonstrating that routing can have implications beyond cost (Chen, Zaharia, & Zou, 2023). RouterBench later formalized the evaluation of routing decisions using metrics at the level of queries, costs of differing models, and quality scores (Hu et al., 2024).

¹ How to cite the article: Rai S.; (November, 2025); Adaptive Multi LLM Routing Under Cost Latency Quality Constraints; *International Journal of Research in Science and Technology*; Vol 15, Issue 4; 102-111, DOI: <http://doi.org/10.37648/ijrst.v15i04.008>

Routing decisions are however only concerned with cost. Interactive systems need to have time to first token guarantees, and regulated and safety-critical domains have minimum quality targets. There are also shifting demand patterns where the arrival rate of queries changes over time, modifying the delay incurred by waiting before routing after a policy has been trained. The routing problem thus has multiple objectives and constraints. This paper introduces a unifying formulation and an implementable architecture for adaptive selection which balances three simultaneous requirements: minimizing expected costs, satisfying latency service-level objectives, and not compromising on task specific quality. It brings together concepts from convex optimization, contextual bandits, selective prediction, model serving, and the recent literature on LLM-router. Figure 1 summarizes the resulting architecture, and Section 6 evaluates it empirically.

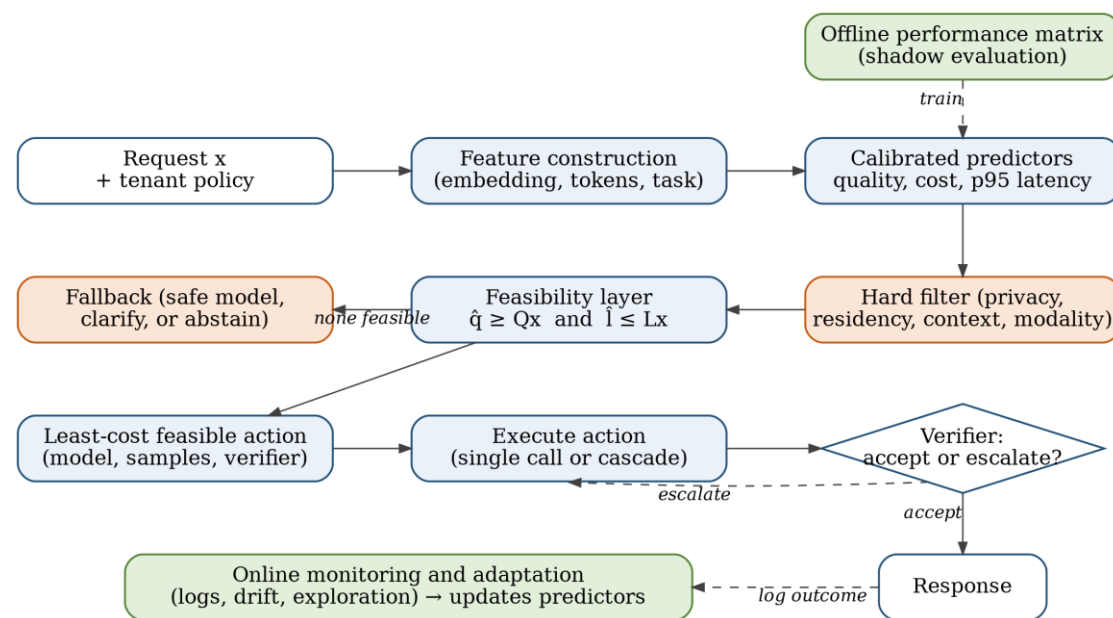


Figure 1: Architecture of the proposed adaptive router. Requests pass through feature construction and calibrated predictors of quality, cost, and latency; hard policy filters and a feasibility layer remove non-compliant actions before the least-cost feasible action is executed. A verifier accepts the response or escalates, a fallback handles requests with no feasible action, and online monitoring feeds observed outcomes back into the predictors.

2. Background and Related Work

Constrained optimization distinguishes between the objective and feasibility conditions and uses Lagrange multipliers to price constraint violations (Boyd & Vandenberghe, 2004). Contextual bandits generalize this notion to a sequence of decisions where an action is selected from observed context, and partial feedback is observed; exploration is traded off against exploitation of observed utility (Li et al., 2010). Latency in online services needs to be modeled as a distribution rather than a mean since rare, slow responses can dominate the tail at scale (Dean & Barroso, 2013). This framework is directly applicable to routing since the quality of unselected models is typically unobserved and the latency depends on the current load.

Recent works have investigated various routing families. FrugalGPT learns cascades that terminate when an inexpensive answer is sufficiently reliable (Chen et al., 2023). RouteLLM trains routers using preference data to decide between stronger and weaker models and reports significant improvements in cost without sacrificing quality (Ong et al., 2024). RouterDC learns representations in a shared contrastive space to improve selection when several models are viable options (Chen et al., 2024). LLM-Blender focuses on generating multiple candidate responses and then ranking or blending them, at the cost of higher computational overhead but greater robustness (Jiang, Ren, & Lin, 2023). In 2025, the work of BEST-Route further expanded the action space to include not only model choices but also

sample counts, demonstrating that multiple inexpensive samples can sometimes outperform one expensive response for some queries (Ding et al., 2025). Lastly, Arch-Router focused on interpretable domain-action preferences and extensibility to new models (Tran et al., 2025). OmniRouter, in turn, replaces per-query greedy selection with a global constrained optimization that minimizes total cost subject to a performance target.

Table 1. Comparison of multi LLM routing strategies

Strategy	Decision signal	Primary strength	Main limitation
Static rules	Task, language, tenant	Transparent and cheap	Brittle under drift; coarse quality estimates
Predictive router	Prompt embedding and metadata	Fast per-query model choice	Needs calibrated labels and retraining
Cascade	Answer confidence or verifier	Natural escalation and early exit	Extra latency and cost after escalation
Contextual bandit	Context plus online reward	Adapts to traffic and preferences	Partial feedback and unsafe exploration
Ensemble or fusion	Multiple candidate outputs	High robustness and complementary skills	Often violates tight cost or latency budgets
Joint model-sampling router	Difficulty, model, sample count	Uses test-time compute efficiently	Larger action space; verifier dependence

3. Problem Formulation

For request x and candidate model m , let $q(x, m)$ be expected quality, $c(x, m)$ the expected monetary or compute cost, and $l(x, m)$ some latency statistic like predicted p95 completion time.

A basic policy π selects an action minimizing expected cost subject to $q(x, m) \geq Q_x$ and $l(x, m) \leq L_x$.

These thresholds may vary by tenant, task, risk tier, or interaction mode.

A soft formulation could minimize $c + \lambda_q \max(0, Q_x - q) + \lambda_l \max(0, l - L_x)$ with multipliers indicating the operational cost of violations.

Hard filters should be applied before soft ranking when there are non-negotiable requirements such as privacy, data residency, context length, modality, or tool support.

The action need not be a single model but could also encode a tuple of model, decoding config, number of samples, verifier, or fallback. Quality is a vector: factuality, task success, safety, style, and user preferences cannot always be collapsed into a single metric. A practical system would therefore estimate a vector of qualities and apply a request-specific utility function. When uncertainty is high, the best action may be to abstain, ask clarifying questions, or escalate to a human rather than make a confident but wrong selection.

4. Adaptive Routing Architecture

4.1. Prediction and calibration

The router takes features constructed from prompt embeddings, token estimates, task and language labels, conversation state, retrieval statistics, and tenant policy and constructs features for a router. Independent predictors estimate each candidate's quality, input-output token cost, time to first token, and completion latency. Model-specific uncertainty is critical: a predicted score of 0.85 is not useful if its confidence interval spans the quality floor.

Calibration should be evaluated on a per-task, per-model basis: aggregate calibration can mask subpopulation breakdowns. Historical shadow evaluations yield full information labels; production feedback is noisy and sparse.

4.2. Feasibility and decision policy

At inference time, policy filters out incompatible models, a feasibility layer filters out likely violators of hard limits, and a scorer picks the least-utility feasible action. If none are feasible, the router falls back to a predefined safe action or asks for clarification. A cascade might first call a cheap model, then use a lightweight verifier to accept, repair, or escalate the response. This makes failures explicit and audit-able, and avoids the possibility that the learned score would silently override policy constraints.

4.3 Online adaptation

The dynamics of pricing, model versions, traffic mixes, and endpoint loads may change over time. Online adaptation should frequently update latency and cost estimators while cautiously updating quality models since labels are often sparse and noisy. Constrained contextual bandits can allocate a small, policy-constrained exploration budget to uncertain alternatives. Off-policy evaluation and canary traffic should precede broader rollouts. Drift monitors should watch for shifts in feature distributions, calibration error, model win rates, and constraint violations. The serving layer is as important as the statistical router: batching, memory management, and queuing impact the latency surface that the policy is trying to optimize (Crankshaw et al., 2017; Kwon et al., 2023).

5. Evaluation Framework

The evaluation of routers should compare policies on the same request set and candidate-output matrix in accordance with the reproducibility motivation of RouterBench (Hu et al., 2024). A single weighted score is insufficient since different weightings may mask unacceptable violations. Results should include cost-quality and latency-quality Pareto frontiers, per-domain slices, and confidence intervals. Baselines should include always-cheapest, always-strongest, random, static task rules, oracle routing, and at least one learned router. The oracle estimates headroom but should not be deployed as a performance baseline.

Table 2 Recommended evaluation measures

Dimension	Metric	Purpose
Quality	Task success, win rate, factuality, safety pass rate	Measures usefulness and risk by task
Cost	Mean and p95 cost per successful request	Captures token and escalation expenditure
Latency	TTFT and end-to-end p50, p95, p99	Exposes interactive and tail behavior
Constraints	Quality-floor and latency-SLO violation rates	Tests feasibility, not merely average utility
Calibration	Brier score or expected calibration error	Validates confidence-based decisions
Adaptation	Regret and recovery time after drift	Measures online resilience
Routing overhead	Router compute and added milliseconds	Prevents hidden coordination cost

Offline replay is necessary but not sufficient. It fixes outputs and cannot represent queue contention, retries, provider incidents, or changed decoding behavior. A production experiment should therefore log predicted and realized cost, latency, selected action, verifier result, fallback path, and outcome proxy. Quality judgments should combine automated task metrics with blinded human or preference evaluation, since benchmark scores may not match subjective utility. HELM's emphasis on broad, transparent evaluation supports this multidimensional approach (Liang et al., 2022).

6. Experimental Evaluation

6.1. Setup

To quantify the cost and latency trade-offs discussed above, the protocol of Section 5 is applied in an offline-replay setting on two public benchmarks. The candidate-output matrix is the one released with RouteLLM (Ong et al., 2024). It records the per-query correctness of a weak model, Mixtral-8x7B-Instruct (Jiang et al., 2024), and a strong model, GPT-4-1106-preview (OpenAI, 2023), on the 1,319 test problems of GSM8K (Cobbe et al., 2021) and on 14,042 MMLU questions across 57 subjects (Hendrycks et al., 2021). Because both outputs are recorded for every query, any routing policy can be replayed exactly on the same requests, in line with the reproducibility motivation of RouterBench (Hu et al., 2024).

Cost follows the pricing assumptions of Ong et al. (2024): USD 10 and USD 30 per million input and output tokens for the strong model, and USD 0.24 per million tokens for the weak model. Token counts are approximated as one token per four characters of the prompt and of the recorded response; for MMLU, whose responses are single answer letters, two output tokens are assumed. Latency was not recorded in the source data and is therefore modelled rather than measured. End-to-end latency is computed as the time to first token plus the number of output tokens divided by the decoding rate, using assumed values of 0.3 s and 90 tokens/s for the weak model and 0.8 s and 30 tokens/s for the strong model. The latency results therefore reflect the real distribution of response lengths under these stated service assumptions. A latency service-level objective (SLO) of 5 s is used for GSM8K.

The learned router is deliberately simple, so that the results reflect the constrained decision layer rather than a sophisticated predictor. It is an L2-regularized logistic regression on word and character n -gram TF-IDF features of the prompt, together with a subject indicator and the log prompt length, and it estimates the probability $\hat{q}$ that the weak model answers correctly. All predictions are out-of-fold from five-fold cross-validation. A request is sent to the strong model when $\hat{q}$ falls below a threshold θ , and sweeping θ traces the cost–quality frontier. The baselines are always-weak, always-strong, random routing, and an oracle that sends a query to the strong model only when the strong model is correct and the weak model is not. The oracle indicates headroom and is not a deployable policy.

To evaluate the constrained formulation of Section 3, the quality floor is set to 90% or 95% of the strong model's accuracy. On each of 200 random 50/50 splits, θ is chosen on one half (the calibration half) as the cheapest threshold that meets the floor, and the resulting policy is evaluated on the other half. Two threshold rules are compared. The point-estimate rule accepts the first θ whose calibration accuracy reaches the floor. The conservative rule additionally requires a one-sided 95% margin of $1.645 \cdot \sqrt{(2\hat{a}(1-\hat{a})/n)}$, where $\hat{a}$ is the calibration accuracy and n is the size of the calibration half. For each rule, the cost saving relative to always-strong is reported as the mean and the 2.5th–97.5th percentile interval across splits, together with the floor-violation rate, which is the share of splits in which test accuracy falls below the floor.

Table 3. Benchmark and single-model endpoint summary

Benchmark	Queries	Weak acc. (%)	Strong acc. (%)	Oracle acc. (%)	Cost per 1,000 queries, weak / strong (USD)	Mean latency, weak / strong (s)*	p95 latency, weak / strong (s)*
GSM8K	1,319	63.8	85.7	92.9	0.03 / 3.74	1.14 / 4.29	1.93 / 7.57
MMLU	14,042	68.1	80.6	85.9	0.03 / 1.23	0.32 / 0.87	0.32 / 0.87

Note. Correctness data from the RouteLLM release (Ong et al., 2024). The oracle routes a query to the strong model only when it is correct and the weak model is not. *Latency is modelled from recorded response lengths under the assumptions stated in Section 6.1, not measured.

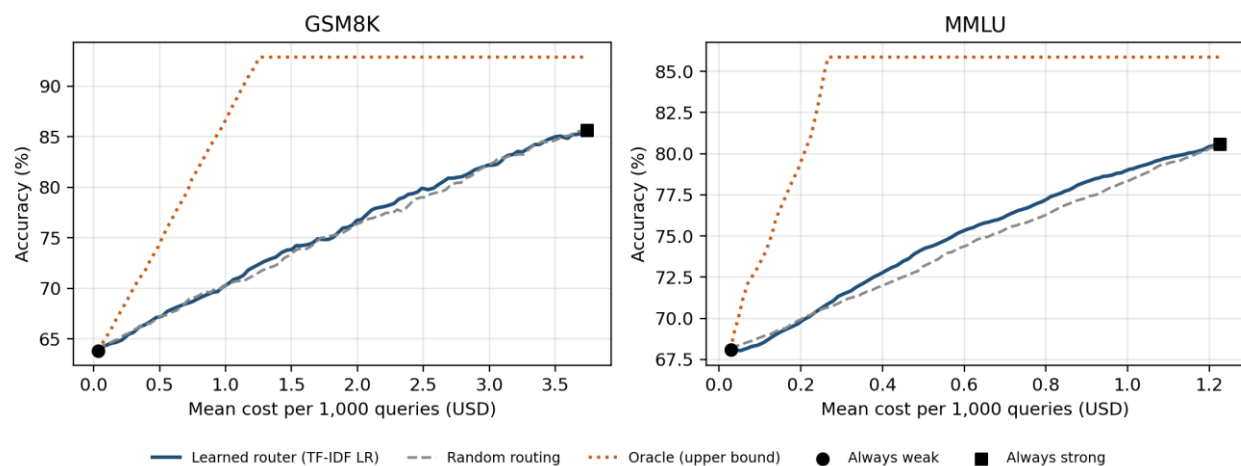


Figure 2: Cost-quality Pareto frontiers on GSM8K and MMLU (offline replay). Each curve sweeps the share of queries sent to the strong model; markers show the single-model endpoints. The oracle curve is an upper bound and not a deployable policy.

6.2. Results

Table 4. Constrained operating points (mean over 200 calibration/test splits)

Benchmark	Quality floor	Threshold rule	Test accuracy (%)	Strong-model share (%)	Cost saving vs. always-strong (%) [95% interval]	Floor violation rate (%)	Mean latency (s)*
GSM8K	90%	Point estimate	77.1	49.7	45.0 [35.9, 51.2]	48.0	2.84
GSM8K	90%	Conservative	80.5	66.8	28.5 [20.0, 36.1]	0.5	3.37
GSM8K	90%	Random (expected)	77.1	60.8	38.9	—	3.05
GSM8K	95%	Point estimate	81.3	71.2	24.4 [16.7, 31.8]	49.5	3.50
GSM8K	95%	Conservative	84.6	90.3	7.7 [2.8, 12.0]	0.0	4.04
GSM8K	95%	Random (expected)	81.4	80.4	19.4	—	3.67
MMLU	90%	Point estimate	72.5	25.2	68.8 [65.3, 71.8]	53.5	0.46

Benchmark	Quality floor	Threshold rule	Test accuracy (%)	Strong-model share (%)	Cost saving vs. always-strong (%) [95% interval]	Floor violation rate (%)	Mean latency (s)*
MMLU	90%	Conservative	73.8	31.6	61.5 [58.8, 64.1]	1.0	0.49
MMLU	90%	Random (expected)	72.5	35.5	63.0	—	0.52
MMLU	95%	Point estimate	76.6	52.4	39.8 [36.6, 43.6]	46.5	0.61
MMLU	95%	Conservative	77.7	62.1	30.8 [28.1, 33.9]	0.0	0.66
MMLU	95%	Random (expected)	76.6	67.8	31.5	—	0.69

*Note. The quality floor is a fraction of the strong model's test accuracy. Random (expected) gives the analytic expectation for random routing calibrated to meet the floor exactly; by construction it also falls below the floor in about half of the splits. *Modelled latency.*

Table 5. Router diagnostics

Benchmark	APGR, learned router	APGR, random	APGR, oracle	Brier score, router	Brier score, base rate	ECE	Router overhead (ms/query)
GSM8K	0.576	0.534	1.137	0.230	0.231	0.048	0.33
MMLU	0.595	0.513	1.296	0.186	0.217	0.026	0.56

Note. APGR is the area under the curve of performance-gap recovered, (accuracy – weak accuracy)/(strong accuracy – weak accuracy), plotted against the strong-model share (Ong et al., 2024). Random routing has an expected APGR of 0.5, and values above 1 indicate accuracy above the strong model. Brier scores and ECE refer to the predicted probability that the weak model is correct. Router overhead was measured on a CPU and includes feature extraction.

Four findings emerge. First, routing yields substantial but benchmark-dependent savings. With point-estimate thresholds, the learned router saves 45.0% of the always-strong cost on GSM8K and 68.8% on MMLU at the 90% floor, and 24.4% and 39.8% at the 95% floor. Random routing calibrated to the same floors would save 38.9%, 63.0%, 19.4%, and 31.5%, so the learned predictor adds roughly 5 to 8 percentage points. The gain is larger on MMLU, where the predictor improves on the base-rate Brier score (0.186 against 0.217), than on GSM8K, where it barely does (0.230 against 0.231) and the frontier in Figure 2 lies close to the random line. This supports the argument that robust gains depend on accurate quality estimates more than on the routing mechanism itself, and it is consistent with the smaller savings reported for GSM8K and MMLU than for conversational benchmarks by Ong et al. (2024).

Second, average savings hide constraint violations. Thresholds tuned to just meet the floor on a calibration sample violated it on the held-out half in 46.5% to 53.5% of the splits, because sampling noise places the realized accuracy on either side of the target. The conservative rule reduced the violation rate to at most 1.0% while still saving 61.5% (MMLU) and 28.5% (GSM8K) at the 90% floor. At the stricter 95% floor on GSM8K, however, reliability is expensive: the saving falls to 7.7% and 90.3% of the requests reach the strong model. This price of reliability is why Section 5 asks for violation rates to be reported alongside average utility.

Third, the headroom is large. The oracle exceeds the strong model's accuracy (92.9% against 85.7% on GSM8K, and 85.9% against 80.6% on MMLU) while sending only 29.0% and 17.8% of queries to the strong model, because the weak model is correct on 7.2% and 5.3% of the queries that the strong model gets wrong. The gap between the learned

and oracle frontiers, with APGR of 0.58–0.60 against 1.14–1.30, is therefore mainly a gap in quality prediction, not in model capability.

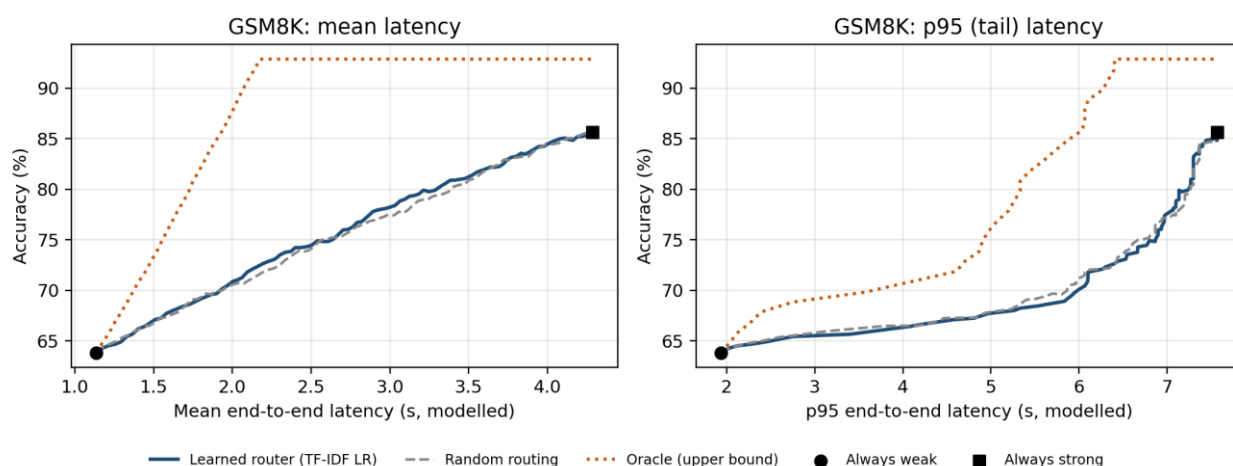


Figure 3: Latency–quality trade-off on GSM8K under the modelled latency of Section 6.1. Left: mean end-to-end latency; right: p95 tail latency.

Fourth, routing improves average latency far more than tail latency. On GSM8K at the 90% floor, the point-estimate policy lowers the mean modelled latency from 4.29 s to 2.84 s and reduces the share of requests exceeding the 5 s SLO from 27.2% to 17.3%. The p95 latency, however, only falls from 7.57 s to 6.96 s (Figure 3, right). As long as more than about 5% of the requests are sent to the strong model, the tail is set by the strong model’s longest responses. Meeting a tail-latency objective therefore requires latency to be predicted per request and enforced as a feasibility constraint (Section 4.2), rather than reduced on average, which is consistent with the tail-at-scale argument of Dean and Barroso (2013). The router itself adds less than 1 ms per request, which is negligible next to model latency.

6.3. Limitations of the Evaluation

The evaluation uses a two-model portfolio and offline replay of fixed outputs. It therefore does not capture queuing, retries, provider incidents, or the multi-model, multi-sample action space described in Section 3. Latency is modelled from response lengths rather than measured, token counts are approximated, and prices reflect the historical assumptions of Ong et al. (2024). The source data contain no verifier signal, so cascades with verification were not evaluated. Finally, the router is intentionally simple. The results should therefore be read as a reproducible baseline for the proposed protocol, not as a claim of state-of-the-art routing performance.

7. Discussion

Adaptive routing introduces novel failure modes. First, feedback is biased: observing the chosen model more often than rejections can exacerbate early model preferences. This can be mitigated with periodic shadow sampling and propensity-aware learning. Second, quality judges could be biased toward verbose or stylistically similar responses. Auditing by humans and heterogeneous judges offer some protection. Third, provider outages may divert traffic to a model significantly different from their usual mix. Capacity-aware policies should reserve headroom and specify degraded mode prior to incidents.

Routing might undermine safety if cheap models are selected for apparently trivial but high-impact prompts. Risk classification therefore should be independent of predicted difficulty. Sensitive requests could have conservative floors, limited model pools, and enhanced verification. Privacy attributes, regional processing rules, and data retention policies should be encoded in the hard feasibility layer, rather than a weighted objective in which costs might offset potential violations.

Finally, the best routing portfolio is diverse beyond just size: adding near-duplicates increases estimation challenge with little added oracle headroom. Policies should periodically prune dominated portfolio members while retaining specialists whose conditional advantage is readily measurable. Transparent domain-action representations, as proposed by Arch-Router, can help operators understand why a candidate is retained valuable (Tran et al., 2025).

8. Conclusion

Adaptive multi-LLM routing is best viewed as a constrained decision-making under uncertainty. The production objective is not to select the nominally best model, but to choose the least costly action that is likely to satisfy a request-specific quality floor and latency objective while respecting hard governance rules. Recent routers illustrate that prompt-conditioned choices, cascades, contrastive model representations and adaptive test-time compute can produce substantial efficiency gains. Reliable deployment requires well-calibrated predictors, tail-aware latency models, explicit fallbacks, controlled exploration, and continuous drift monitoring. The offline replay on GSM8K and MMLU supports this view: conservative thresholds retained cost savings of 29–62% at a 90% quality floor with almost no floor violations, whereas thresholds tuned to just meet the floor violated it in about half of the splits, and gains in mean latency did not carry over to tail latency. Future research should prioritize joint routing/scheduling, causal correction for selective feedback, robustness under provider change, and standardized evaluation of constraint violations, enabling routing to move from benchmark optimization to building dependable infrastructure for heterogeneous AI systems.

References

- [1]. Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.
- [2]. Chen, L., Zaharia, M., & Zou, J. (2023). FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*.
- [3]. Chen, S., Jiang, W., Lin, B., Kwok, J. T., & Zhang, Y. (2024). RouterDC: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37.
- [4]. Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., & Stoica, I. (2017). Clipper: A low-latency online prediction serving system. *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation*, 613–627.
- [5]. Dean, J., & Barroso, L. A. (2013). The tail at scale. *Communications of the ACM*, 56(2), 74–80.
- [6]. Ding, D., Mallick, A., Zhang, S., Wang, C., Madrigal, D., Garcia, M. D. C. H., Xia, M., Lakshmanan, L. V. S., Wu, Q., & Rühle, V. (2025). BEST-Route: Adaptive LLM routing with test-time optimal compute. *arXiv:2506.22716*.
- [7]. Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., & Upadhyay, S. K. (2024). RouterBench: A benchmark for multi-LLM routing system. *arXiv:2403.12031*.
- [8]. Jiang, D., Ren, X., & Lin, B. Y. (2023). LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. *Proceedings of ACL 2023*.
- [9]. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., & Stoica, I. (2023). Efficient memory management for large language model serving with PagedAttention. *Proceedings of the 29th ACM Symposium on Operating Systems Principles*.
- [10]. Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th International Conference on World Wide Web*, 661–670.
- [11]. Liang, P., Bommasani, R., Lee, T., et al. (2022). Holistic evaluation of language models. *arXiv:2211.09110*.
- [12]. Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2024). RouteLLM: Learning to route LLMs with preference data. *arXiv:2406.18665*.

- [13]. Tran, C., Paracha, S., Hafeez, A., & Chen, S. (2025). Arch-Router: Aligning LLM routing with human preferences. arXiv:2506.16655.
- [14]. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021). Training verifiers to solve math word problems. arXiv:2110.14168.
- [15]. Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021). Measuring massive multitask language understanding. International Conference on Learning Representations (ICLR 2021).
- [16]. Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., et al. (2024). Mixtral of experts. arXiv:2401.04088.
- [17]. OpenAI. (2023). GPT-4 technical report. arXiv:2303.08774.