MODEL METROPOLIS: A DIGITAL-TWIN CONTEXTUAL BANDIT FOR DIFFICULTY-AWARE LLM ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic LLM routing has recently emerged as an effective way to balance model quality and cost across diverse queries. While real-world deployments increasingly demand adaptive routing, existing methods often rely on offline estimates and lack fine-grained evidence of model performance. We introduce Model Metropolis, a live digital-twin framework for difficulty-aware LLM routing. We leverage time-decayed evidence from observed outcomes to maintain capability- and difficulty-specific delivery states for each model. A contextual-bandit router then uses these states to adjust the quality requirement based on query difficulty and to balance estimated delivery against cost among evidence-supported models. Our experiments demonstrate three key findings. First, difficulty-aware routing substantially improves delivery on challenging queries while preserving cost efficiency. We demonstrate this on a production gateway of 30 models, where hard and expert delivery improve from 73% to 91% and from 69% to 85%, with hard-query routing reaching 91% delivery at $0.44\times$ the all-strong cost. Second, we show that the learned routing state generalizes to held-out benchmarks when the calibration evidence covers the target task family, achieving competitive cost–delivery trade-offs across public coding and general-purpose benchmarks. Third, we demonstrate that the framework extends to both escalation control and pre-generation routing. A pool ceiling reduces gateway errors from 62% to 0% on unwinnable queries. Overall, our results establish that continuously observed, evidence-updated model states provide a practical basis for adaptive LLM routing.

1 INTRODUCTION

Large language models (Vaswani et al., 2017; Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023) differ by orders of magnitude in size, price, and latency (Kaplan et al., 2020). Moreover, because they are trained on different data and architectures, their competence varies substantially across tasks. Therefore, dynamic routing is a challenging model-selection problem under competing quality and cost objectives. The strongest model is unaffordable at scale, while cheaper models often fail on the hard, multi-step queries users notice. This creates a fundamental trade-off between delivery quality and cost: Routing every query to the strongest model is expensive, while always choosing cheaper models sacrifices delivery on difficult queries. To address this trade-off, we develop an LLM router that continually learns and updates from observed model outcomes: it first identifies each query's capability and difficulty, then decides which model should handle it, routing easier queries to cheaper models and harder queries to more reliable, stronger ones.

Existing routing methods approach this trade-off from several directions. HybridLLM and RouteLLM learn when to dispatch a request to a stronger or weaker model (Ding et al., 2024; Ong et al., 2025), but largely formulate routing as choices between a small candidate set. RouterDC and GraphRouter instead predict query–model compatibility from learned representations or interaction structure (Chen et al., 2024; Feng et al., 2025). AutoMix and Self-REF take a complementary cascading approach to decide whether to accept an initial response or escalate it to a stronger model (Aggarwal et al., 2024; Chuang et al., 2024). However, a common limitation remains: routing decisions are not continuously grounded in what each model has actually delivered at each difficulty and what it costs per request. As shown in Fig. 1, a cost-aware router can overpay

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

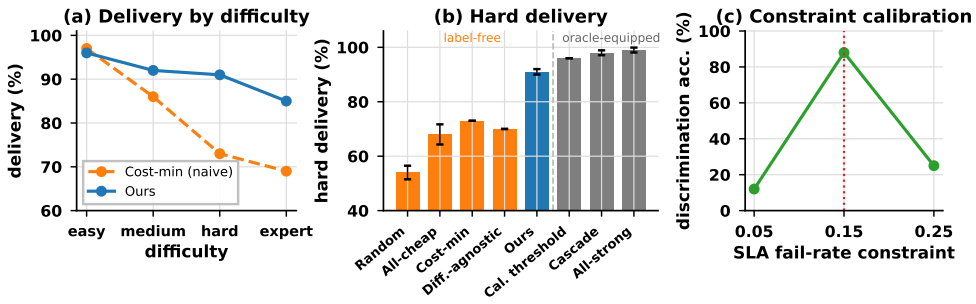


Figure 1: **Difficulty-aware routing recovers hard-query delivery without abandoning cost savings.** **(a)** Executable delivery by difficulty: naive cost-minimizing routing collapses on hard/expert queries while ours tracks strong-model quality. **(b)** Hard-query delivery of all eight policies, ± 1 s.d. over three repeats. **(c)** Discrimination accuracy versus the fail-rate constraint.

by $11\times$–$35\times$ on easier queries yet still lose delivery quality on the hard tail. Prices and aggregate capability priors can mis-rank models, while a single quality threshold cannot distinguish easy requests from hard ones. Effective routing, therefore, requires online, evidence-calibrated estimates of delivery and per-request cost.

In this paper, we introduce MODEL METROPOLIS, a routing framework organized around a digital twin (Grieves & Vickers, 2016; Tao et al., 2018; Almasan et al., 2022) of the model ecosystem: providers are buildings, models are floors, capabilities are rooms, and within a room competence is a difficulty-stratified, time-decayed Beta posterior, a capability boundary, updated from an executable delivery signal that is grounded in executable task success (Chen et al., 2022; Li et al., 2022). A Thompson-sampling (Thompson, 1933) bandit over this twin raises the quality bar with difficulty, uses evidence-gated qualification, and balances sampled delivery against cost where uncertainty matters, while a bounded exploration budget keeps the twin state fresh. In summary, our contributions are:

- We introduce a digital-twin routing framework that learns each model's reliability and cost across difficulty levels, continually updates these estimates from observed outcomes, and uses them to select the most appropriate model for each query, improving hard-query delivery by 18 points and achieving 88% discrimination accuracy (DA) on a 30-model gateway.
- We show that Model Metropolis adapts to changing workloads and model pools without retraining the routing policy: capability- and difficulty-conditioned delivery states transfer to held-out workloads once the target task family is set. In contrast, newly added models can be initialized from priors and updated as evidence arrives.
- We extend Model Metropolis with two complementary controls: a ceiling-aware controller that stops futile escalation when the pool has little chance of success, and a pre-generation gate that reads an open-source model's activations to decide whether to handle a request locally. Together, they reduce gateway errors from 62% to 0% and lift local-model delivery from 69% to 83%.

While our experiments focus on LLM routing, Model Metropolis's success suggests a broader shift from fixed model selection to continuously observed, evidence-updated model ecosystems that adapt as models, workloads, costs, and deployment conditions evolve.

2 RELATED WORK

Learned query routers. Early work on LLM routing framed model selection as a quality–cost trade-off between a small number of candidate models. Inspired by the complementary capabilities of different LLMs, recent methods learn to predict the model best suited to an incoming request. The most related are RouteLLM and HybridLLM (Ong et al., 2025; Ding et al., 2024), which learn strong–weak routing policies from preference or performance data. RouteLLM focuses on pairwise model selection, while HybridLLM learns a router over a fixed model pair. Other approaches model query–model compatibility more explicitly, including RouterDC (Chen et al., 2024), GraphRouter (Feng et al., 2025). These methods typically require a new profiling or training pass as the pool or workload changes. In contrast, we maintain difficulty-conditioned evidence for each model and update it continuously from deployment outcomes.

Cascades, ensembles, and online selection. Our work is also related to recent progress in cascaded and online model selection. Cascading methods such as FrugalGPT (Chen et al., 2023), AutoMix (Aggarwal et al., 2024), and language-model cascades (Dohan et al., 2022; Jitkrittum et al., 2023; Gupta et al., 2024; Yue et al., 2023) use response quality or uncertainty to decide whether to defer or escalate, while ensemble methods such as LLM-Blender (Jiang et al., 2023), Mixture-of-Agents (Wang et al., 2025), and expertise fusion (Wang et al., 2024a) combine outputs from multiple models. While prior LLM routing work focuses primarily on the training selection rule itself, we instead propose a live digital twin of the model ecosystem, in which providers, models, capabilities, and difficulty levels form a continuously synchronized city-like state for routing.

Digital twins and internal representations. A key question we pose is whether an LLM ecosystem can be represented as a continuously updated state for routing. Recent studies on digital twins synchronize virtual counterparts with observations from evolving physical and networked systems (Grieves & Vickers, 2016; Tao et al., 2018; Nguyen et al., 2021; Almasan et al., 2022). Several works have also investigated what information internal model representations can recover (Alain & Bengio, 2016; Azaria & Mitchell, 2023; Burns et al., 2022; Kadavath et al., 2022; Tenney et al., 2019; Geva et al., 2021), showing that hidden states encode task-relevant signals at different depths. We build on this work by treating delivery posteriors as the LLM ecosystem's synchronized state and using capability-localized activations as a pre-generation routing signal.

3 Difficulty-Aware Routing

Formulation. Consider a pool $\mathcal{M}$ of models served through one gateway, each mapping a query q to an answer $m(q)$ at price $\mathrm{cost}(m)$. Let $\mathcal{C}$ denote the set of capabilities and $\mathcal{D} = \{\text{easy}, \text{medium}, \text{hard}, \text{expert}\}$ the ordered set of difficulty levels. For each query, a capability $c(q)$ and difficulty $d(q)$ are inferred by lightweight classifiers (Appendix C). Unlike proxy scores, delivery is a hard binary gate: delivery means that the model's response fully satisfies the user's request. For example, for a code-generation request with hidden tests T_q,

$$g(q, m) = \mathbb{1}\big[\textsc{extract}(m(q)) \text{ passes all of } T_q\big] \in \{0, 1\}. \tag{1}$$

Unlike learned rewards, delivery is defined by explicit, task-specific criteria and may rely on multiple labels or verification signals. We therefore define quality as a constraint and cost as the sole objective to minimize. With $\widehat{g}_d(m, c)$ denoting the estimated delivery probability of model m for capability c at difficulty d,

$$R(q) = \arg\min_{m \in \mathcal{F}_d(c)} \mathrm{cost}(m), \qquad \mathcal{F}_d(c) = \big\{\, m : \widehat{g}_d(m, c) \geq \tau_d \,\big\}, \tag{2}$$

where the qualified set $\mathcal{F}_d(c)$ must satisfy a delivery threshold that rises with difficulty, $(\tau_{\text{easy}}, \tau_{\text{medium}}, \tau_{\text{hard}}, \tau_{\text{expert}})$. The rising threshold excludes cheap but unreliable models on harder queries while retaining them when they are sufficient for easier ones. If $\mathcal{F}_d(c) = \varnothing$, R falls back to the best evidence-supported alternative, as detailed in Sec. 4.2. Equation (2) defines the underlying constrained routing objective. Because delivery probabilities are uncertain online and evidence is incomplete, Sec. 4.2 operationalizes this objective with posterior sampling.

This formulation raises two practical questions. (i) Evidence. How can the gateway determine whether a model has sufficient evidence to clear the delivery requirement at a given capability and difficulty? (ii) Adaptation. How can these estimates remain valid as model performance, cost, and availability change over time? Sec. 4 addresses both through the live digital twin.

4 Model Metropolis

Fig. 2 provides an overview of how Model Metropolis represents model capability and turns this evidence into a routing decision.

4.1 The Digital Twin: A Live Evidence Map of Model Capability

A digital twin is a virtual counterpart of a real system that stays synchronized with its observed state and uses it for analysis and decision-making. We apply this idea to the LLM ecosystem by maintaining a live representation of how models perform across capabilities and difficulty levels. We organize the twin as a graph of buildings (providers), floors (models), and rooms (model×capability), with cross-provider skybridges recording learned handoffs used during escalation.

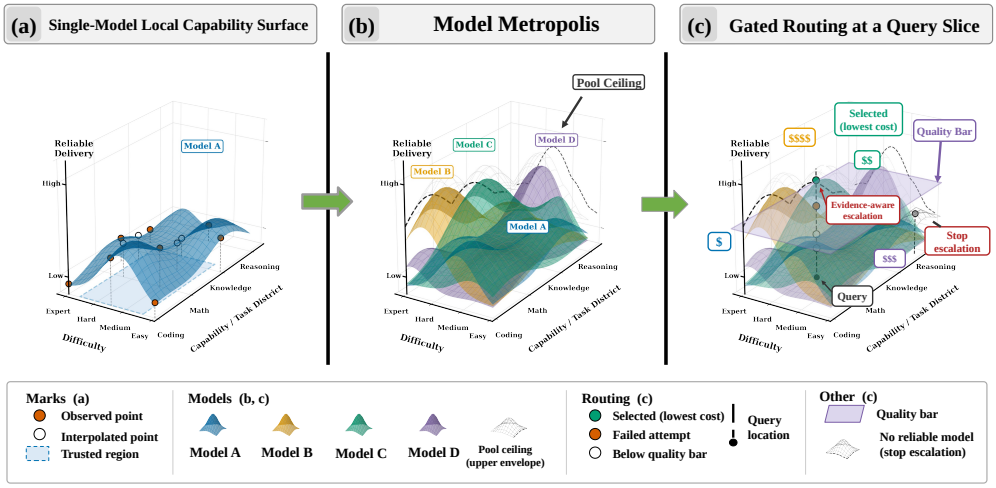


Figure 2: The Model Metropolis demonstrates the process from observed model evidence to capability mapping for reliable low-cost routing.

Each room records how often a model succeeds or fails on queries of different difficulty levels and continually updates how reliably the model can handle similar tasks. Specifically, for each difficulty d, the room maintains a Beta posterior over delivery using decayed pseudo-counts (α_d, β_d). This time-decayed evidence weights recent outcomes more heavily, allowing the twin to track changes in model behavior without retraining.

Boundary estimation. We treat a model's capability boundary as a continuously updated reliability record: past successes and failures affect how much the system should trust the model, while older evidence gradually loses influence. For every outcome $g \in \{0, 1\}$ at difficulty d, we decay the existing evidence before incorporating the new observation:

$$\alpha_d \leftarrow \gamma\, \alpha_d + g, \qquad \beta_d \leftarrow \gamma\, \beta_d + (1 - g), \qquad \gamma = 0.96, \tag{3}$$

So recent observations contribute more strongly than older ones, allowing the twin to follow changes in model behavior without being dominated by outdated records, as in discounted estimators for non-stationary bandits (Garivier & Moulines, 2011; Raj & Kalyani, 2017).

For an untested cell, we still require an initial estimate before sufficient delivery evidence has been observed. We use the benchmark score $s_m \in [0, 1]$ as weak initial evidence and discount it according to difficulty:

$$(\alpha_d^0, \beta_d^0) = \big(\kappa\, s_m^{\eta_d},\ \kappa\,(1 - s_m^{\eta_d})\big), \qquad \kappa = 3, \tag{4}$$

where $\eta_d = (0.5, 1.0, 1.6, 2.4)$ from easy to expert. Thus, leaderboard performance provides only limited initial credit, and this credit weakens as the query becomes harder. A model must therefore earn a high boundary on hard and expert queries through observed delivery. The resulting posterior estimate is

$$\widehat{g}_d(m, c) = \frac{\alpha_d + \alpha_d^0}{\alpha_d + \beta_d + \alpha_d^0 + \beta_d^0}. \tag{5}$$

A uniform prior, in contrast, would make an untested model on a hard query appear as plausible as one supported by observed delivery.

Evidence Refinement. A tiered room tells us how reliably a model performs on a broad class of problems, but queries within the same capability and difficulty can still differ in important ways. To refine this coarse estimate into a query-specific one, we additionally use the calibration queries most similar to q as local evidence. Their outcomes indicate how the model performs on problems close to the current query and are weighted by similarity before being combined with the tier-level estimate, which serves as the prior:

$$\widehat{g}_d(m, q) = \frac{\sum_{q' \in \mathcal{N}(q)} w_{qq'}\, g(q', m) + \kappa\, \widehat{g}_d(m, c)}{\sum_{q' \in \mathcal{N}(q)} w_{qq'} + \kappa}, \tag{6}$$

where $\mathcal{N}(q)$ contains the k nearest calibration queries and $w_{qq'}$ denotes their cosine similarity. The resulting estimate therefore combines broad capability and difficulty evidence with fine-grained evidence from nearby queries. When no local neighbors are available, $\mathcal{N}(q) = \varnothing$, and the estimate reduces to the tier-level room alone.

4.2 The Contextual-Bandit Router

To route over the live twin, we need a policy that decides, for each incoming query, which model should handle it. Given a query with capability c and difficulty d, the router considers every online model serving c, draws a Thompson sample (Chapelle & Li, 2011; Russo et al., 2017), and reads the posterior mean $\bar{p}_m = \widehat{g}_d(m, c)$ together with the observed evidence $n_m = \alpha_d^{(m)} + \beta_d^{(m)}$:

$$p_m \sim \mathrm{Beta}\big(\alpha_d^{(m)} + \alpha_d^0,\ \beta_d^{(m)} + \beta_d^0\big). \tag{7}$$

The Thompson sample p_m is used when uncertainty matters: it ranks candidates on hard and expert queries, in cells without calibrated per-query cost. The posterior mean $\bar{p}_m$ and accumulated evidence n_m determine whether to trust a model, while the exploration budget described next handles targeted exploration of under-sampled cells.

Evidence Collection. The effectiveness of online routing depends on whether the twin continues to collect evidence for under-observed models. With difficulty-dependent probability ϵ_d, and only when cold cells remain, the router selects the least-observed cell, actively probing models whose capability is still uncertain, especially strong models on hard problems. As these cells accumulate sufficient delivery evidence, exploration naturally decreases and the routing budget returns to normal task handling.

Exploit. Outside targeted exploration, the router first applies the evidence floor $n_m \geq \nu_d$ so that unproven models cannot be trusted on the hard tail. It then ranks the remaining candidates using a difficulty-dependent value rule. On hard and expert queries, the router uses the Thompson-sampled value $p_m - \lambda\,\mathrm{cost}(m)$. On easy and medium queries with calibrated costs, it instead uses the posterior-mean value $\bar{p}_m - \lambda\,\mathrm{cost}(m)$ and selects the cheapest model within a 0.05 tie band.

The selected model is then checked against the mean-based quality bar τ_d. If its posterior mean falls below the bar, the router falls back to the reliable frontier when available. Thus, Thompson sampling can change the ranking under uncertainty. Still, a favorable draw alone cannot clear the quality gate.

Failure Handling and Escalation. The router normally makes a single model call and escalates only after an observed failure. For hard and expert queries, escalation follows a strength-first rule: rank untried models by the Thompson sample p_m and break ties by lower cost. For easy and medium queries, the router first restricts escalation to untried models with $\bar{p}_m \geq 0.85\tau_d$ and then selects the model with the largest $p_m/\mathrm{cost}(m)$. This keeps the quality requirement mean-based while using posterior uncertainty to order the remaining escalation choices, avoiding preemptive calls to multiple expensive models.

4.3 Ceiling-Aware Routing

Escalation is useful only when a stronger model has a reasonable chance of delivering after the routed model fails. Before repeatedly escalating, we therefore first ask whether the query may already lie beyond the reliable capability of the entire model pool. For some hard cells, even the strongest available model succeeds only about half the time, so repeated retries provide little benefit and can create an escalation storm on the shared gateway.

To identify such cases, the twin estimates an **honest pool ceiling** from models with at least $n_{\text{ceil}}{=}15$ real observations. Rather than using the raw delivery estimate $\hat{p}_m$, we use its Wilson lower bound (Wilson, 1927), $\tilde{p}_m = \mathrm{WLB}(\hat{p}_m, n_m, z)$ (Appendix M), and define

$$d_{\text{ceil}}(c, d) = \max_{m:\, n_m \geq n_{\text{ceil}}} \tilde{p}_m, \qquad m^\star = \arg\max_{m:\, n_m \geq n_{\text{ceil}}} \tilde{p}_m, \tag{8}$$

where $m^\star$ denotes the reliable frontier. The lower bound prevents a model supported by only a few favorable outcomes, such as $15/15$, from overstating what the pool can reliably deliver.

We then use this ceiling to decide whether further escalation is worthwhile. If $d_{\text{ceil}} < \tau_{\text{ceil}}$, with $\tau_{\text{ceil}} = 0.65$ in our experiments, the query is sent directly to $m^\star$ with no escalation budget. Other-

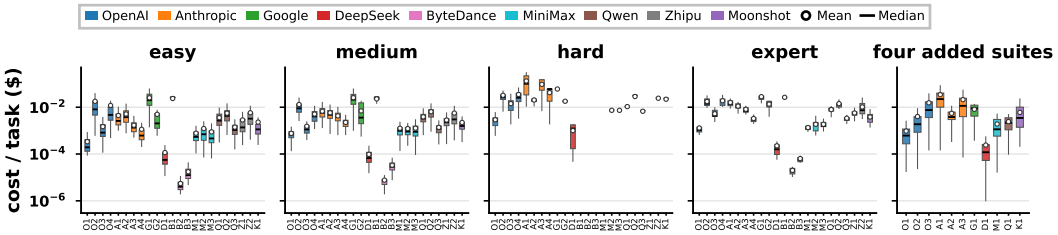


Figure 3: **Per-task cost of every measured model, by difficulty tier**.

wise, the qualified-set router proceeds unchanged. Therefore, escalation is allowed only when the model pool still has a reasonable chance of success.

4.4 Pre-Generation Delivery Estimation

Open-source models offer a low-cost alternative to paid frontier APIs. Recent improvements have made local deployment increasingly practical. To exploit this option, we consider whether an open-source model m_0 can reveal, before generation, whether it is likely to handle the current query successfully. Our twin normally estimates delivery from observed outcomes. We also use the model's internal state as a pre-generation signal. If the estimate is high, we handle the request locally. Otherwise, we send it directly to the paid strong model. A single forward pass exposes hidden activations h_r at selected layers, and a linear probe maps these activations to a pre-generation delivery estimate. Specifically, we combine this signal with the twin's prior through $d_{r,m_0} = \sigma\big(\mu_{m_0} + w^\top h_r\big)$, where μ_{m_0} is the twin's difficulty-conditioned prior and $w^\top h_r$ provides a per-request correction.

While no probe or read is available, $w^\top h_r \equiv 0$, so the gate falls back to the twin prior. The twin stores, for each capability, the layers its probe uses. The read-path gate sends the request to the local model when $d_{r,m_0} \geq \tau_{\text{gate}}$ and otherwise sends it to the paid gateway before generation. This provides a pre-generation routing signal without requiring a second model call.

5 Experiments

Setup. We evaluate Model Metropolis on a production gateway containing 30 models from 9 providers (Achiam et al., 2023; Team et al., 2023; Liu et al., 2024; Bai et al., 2023) (Appendix A). We evaluate on HumanEval+ (164), MBPP (257), MMLU-Pro, GPQA-Diamond, BiGGen-Bench, LiveBench, Chatbot Arena, RouterBench, LiveCodeBench, SWE-bench Verified, and a set of benchmarks we constructed. The benchmark suite contains 96 graded coding tasks across four facets, each equipped with hidden tests (Appendix B). Difficulty experiments use a 115-task subset labeled easy/medium/hard/expert ($27/26/51/11$) and run the router for 5 rounds (575 gateway calls) with the values of Sec. 4 (Appendix N).

Metrics. We report (i) the delivery rate $\text{Deliver}(Q) = \frac{1}{|Q|}\sum_{q\in Q} g(q, R(q))$ per difficulty; (ii) **cost**, the mean price per query $\text{Cost}(R) = \frac{1}{|Q|}\sum_q \text{cost}(R(q))$, normalized to the all-strong policy. (iii) discrimination accuracy, the fraction of queries sent to a model that demonstrably delivers that facet at that difficulty, $\mathcal{G}_d(c) = \{m : \widehat{g}_d(m, c) \geq \tau_d\}$: $\text{DA} = \frac{1}{|Q|}\sum_{q\in Q} \mathbb{1}\big[R(q) \in \mathcal{G}_{d(q)}(c(q))\big]$.

Baselines. We compare eight policies on the identical stream and gate: Random, All-cheap, Cost-min, a single fixed bar with sample-based feasibility, Calibrated threshold, the learned-threshold family (Ong et al., 2025; Ding et al., 2024) given a manual strong/weak tiering. Cascade (Chen et al., 2023; Aggarwal et al., 2024), cheapest first with one escalation on a verified failure.

5.1 Does difficulty-aware routing recover hard delivery?

Q. Does difficulty-aware routing recover hard delivery without giving up easy-query savings?

A. Yes. Hard delivery rises from 73% to 91% while easy queries stay on the cheapest model.

Fig. 1(b) and Tab. 9 (Appendix N) split the eight policies into label-free and oracle-equipped baselines. On hard tasks, label-free policies exhibit a clear ablation progression: All-cheap and naive cost-minimization achieve only 68–73%, while a difficulty-agnostic bandit reaches 70%. Adding difficulty conditioning boosts hard delivery to 91% ($91.2 \pm 1.0\%$), a +21 percentage-point improvement. Fig. 4 confirms that delivery is uniform on easy/medium tasks but varies (0%–86%) on hard

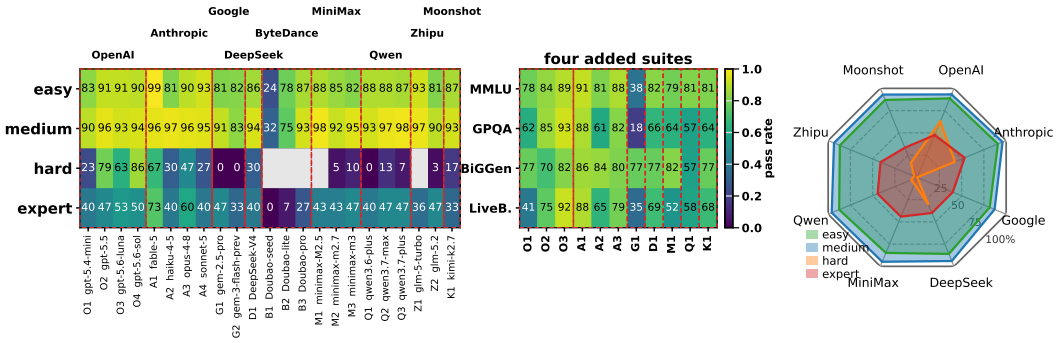


Figure 4: **Measured delivery matrix across models and difficulty levels.** Left: delivery for each model–difficulty cell from the calibration sweep. Middle: delivery for each model on the four added suites (Appendix H). Right: mean delivery rate per provider. Pairwise boundary agreement is reported in Appendix J.

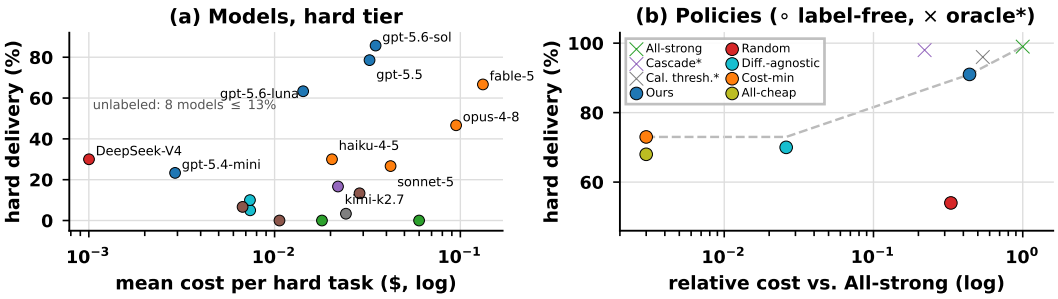


Figure 5: **Cost–delivery landscape on the hard tier.** (a) Mean cost per hard task versus hard-tier delivery for every measured model. (b) The eight routing policies (∘ label-free, × oracle-equipped).

tasks. Ultimately, our router sends easy/medium inputs to cheap models and hard/expert inputs to the cheapest strong model (Tab. 11).

Ablation. Naive routing fails for three reasons that compound (Tab. 12, Appendix N): a single low bar lets 55%-deliverers qualify on hard. A Thompson sample for feasibility lets wide-posterior cold models pass optimistically. The evidence floor ν_d excludes the unproven and lifts hard routing from 73% to 91%. **Constraint calibration.** Discrimination is strikingly sensitive to the fail-rate constraint (Fig. 1c; Tab. 17): at a too-strict 0.05 no model clears the bar and DA is 12%, at a too-loose 0.25 weak-but-cheap models slip in (25%), at the calibrated 0.15 the router separates proven from weak models and reaches 88%.

5.2 Where does the router sit on the cost-delivery frontier?

Q. Does the recovered hard delivery come at a reasonable price?

A. Yes. Ours is on the label-free efficient frontier.

Fig. 5(b) places all 8 policies on one cost-delivery basis (Tab. 18): ours delivers the best label-free hard rate at any cost below the strong tier; cascade's sharper point (98% at 0.22×) needs an inference-time oracle and a second call. Fig. 5(a) shows on the hard frontier models sit 50 points above cheaper ones.

5.3 Does routing transfer to public benchmarks?

Q. Does the router transfer to public benchmarks graded by their official tests?

A. Yes. It matches the frontier on LiveCodeBench and surpasses the strong model on SWE-bench Verified.

Routing quality scales with calibration evidence. Having established the routing framework, we next assess how its decisions improve as more evidence becomes available. Fig. 6 shows held-out delivery and cost as the number of calibration tasks per model increases. On the easy and medium tiers, evidence allows cheaper models to qualify, reducing cost by more than an order of magnitude

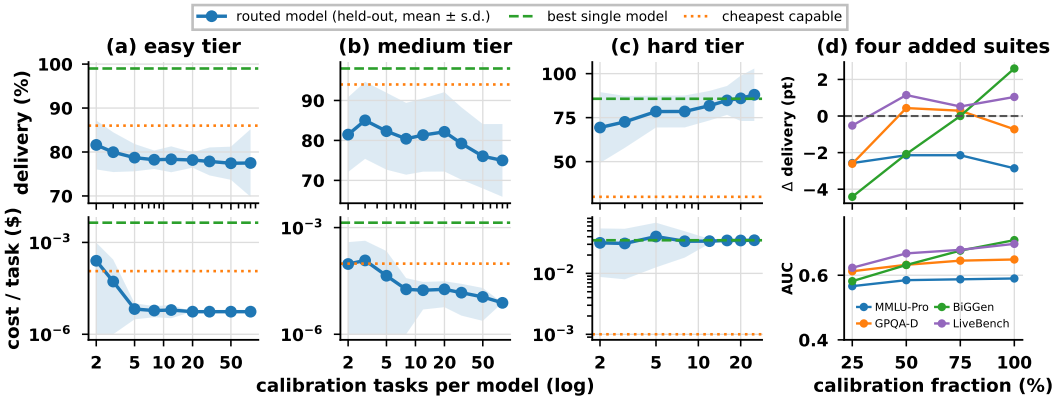


Figure 6: **Routing quality scales with calibration evidence.** We evaluate held-out delivery (top) and cost (bottom) as the number of calibration tasks per model increases on a log scale.

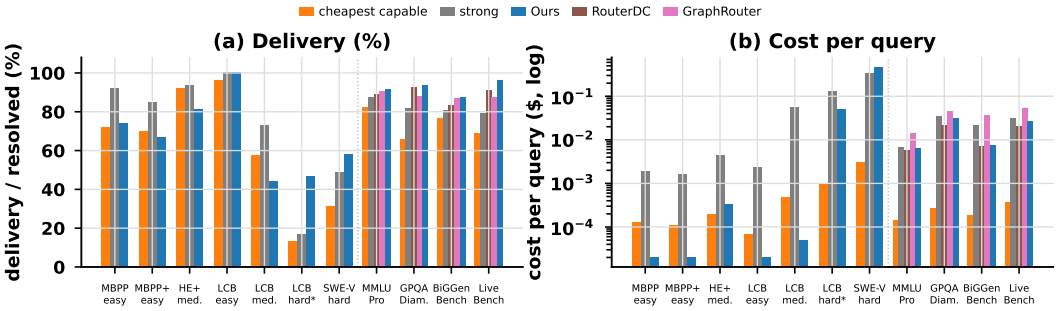


Figure 7: **Performance on public benchmarks with official grading.** We compare **(a)** delivery or resolve rate and **(b)** mean cost per query on a log scale.

while maintaining competitive delivery. These results show that routing quality is closely tied to the amount of evidence supporting each difficulty cell.

Additional Benchmarks. Transfer evaluation (Fig. 7) shows the router reaching the Pareto frontier on MBPP and LiveCodeBench-easy. On hard tasks, expanding calibration with LiveCodeBench-hard yields 46.7% accuracy on LiveCodeBench-hard at $0.4\times$ cost and a 57.8% resolution rate on SWE-bench Verified. Across MMLU-Pro, GPQA-Diamond, BiGGen-Bench, and LiveBench, the router matches the best single model (0.5–$1.3\times$ cost) while remaining competitive with RouterDC and GraphRouter without retraining. Overall, with continued updates, the router matches or surpasses strong single models while offering greater flexibility and lower cost.

5.4 What happens beyond the pool's reach?

Q. What happens when even the frontier cannot reliably solve a query?

A. When the estimated pool ceiling falls below the required threshold, the ceiling routes directly to the frontier, matching its delivery while reducing gateway errors from 62% to 0%.

We now test a distribution beyond the pool's reliable reach: the 108 deterministically reproducing BigCodeBench tasks, on which even the frontier delivers only 52%, so the honest ceiling (8) sits below τ_{ceil} and every hard query goes direct-to-frontier. On the in-house suite, where a proven strong model exists, ceiling-aware routing delivers 97.8% for \$0.22 in total, above the frontier (86.3%, \$0.46) and the strong-cheap model (92.6%, \$0.77); on the BigCodeBench ceiling subset it matches the frontier (47.5% vs. 45.0%) at the frontier's cost with zero escalations (Fig. 8a, Tab. 16).

5.5 Can a self-hosted model's internal state predict delivery?

Q. Can a self-hosted model's activations predict delivery before it generates?

A. Yes: a probe on the capability-matched layer lifts the local model from 69% to 83%.

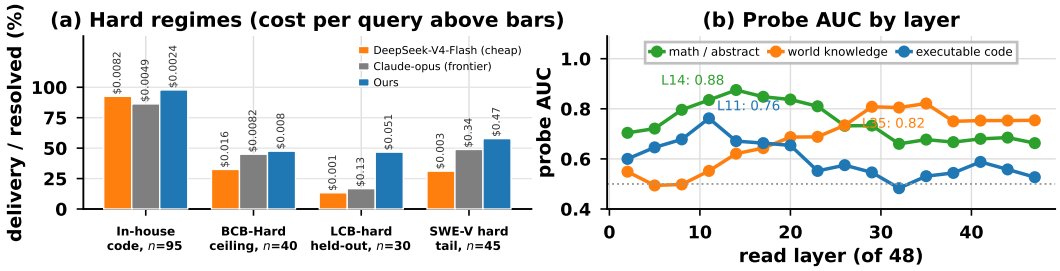


Figure 8: **Beyond the pool's reach, and inside a local model.** **(a)** Three anchors on four hard regimes : the in-house code suite (Tab. 3), the BigCodeBench-Hard ceiling subset (Tabs. 16, 14, 15). **(b)** Pre-generation probe AUC by read layer: a double dissociation (math mid-network, knowledge deep, code early-to-mid).

Table 1: **Verify escalation: Delivery (%) and cost vs. best single model** (10 **seeds; Tab. 7).**

Policy	MMLU-Pro deliv.	MMLU-Pro cost	GPQA-D deliv.	GPQA-D cost	BiGGen deliv.	BiGGen cost	LiveBench deliv.	LiveBench cost
Best single model	89.8	1.00	88.9	1.00	78.9	1.00	89.4	1.00
Cheapest, escalate to strong on failure	91.5	0.36	93.2	0.55	86.3	0.24	93.2	0.48
Ours, evidence-gated escalation	91.0	0.53	93.7	0.78	85.0	0.27	94.9	0.65

We place a local Qwen2.5-14B (Bai et al., 2023) in the pool and train the linear read-path probe described in Sec. 4.4 on the same pass/fail signal. As a black box, the local model delivers 69.1%, the paid frontier 87.3%. Gating on the internal read (Tab. 19) reaches 82.7% at a 38% escalation rate, +6.7 points over random escalation of the same fraction. Layer choice is decisive (Fig. 8b): math peaks mid-network (layer 14, AUC 0.88), knowledge deep (layer 35, 0.82), code early-to-mid (layer 11, 0.76), a double dissociation (Tenney et al., 2019; Geva et al., 2021; Meng et al., 2022).

5.6 Does a verifier turn parity into a gain?

Q. Where a routed call only matches the best model, does outcome evidence enable a gain through escalation?

A. Yes: evidence-gated escalation boosts delivery by +1.2–6.1 points across all suites at 0.27–0.78× cost.

For the four general suites, a single-routed call matches only the best single model, a limitation shared by baseline learned routers (Appendix H). To surpass this, we evaluate a deterministic verifier-enabled escalation mechanism (Sec. 4.2) using official verdicts. If the initial call fails, the router makes a second d/c-optimal call among the remaining evidence-supported models. As shown in Tab. 1, this selective escalation boosts delivery to 91.0%, 93.7%, 85.0%, and 94.9%, while escalating only 9–22% of queries. Although a cheapest-first cascade achieves similar delivery, it triggers significantly more escalations (19–37%). These results demonstrate that verifier-backed selective escalation efficiently breaks the single-model performance ceiling.

6 Conclusion

This paper introduced Model Metropolis, a live digital-twin framework for difficulty-aware LLM routing. Model Metropolis continuously updates models' performance across capabilities and difficulty levels and uses this evidence to balance quality and cost. Our experiments support this design. Difficulty-aware routing improves hard-query delivery from 73% to 91% while preserving substantial savings on easier queries. Ceiling-aware routing reduces gateway errors from 62% to 0% when the model pool cannot reliably handle a query. At the same time, pre-generation internal states improve self-hosted-model delivery from 69% to 83% at approximately one third of the frontier model's cost. We also find that transfer to workloads depends on whether calibration evidence covers the target task family. Overall, Model Metropolis ' goal is not to identify a permanently "best" model, but to determine which model is sufficiently reliable for the current query given the available evidence and cost. Future work will explore larger and more dynamic model pools, broader task families, and richer signals for updating the digital twin.

AI use statement

This paper uses AI only for polishing writing.

Reproducibility Statement

The datasets, routing code, and evaluation scripts are publicly available at https://anonymous.4open.science/r/ICLR--E6CE/. Additional experimental details and supplementary results are provided in Appendix A.

Ethics statement

This work involves no human subjects or personal data.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. Automix: Automatically mixing language models. *Advances in Neural Information Processing Systems*, 37:131000–131034, 2024.

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, 2012. URL https://api.semanticscholar.org/CorpusID:96146.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

Paul Almasan, Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Diego Perino, Diego López, Antonio Agustin Pastor Perales, Paul Harvey, Laurent Ciavaglia, Leon Wong, et al. Network digital twin: Context, enabling technologies, and opportunities. *IEEE Communications Magazine*, 60(11):22–27, 2022.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, 2023.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. *Advances in neural information processing systems*, 27, 2014.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Neural Information Processing Systems*, 2011. URL https://api.semanticscholar.org/CorpusID:6002655.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests. *arXiv preprint arXiv:2207.10397*, 2022.

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

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37:66305–66328, 2024.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 2024.

Xiangxiang Dai, Jin Li, Xutong Liu, Anqi Yu, and John C. S. Lui. Cost-effective online multi-llm selection with versatile reward models. *ArXiv*, abs/2405.16587, 2024.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks Lakshmanan, and Ahmed H Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, volume 2024, pp. 41348–41366, 2024.

David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A. Saurous, Jascha Narain Sohl-Dickstein, Kevin Murphy, and Charles Sutton. Language model cascades. *ArXiv*, abs/2207.10342, 2022.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. In *International Conference on Learning Representations*, volume 2025, pp. 26186–26203, 2025.

Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow. Digital twin: Enabling technologies, challenges and open research. *IEEE access*, 8:108952–108971, 2020.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *International conference on algorithmic learning theory*, pp. 174–188. Springer, 2011.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 5484–5495, 2021.

Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems: New findings and approaches*, pp. 85–113. Springer, 2016.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. *ArXiv*, abs/2404.10136, 2024.

Surya Narayanan Hari and Matt Thomson. Tryage: Real-time, intelligent routing of user prompts to large language models. *arXiv preprint arXiv:2308.11601*, 2023.

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

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, 2023.

Wittawat Jitkrittum, Neha Gupta, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, and Surinder Kumar. When does confidence-based cascade deferral suffice? *ArXiv*, abs/2307.02764, 2023. URL https://api.semanticscholar.org/CorpusID:259360997.

David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP journal of manufacturing science and technology*, 29: 36–52, 2020.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Latif U Khan, Walid Saad, Dusit Niyato, Zhu Han, and Choong Seon Hong. Digital-twin-enabled 6g: Vision, architectural trends, and future directions. *IEEE Communications Magazine*, 60(1): 74–80, 2022.

Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, et al. The biggen bench: A principled benchmark for fine-grained evaluation of language models with language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5877–5919, 2025.

Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. Digital twin in manufacturing: A categorical literature review and classification. *Ifac-PapersOnline*, 51(11): 1016–1022, 2018.

Tor Lattimore and Csaba Szepesvari. Bandit algorithms. 2020. URL https://api.semanticscholar.org/CorpusID:242172176.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *The Web Conference*, 2010. URL https://api.semanticscholar.org/CorpusID:207178795.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1964–1974, 2024.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Alireza Mohammadshahi, Arshad Shaikh, and Majid Yazdani. Routoo: Learning to route to large language models effectively. 2024. URL https://api.semanticscholar.org/CorpusID:267212103.

Huan X Nguyen, Ramona Trestian, Duc To, and Mallik Tatipamula. Digital twin for 5g and beyond. *IEEE Communications Magazine*, 59(2):10–15, 2021.

Quang H. Nguyen, Duy Cao Hoang, Juliette Decugis, Saurav Manchanda, N. Chawla, and Khoa D. Doan. Metallm: A high-performant and cost-efficient dynamic framework for wrapping llms. *ArXiv*, abs/2407.10834, 2024.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Vishnu Raj and Sheetal Kalyani. Taming non-stationary bandits: A bayesian approach. *arXiv preprint arXiv:1707.09727*, 2017.

Adil Rasheed, Omer San, and Trond Kvamsdal. Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE access*, 8:21980–22012, 2020.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, and Ian Osband. A tutorial on thompson sampling. *Found. Trends Mach. Learn.*, 11:1–96, 2017. URL https://api.semanticscholar.org/CorpusID:3929917.

Marija Šakota, Maxime Peyrard, and Robert 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*, pp. 606–615, 2024.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

KV Aditya Srivatsa, Kaushal Kumar Maurya, and Ekaterina Kochmar. Harnessing the power of multiple minds: Lessons learned from llm routing. *ArXiv*, abs/2405.00467, 2024. URL https://api.semanticscholar.org/CorpusID:269484704.

Dimitris Stripelis, Zhaozhuo Xu, Zijian Hu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, and Chaoyang He. Tensoropera router: A multi-model router for efficient llm inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 452–462, 2024.

Fei Tao, He Zhang, Ang Liu, and Andrew YC Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on industrial informatics*, 15(4):2405–2415, 2018.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4593–4601, 2019.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. Fusing models with complementary expertise. In *International Conference on Learning Representations*, volume 2024, pp. 45284–45306, 2024a.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, volume 2025, pp. 33944–33963, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024b.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, et al. Livebench: A challenging, contamination-limited llm benchmark. In *International Conference on Learning Representations*, volume 2025, pp. 91595–91631, 2025.

Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Yiwen Wu, Ke Zhang, and Yan Zhang. Digital twin networks: A survey. *IEEE Internet of Things Journal*, 8(18):13789–13804, 2021.

Murong Yue, Jie Zhao, Min Zhang, Li Du, and Ziyu Yao. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *ArXiv*, abs/2310.03094, 2023. URL https://api.semanticscholar.org/CorpusID:263671564.

Zesen Zhao, Shuowei Jin, and Z. Morley Mao. Eagle: Efficient training-free router for multi-llm inference. *ArXiv*, abs/2409.15518, 2024. URL https://api.semanticscholar.org/CorpusID:272832307.

A Model Pool and Provider Buildings

Tab. 2 lists the 30 models of our gateway, grouped by the 9 provider buildings of the digital twin. Building height (floor count) varies from a single model (DeepSeek, Moonshot) to seven (Anthropic). The twin instantiates one room per (model, capability) and, within each room, one Beta boundary per difficulty level, so the total state is $|\mathcal{M}| \times |\mathcal{C}| \times |\mathcal{D}|$ pairs of decayed counts.

Table 2: The 30-model text pool grouped by provider building.

Provider (building)	# floors	Models (floors)
Anthropic	7	claude-fable-5, claude-haiku-4-5, claude-opus-4-{6,7,8}, claude-sonnet-{4-6,5}
OpenAI	6	gpt-5.4, gpt-5.4-mini, gpt-5.5, gpt-5.6-{luna,sol,terra}
Google	4	gemini-3.5-flash, gemini-3-flash-preview, gemini-2.5-pro, gemini-3.1-pro-preview
ByteDance	3	Doubao1.5-pro-32k, Doubao1.5-lite-32k, doubao-seed-2.1-pro
MiniMax	3	minimax-m3, minimax-m2.7, Minimax-M2.5
Qwen	3	qwen3.7-max, qwen3.7-plus, qwen3.6-plus
Zhipu	2	glm-5.2, glm-5-turbo
DeepSeek	1	DeepSeek-V4-Flash
Moonshot	1	kimi-k2.7-code

B Executable Task Suite and Anti-Contamination Construction

Our supervision is an executable pass/fail signal, so a task is a triple (prompt, entry point, hidden tests). Tab. 3 lists the six facets and their sizes. The public benchmarks (HumanEval+, MBPP) provide breadth; the four hand-built facets provide a discriminative hard tail.

To reduce direct memorization effects on the hard tail, we construct semantic variants of well-known problems whose hidden tests differ from the original task. Examples include decoding with a forbidden token pair, merging intervals that only touch (gap ≤ 1), searching a rotated array with duplicates, longest palindromic subsequence in place of substring, and a budgeted variant of a classic DP problem. Grading extracts the first fenced code block and executes it against hidden tests in an isolated subprocess with a timeout. The resulting executable verdict provides the binary signal (1).

Table 3: The executable task suite. The four hand-built facets supply a discriminative hard tail; the public benchmarks supply breadth.

Facet	# tasks	construction
Anti-contamination variants	30	hand-written semantic mutations
Domain (Web / backend / ML)	30	hand-written realistic engineering tasks
Math & abstract reasoning	20	hand-written reasoning-to-code tasks
Competitive / olympiad	16	hand-written contest-style tasks
HumanEval+	164	public benchmark
MBPP	257	public benchmark
Total	517	

C Difficulty and Capability Estimation

Capability $c(q)$ is inferred by an ordered intent classifier that scans for facet cues (e.g., translate, summarize, explain, estimate, design, audit, debug, implement) and returns the first match, defaulting to code generation when ambiguous; ordering places more specific intents first so terse or lazy queries route correctly. Difficulty $d(q)$ is bucketed from surface and structural cues into $\{$easy, medium, hard, expert$\}$. For instance, a request to "write hello world" is easy, "implement an LRU cache" is medium, "solve weighted interval scheduling by DP" is hard, and "implement a Raft-like consensus protocol handling leader election and log replication" is expert. These estimators are deliberately lightweight and both are pluggable. A learned per-request difficulty predictor can replace the heuristic without changing the remainder of the routing pipeline.

D BANDIT HYPERPARAMETERS AND ROUTING ALGORITHM

Tab. 4 lists the hyperparameters of the online executable-routing policy. The quality bar τ_d and evidence floor ν_d rise with difficulty, while the exploration rate ϵ_d is difficulty-dependent and controls targeted evidence collection for under-observed cells. The decay γ gives an effective memory of roughly $1/(1-\gamma) \approx 25$ recent observations per cell. The deterministic public-benchmark replay of Appendix H turns off Thompson sampling and targeted exploration and uses its own mean-based delivery band.

Table 4: Online gateway-router hyperparameters. Quality thresholds and evidence floors depend on difficulty; exploration rates match the executable-routing configuration.

	easy	medium	hard	expert
Quality bar τ_d	0.55	0.70	0.85	0.90
Evidence floor ν_d	0	0	3	3
Exploration rate ϵ_d	0.00	0.05	0.20	0.35
Prior exponent η_d	0.5	1.0	1.6	2.4

Global	value
Decay γ	0.96
Prior strength κ	3
Thompson prior a_0, b_0	$\kappa s_m^{\eta_d},\ \kappa(1-s_m^{\eta_d})$
Cost weight λ	15
Warm-cell tie band	0.05
Escalation eligibility factor	$0.85\tau_d$
Ceiling evidence n_{ceil}	15
Ceiling bar τ_{ceil}	0.65
Effective memory	≈ 25 observations

Routing decision for a query (c, d)*:*

1. Candidates $\leftarrow$ online floors serving c, excluding any already-tried models on escalation.
2. For each candidate m, draw $p_m \sim \text{Beta}(\alpha_d^{(m)} + \alpha_d^0, \beta_d^{(m)} + \beta_d^0)$, and read its posterior mean $\bar{p}_m$, evidence n_m, and cost.
3. *Targeted exploration:* with probability ϵ_d, if under-sampled cells remain, route to the least-observed cell; ties favor strong models that cost minimization would otherwise starve.
4. *Evidence filter:* exclude candidates with $n_m < \nu_d$ from normal exploitation.
5. *Value ranking:* on hard/expert queries, or for cells without calibrated per-query cost, rank candidates by $p_m - \lambda\,\text{cost}(m)$. On easy/medium queries with calibrated costs, rank by $\bar{p}_m - \lambda\,\text{cost}(m)$ and select the cheapest candidate within a 0.05 tie band.
6. *Quality gate:* if the selected model has $\bar{p}_m < \tau_d$, replace it with the reliable frontier when available; otherwise use the evidence-supported model with the largest posterior mean.
7. *Escalation after observed failure:* for hard/expert queries, rank untried models by $(p_m, -\text{cost}(m))$; for easy/medium queries, restrict to models with $\bar{p}_m \geq 0.85\tau_d$ and select the largest $p_m/\text{cost}(m)$. Record the resulting skybridge handoff.

E COST CALCULATION

For the original 115-task executable-routing suite, Tab. 18 reports relative cost as the mean output-token unit price of the model selected for each query, normalized to All-strong:

$$\bar{c}_\pi = \frac{1}{N}\sum_{i=1}^{N} \text{price}(m_\pi(q_i))\,. \tag{9}$$

This price-based relative-cost measure is used only for the original executable-routing suite. In the later public-benchmark experiments, where completion-token counts are available for each query, cost is computed as completion tokens multiplied by the corresponding unit price. The two reporting conventions are kept separate and are not compared as absolute dollar costs.

F PER-CAPABILITY-FACET DELIVERY

Tab. 5 lists executable delivery per capability facet across the pool and Fig. 9 visualizes the same measurements. The per-facet view exposes variation across capabilities and supplies the evidence used by the corresponding twin rooms.

Table 5: Executable delivery per capability facet across the 30-model pool, with the highest observed model rate for each facet; † denotes a small-sample estimate.

Capability facet	pool delivery	strongest model (rate)
Anti-contamination variants	93%	Doubao-pro (0.97)
Abstract-concept impl.	90%	Doubao-pro (0.94)
HumanEval+	86%	Doubao-pro (0.87)
Math reasoning	78%	Doubao-pro (0.89)
Competitive / olympiad	77%	minimax-m3 (0.85)
MBPP	72%	claude-fable-5 (0.95)
AI / ML	65%	minimax-m3 (0.93)
Backend	58%	minimax-m3 (0.88)
Web / UI	**30%**	gpt-5.6-luna (1.0†)

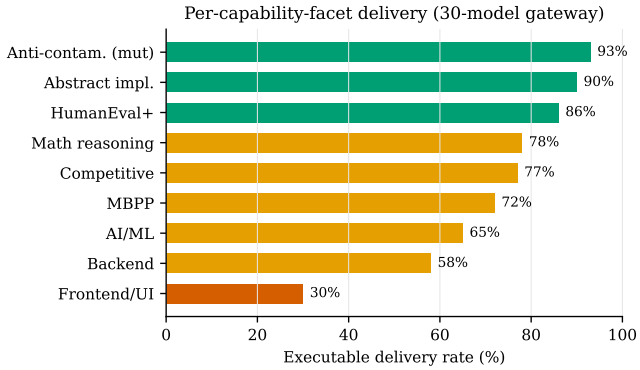


Figure 9: Executable delivery rate per capability facet across the 30-model pool (values in Tab. 5, Appendix F).

G ROUTING DECISIONS IN PRACTICE: INSTANCES, DISTRIBUTION, AND PIPELINES

Fig.s 10, 11, and 12 provide concrete views of routing decisions, the observed routing distribution, and the state used along representative routing pipelines.

H ADDITIONAL EVALUATION SUITES

We additionally evaluate MMLU-Pro (Wang et al., 2024b), GPQA-Diamond (Rein et al., 2023), BiGGen-Bench (Kim et al., 2025), and LiveBench (White et al., 2025). These experiments are reported as held-out routing replays rather than as part of the online executable-routing study.

Protocol. Each suite is split into calibration and held-out subsets. Every pool model answers each sampled item once at temperature 0, and grading follows the corresponding benchmark procedure. For these public-suite experiments, cost is completion tokens multiplied by the model's unit price. The twin is built from the calibration split only. Because the query profiler in Sec. 3 was designed for code queries, difficulty tiers for these suites are assigned from calibration-neighbor pass rates.

The replay uses a deterministic readout of the same twin with Thompson sampling and targeted exploration disabled:

$$\mathcal{F} = \{m : n_m \geq \nu_d \ \wedge \ \bar{p}_m \geq \tau_d\}, \qquad R_{\text{det}}(q) = \arg \min_{m \in \mathcal{F}:\, \bar{p}_m \geq \max_{\mathcal{F}} \bar{p} - \delta} \text{cost}(m), \qquad \delta = 0.03. \tag{10}$$

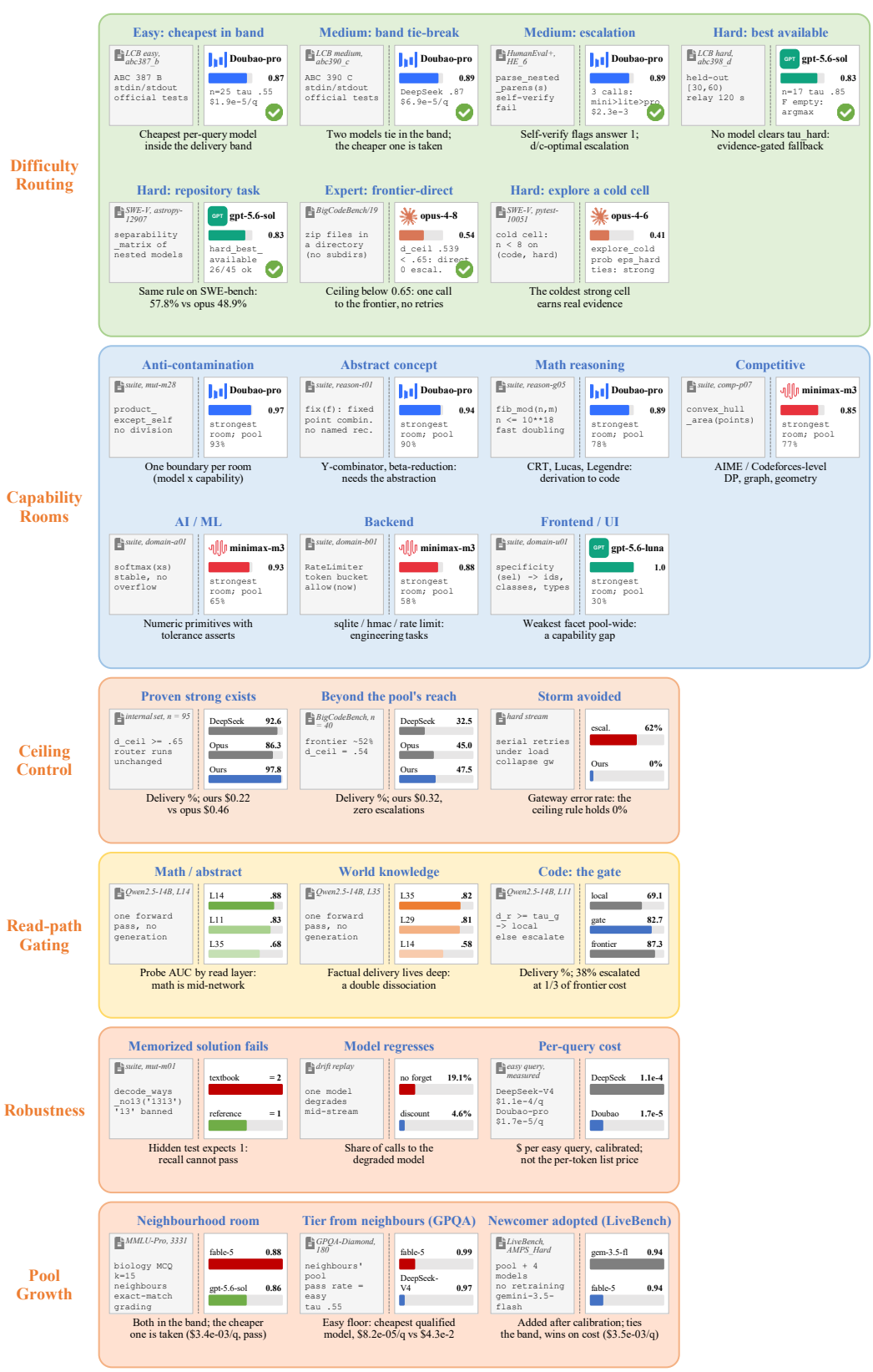
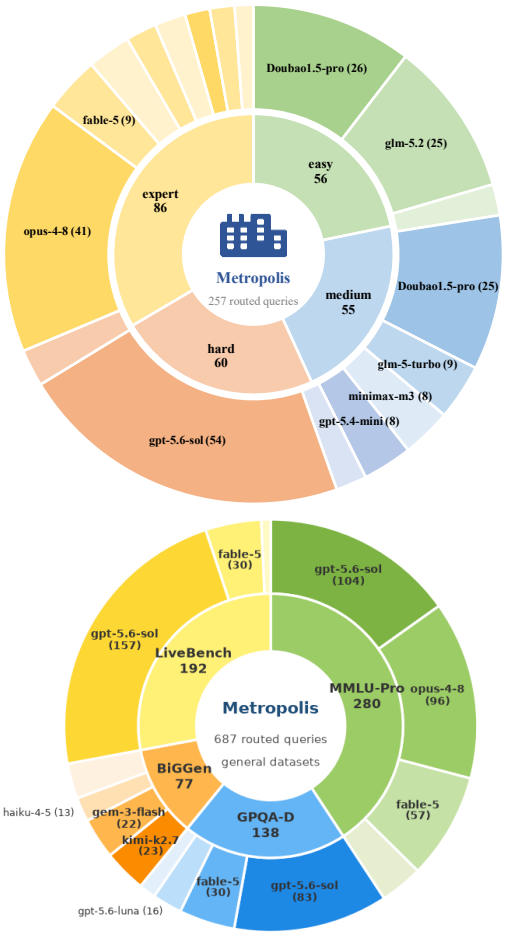


Figure 10: **Routing decision types with one real routed instance per card.** Capability rooms: the strongest room per facet (Tab. 5). Ceiling control, read-path gating, and robustness cards carry the measured values of Secs. 4.3, 5.5, and B.



Routing records behind the chart (official / executable grading)

Tier	*Routed model (queries)*
easy (56)	Doubao1.5-pro 26, glm-5.2 25, minimax-m3 5
medium (55)	Doubao1.5-pro 25, glm-5-turbo 9, minimax-m3 8, gpt-5.4-mini 8, other 5
hard (60)	gpt-5.6-sol 54, opus-4-6 6
expert (86)	opus-4-8 41, fable-5 9, glm-5-turbo 7, sonnet-5 5, haiku-4-5 5, gpt-5.6-sol 4, opus-4-6 4, other 3

General datasets: 687 held-out queries (neighbourhood-room router)

Suite	*Routed model (queries)*
MMLU-Pro (280)	gpt-5.6-sol 104, opus-4-8 96, fable-5 57, gpt-5.6-luna 22, other 1
GPQA-D (138)	gpt-5.6-sol 83, fable-5 30, gpt-5.6-luna 16, DeepSeek-V4 4, other 5
BiGGen (77)	kimi-k2.7-code 23, gem-3-flash 22, haiku-4-5 13, minimax-m3 13, other 6
LiveBench (192)	gpt-5.6-sol 157, fable-5 30, opus-4-8 5

Figure 11: **Observed routing distribution.** Inner ring: difficulty tier; outer ring: routed model, over the 257 routed queries with official or executable grading in the routing logs.

Table 6: Held-out delivery and realized mean query cost on four additional suites. The best-single row is computed directly from the held-out per-model records; the two Model Metropolis rows use the same calibration split.

Policy	MMLU-Pro		GPQA-Diamond		BiGGen-Bench		LiveBench	
	delivery	$/query	delivery	$/query	delivery	$/query	delivery	$/query
Best single model	90.7	1.39e-2	92.8	2.21e-2	85.7	4.21e-2	91.7	2.17e-2
Ours, tier rooms	87.9	6.50e-3	90.6	2.83e-2	80.5	6.36e-3	91.7	2.17e-2
Ours, neighbourhood rooms	90.7	6.89e-3	91.3	2.85e-2	77.9	4.74e-3	92.7	2.51e-2

The four suites expose different cost–delivery trade-offs. We therefore report the measurements without assigning an overall winner across datasets.

Table 7: Verifier replay on the four held-out suites. The recorded benchmark verdict is used as an oracle verifier after the first call; these suites do not provide this signal during normal inference. Cost is relative to the corresponding best single model.

Policy	MMLU-Pro			GPQA-Diamond			BiGGen-Bench			LiveBench		
	deliv.	cost	esc.	deliv.	cost	esc.	deliv.	cost	esc.	deliv.	cost	esc.
Best single model	89.8	1.00	0%	88.9	1.00	0%	78.9	1.00	0%	89.4	1.00	0%
Cheapest + escalation to strong model	91.5	0.36	19%	93.2	0.55	37%	86.3	0.24	26%	93.2	0.48	31%
Ours + evidence-gated escalation	91.0	0.53	13%	93.7	0.78	10%	85.0	0.27	22%	94.9	0.65	9%

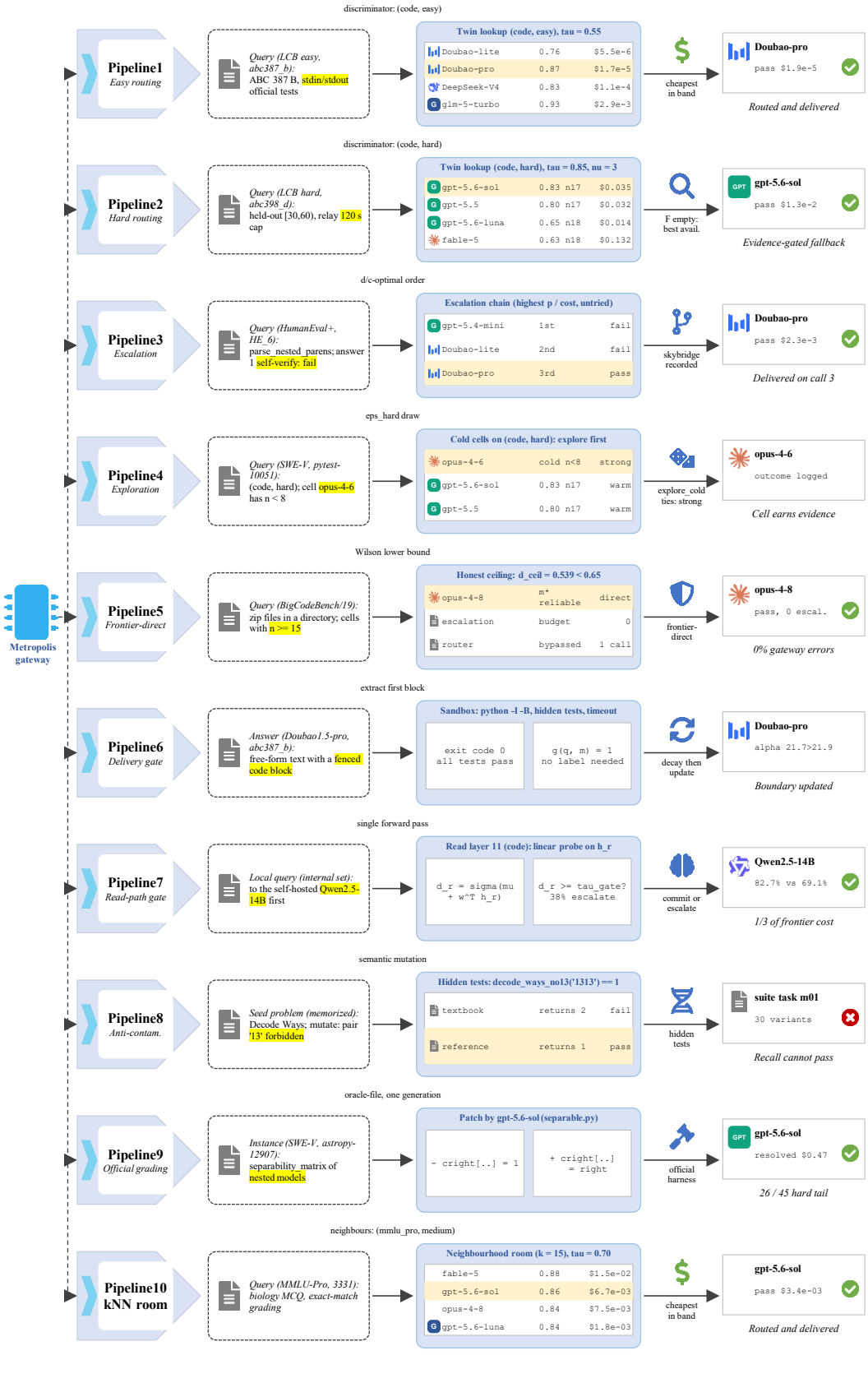


Figure 12: **One pipeline per decision type, with the real intermediate state at every stage.** Each row follows one query from Fig. 10: the dashed box is the input, the blue panel the twin's state.

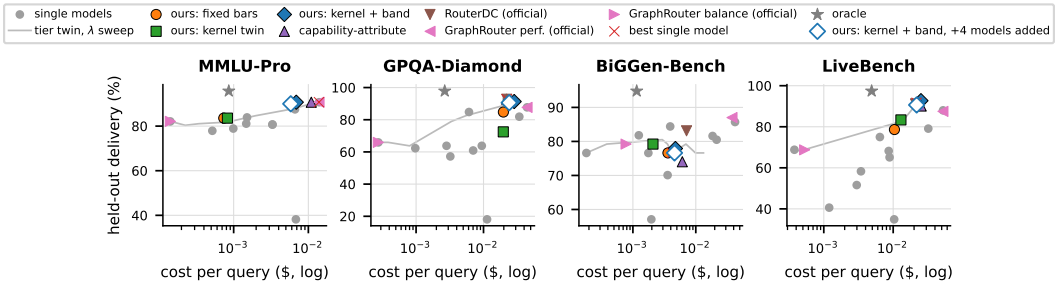


Figure 13: **Cost–delivery landscape on four held-out question suites.** Grey points show pool models, the grey curve shows a deterministic twin readout under a cost-penalty sweep, and markers show selected routing operating points.

I Adaptation to a Growing Model Pool

We evaluate whether the online router can begin collecting evidence for models introduced after routing has started. In the executable-code stream, four models are introduced halfway through with no calibration data, so they can be learned only through subsequent exploration and observed outcomes. Across 10 seeds, second-half delivery is 92.5% with the original pool and 92.6% after the four models are introduced; 13% of first calls in the second half are assigned to the new models. This experiment measures online incorporation of new candidates without retraining the routing policy.

Table 8: Executable-stream pool-growth test over 10 seeds. The four added models enter without calibration data.

Setting	second-half delivery	first calls to added models
Original pool	92.5%	0%
Pool + 4 models	92.6%	13%

J Pairwise Boundary Agreement

Fig. 14 reports pairwise agreement of executable verdicts on shared calibration tasks. Agreement is the fraction of shared tasks for which two models return the same pass/fail outcome. The visualization is descriptive and summarizes similarity among observed model outcomes.

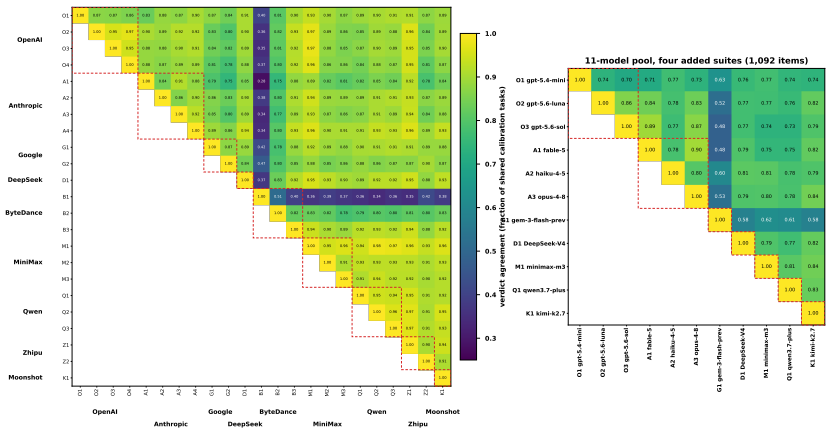


Figure 14: **Pairwise verdict agreement between pool models** on shared calibration tasks. The right panel shows the corresponding agreement matrix for the additional question-suite pool using the same color scale.

K Complete Related Work

Reward modeling evaluates generated responses, whereas routing selects a model before generation and therefore requires an estimate of query difficulty and model suitability. Model Metropolis focuses on routing and represents suitability with difficulty-conditioned delivery posteriors updated from observed outcomes.

Learned query routers. RouteLLM (Ong et al., 2025), Hybrid-LLM (Ding et al., 2024), RouterDC (Chen et al., 2024), GraphRouter (Feng et al., 2025), and related meta-model approaches learn or profile query–model compatibility from preference, benchmark, or embedding features (Šakota et al., 2024; Shnitzer et al., 2023; Hari & Thomson, 2023; Mohammadshahi et al., 2024; Srivatsa et al., 2024; Nguyen et al., 2024; Zhao et al., 2024; Stripelis et al., 2024). RouterBench (Hu et al., 2024) provides a standardized offline benchmark for this family. Many such methods are trained or profiled for a particular candidate pool. Our implementation instead maintains per-model posteriors that can be updated as new outcomes arrive.

Cascades, ensembles, and reward-guided selection. FrugalGPT (Chen et al., 2023), AutoMix (Aggarwal et al., 2024), and language-model cascades (Dohan et al., 2022; Jitkrittum et al., 2023; Gupta et al., 2024; Yue et al., 2023) use additional model calls when escalation is triggered. Ensemble approaches such as LLM-Blender (Jiang et al., 2023), Mixture-of-Agents (Wang et al., 2025), and expertise fusion (Wang et al., 2024a) combine outputs from multiple models. Zooter (Lu et al., 2024) uses reward-derived routing labels. Model Metropolis instead uses observed delivery outcomes to update its routing state and escalates after an observed failure when a verifier is available.

Bandits and online selection. Online model selection with bandits has been studied for LLM routing (Dai et al., 2024). Our online gateway uses Thompson sampling (Thompson, 1933; Chapelle & Li, 2011; Agrawal & Goyal, 2012; Russo et al., 2017), contextual-bandit ideas (Li et al., 2010; Lattimore & Szepesvari, 2020), and discounted estimators for non-stationary observations (Garivier & Moulines, 2011; Besbes et al., 2014; Raj & Kalyani, 2017). The quality gate is based on posterior means and accumulated evidence, while Thompson samples are used for ranking under uncertainty and the exploration budget controls targeted evidence collection.

Digital twins. Digital twins are virtual counterparts whose state is updated from observations of the represented system (Grieves & Vickers, 2016; Tao et al., 2018; Kritzinger et al., 2018; Jones et al., 2020; Fuller et al., 2020; Rasheed et al., 2020). Network digital twins similarly maintain topology, traffic, or resource state for analysis and control (Nguyen et al., 2021; Wu et al., 2021; Almasan et al., 2022; Khan et al., 2022). Here the represented state is a set of difficulty-stratified delivery posteriors for the model pool.

Reading internal representations. Linear probes of intermediate activations have been used to analyze model representations (Alain & Bengio, 2016; Belinkov, 2022), including signals related to truthfulness, factual processing, and internal knowledge (Azaria & Mitchell, 2023; Burns et al., 2022; Kadavath et al., 2022; Marks & Tegmark, 2023; Tenney et al., 2019; Geva et al., 2021; Meng et al., 2022). Our read-path experiment uses activations as an additional pre-generation delivery signal for a self-hosted model.

L Baselines and Protocol Details

Baselines. We compare eight routing policies on the same executable task stream and delivery gate (1). Random chooses an online model uniformly; All-cheap always selects the cheapest model; Cost-min uses a single fixed bar with sample-based feasibility; Calibrated threshold uses a fixed strong/weak split by difficulty; Cascade starts from the cheapest model and makes one additional strong-model call after an observed failure; Difficulty-agnostic bandit removes the difficulty axis from our learning router; All-strong always selects the designated frontier model; and Ours is the difficulty-aware, evidence-gated bandit. Stateless policies are evaluated in one pass, while the learning policies are reported after five rounds. The hard bucket is additionally repeated three times (Tab. 10). Relative cost is normalized to All-strong on the same executable suite.

M Additional Definitions

Strong-tier share. Per difficulty, the fraction of traffic routed to the strong tier $\mathcal{S}$ (the highest-delivery models):

$$\mathrm{Share}_d = \frac{1}{|Q_d|} \sum_{q \in Q_d} \mathbb{1}\big[R(q) \in \mathcal{S}\big]. \tag{11}$$

Table 11: Primary routed model by difficulty after five rounds in the executable-routing experiment.

Difficulty	primary model routed to	tier	latency
easy	Doubao-lite	cheap	1.8s
medium	DeepSeek-V4-Flash	cheap	2.9s
hard	`gpt-5.6-sol`	strong	8.1s
expert	`gpt-5.6-sol` / Claude	strong	12.4s

This quantity summarizes how strongly routing shifts toward the designated strong tier as difficulty changes.

Wilson lower bound. The honest ceiling of Sec. 4.3 uses, for a model with observed delivery $\hat{p}_m$ over n_m real observations and confidence width z,

$$\tilde{p}_m = \mathrm{WLB}(\hat{p}_m, n_m, z) = \frac{\hat{p}_m + \frac{z^2}{2n_m} - z\sqrt{\frac{\hat{p}_m(1-\hat{p}_m)}{n_m} + \frac{z^2}{4n_m^2}}}{1 + \frac{z^2}{n_m}}. \tag{12}$$

N Additional Results

Eight policies by difficulty. Tab. 9 lists the executable delivery of all eight policies per difficulty tier behind Fig. 1, and Tab. 10 the hard bucket over three independent repeats (the learning policies re-learn from scratch each repeat).

Table 9: **Delivery by difficulty.** Executable delivery for all eight policies on the 30-model gateway (n=27/26/51/11). Relative cost is the mean selected-model output-token unit price normalized to All-strong on the same suite; hard values for repeated policies are three-run means (Tab. 10).

Policy	easy	medium	hard	expert	rel. cost
Label-free, online (no oracle)					
Random	59%	62%	54%	73%	0.33×
All-cheap	96%	85%	68%	82%	0.003×
Cost-min (naive)	97%	86%	73%	69%	0.003×
Difficulty-agnostic bandit	96%	87%	70%	73%	0.026×
Difficulty-aware bandit (ours)	96%	92%	**91%**	85%	0.44×
Reference policies using additional information					
Calibrated threshold (Ong et al., 2025)	96%	88%	96%	100%	0.54×
Cascade (Chen et al., 2023) (2× calls)	100%	96%	98%	100%	0.22×
All-strong	96%	100%	99%	100%	1.00×

Table 10: Hard-bucket delivery (n=51) over three independent repeats; the learning policies re-learn from scratch in each repeat. Reference policies are shown for context.

Policy	hard delivery (mean ± s.d., 3 repeats)
Random	53.6 ± 2.5
All-cheap	68.0 ± 3.7
Difficulty-aware bandit (ours)	$\mathbf{91.2 \pm 1.0}$
Cascade (verifier reference)	98.7 ± 0.9
All-strong (frontier reference)	99.4 ± 0.9

Routing distribution and ablation. Tab. 11 shows where the bandit routes each difficulty after five rounds, and Tab. 12 the ablation of the three fixes discussed in Sec. 5.1.

Held-out MBPP, MBPP+, and HumanEval+. Tab. 13 gives the real-gateway results on the held-out public code splits summarized in Sec. 5.3.

Table 12: Qualitative routing behavior as the three gating components are added.

Configuration	observed hard-query routing behavior
Fixed bar 0.6, sample gate	routes primarily to cheap models
+ difficulty-dependent bar τ_d	cold cells remain competitive through prior uncertainty
+ mean-based feasibility gate	cold cells can still enter without an evidence floor
+ evidence floor ν_d	routing concentrates on evidence-supported strong models

Table 13: Real-gateway cost–delivery on held-out public code benchmarks with disjoint calibration and evaluation splits. The table reports the observed operating points for the selected reference models and the router; no single trend is assumed to hold across all three benchmarks.

Benchmark	Policy	delivery	cost/query
MBPP (easy), n=100	cheapest capable (DeepSeek-V4-Flash)	72.0%	1.3×10^{-4}
	strong (Claude-opus)	92.0%	1.9×10^{-3}
	Ours (routes Doubao-pro)	**74.0%**	$\mathbf{2\times10^{-5}}$
MBPP+ (easy), n=120	cheapest capable (DeepSeek-V4-Flash)	70.0%	1.1×10^{-4}
	strong (Claude-opus)	85.0%	1.6×10^{-3}
	Ours (routes Doubao-pro)	66.7%	$\mathbf{2\times10^{-5}}$
HumanEval+ (med.), n=64	cheapest capable (DeepSeek-V4-Flash)	92.2%	2.0×10^{-4}
	strong (Claude-opus)	93.8%	4.5×10^{-3}
	Ours (routes Doubao-pro)	81.2%	3.4×10^{-4}

LiveCodeBench, SWE-bench Verified, and ceiling regimes. Tabs. 14, 15, and 16 list the values plotted in Figs. 7 and 8(a).

Table 14: **LiveCodeBench** (2025 problems, official stdin/stdout grading), single call. First three blocks: twin calibrated *without* any LiveCodeBench evidence. †: held-out hard problems $[30, 60)$ before and after adding LCB-hard $[0, 30)$ to the hard-tier calibration; 8 of 30 queries count as failures from the relay's 120 s cap, 4 are exploration calls.

Tier	Policy	delivery	cost/query
easy, n=26	cheapest capable (DeepSeek-V4-Flash)	96.2%	7×10^{-5}
	strong (Claude-opus)	100.0%	2.4×10^{-3}
	Ours (routes Doubao-pro)	**100.0%**	$\mathbf{2\times10^{-5}}$
medium, n=26	cheapest capable (DeepSeek-V4-Flash)	57.7%	4.8×10^{-4}
	strong (Claude-opus)	73.1%	5.5×10^{-2}
	Ours (routes Doubao-pro)	44.0%	5×10^{-5}
hard, n=60	cheapest capable (DeepSeek-V4-Flash)	22.4%	9.2×10^{-4}
	strong (Claude-opus)	35.0%	1.1×10^{-1}
	Ours (routes gpt-5.4-mini, others)	20.0%	3.5×10^{-2}
hard†, n=30	cheapest capable (DeepSeek-V4-Flash)	13.3%	1.0×10^{-3}
	strong (Claude-opus)	16.7%	1.3×10^{-1}
	Ours, hard tier from BCB-Hard only	10.0%	5.7×10^{-2}
	Ours, hard tier from BCB-Hard + LCB-hard	**46.7%**	$\mathbf{5.1\times10^{-2}}$

Table 15: **SWE-bench Verified** (official harness), oracle-file setting, one generation per instance. "All 77" is the stratified sample with oracle context; "shared 45" the hard/expert instances all policies completed (the router's medium instances were blocked by the Doubao endpoint's rate limit). Cost is mean gateway cost per generated instance, prompt and completion tokens included.

Policy	all 77	shared 45 (hard tail)	cost/instance
cheapest capable (DeepSeek-V4-Flash)	39.0%	31.1%	$0.003
strong (Claude-opus)	62.3%	48.9%	$0.345
Ours (hard tier: BCB-Hard only)	n/a	24.4%	$0.090
Ours (hard tier: BCB-Hard + LCB-hard)	n/a	**57.8%**	$0.470

Table 16: **Ceiling-aware routing in two evaluated regimes.** The table reports delivery, total cost, escalation count, and the gateway failure rate in the accompanying stress replay.

Regime	Policy	delivery	cost (total)	escal.
Internal set (n=95)				
	DeepSeek (strong-cheap)	92.6%	$0.77	n/a
	Frontier (Opus)	86.3%	$0.46	n/a
	Ceiling-aware (ours)	**97.8%**	**$0.22**	15
BigCodeBench subset (n=40)				
	DeepSeek (strong-cheap)	32.5%	$0.62	n/a
	Frontier (Opus)	45.0%	$0.33	n/a
	Ceiling-aware (ours)	**47.5%**	$0.32	**0**
Gateway failure rate in stress replay				
	Escalation-driven		62%	
	Ceiling-aware (ours)		**0%**	

Constraint sensitivity. Tab. 17 reports discrimination accuracy for three service-level settings.

Table 17: Sensitivity of discrimination accuracy to the service-level fail-rate constraint.

SLA fail-rate constraint	0.05	0.15	0.25
Discrimination accuracy	12%	88%	25%

Cost basis of all eight policies. Tab. 18 reports the same price-based relative-cost measure defined in Appendix E, normalized to All-strong on the original 115-task executable-routing suite.

Table 18: Cost–delivery of all eight policies on the same 115-task executable-routing suite. Relative cost is normalized to All-strong using the selected-model output-token unit-price measure of Appendix E.

Policy	rel. cost	hard	note
Label-free, online			
Random	0.33×	54%	random routing
All-cheap	0.003×	68%	always cheapest
Cost-min (naive)	0.003×	73%	fixed low-bar selection
Difficulty-agnostic	0.026×	70%	difficulty-blind online policy
Ours	0.44×	**91%**	difficulty-aware single-call routing
Reference policies using additional information			
Cascade	0.22×	98%	uses an inference-time verifier
Calibrated threshold	0.54×	96%	uses a fixed strong/weak split
All-strong	1.00×	99%	designated frontier model

Read-path gate. Tab. 19 reports the operational read-path experiment using a fixed activation read-out.

Table 19: Read-path gating at the reported 38% escalation operating point. The result is reported as the measured operating point of this experiment.

Policy (local)	delivery	rel. cost	escal.
Local model only (black box)	69.1%	n/a	n/a
Random escalation (same budget)	76.0%	$\frac{1}{3}\times$	38%
Read-path gate (ours)	**82.7%**	$\frac{1}{3}\times$	38%
Full gateway (frontier)	87.3%	$1.00\times$	100%

Routing overhead. The online gateway decision requires one Beta draw per candidate followed by evidence filtering and a linear value scan over at most 30 candidate rooms. Hard/expert and cost-cold cells use the sampled value $p_m - \lambda\,\mathrm{cost}(m)$, while warm easy/medium cells use the corresponding posterior-mean value. Targeted exploration and fallback require only additional linear scans. Tab. 20 reports the per-decision latency and state footprint. Relative to multi-second model generations, the measured routing computation is small and runs in-process on CPU; the twin update uses the same observed delivery signal as the router state.

Table 20: Measured routing overhead for the online gateway decision and per-outcome twin update.

Component	latency	scaling	infrastructure
Gateway routing decision (Thompson + value ranking)	<1 ms	$O(\lvert\mathcal{M}\rvert)$	CPU, in-process
Twin update per outcome	<0.1 ms	$O(1)$	CPU, in-process
State footprint	$\lvert\mathcal{M}\rvert\times\lvert\mathcal{C}\rvert\times\lvert\mathcal{D}\rvert$ Beta counts		memory