The Agent Loop Is a Routing Decision: Measuring the Switching Tax in Multi-Turn LLM Routing

Anonymous authors
Paper under double-blind review

Abstract

Multi-turn LLM routers choose a model at every step of an agent loop, but they price each step as if billing had no memory. Providers bill statefully: a model reads at a discount only the prefix it has cached. We measure this on live APIs with 211 SWE-bench agent trajectories and 3,473 billed calls, and replay their switch schedules on four more deployments. A switch to a model not yet used in the session re-bills the whole prefix at the write rate (315 of 322 such switches read nothing), while a return re-reads most of the prefix that model last cached. The switching tax is therefore $(p^{\mathrm{cw}} - p^{\mathrm{cr}})(c_{t-1} - q\, c_{\mathrm{seen}})$, where q is the effective cache retention of the deployment. Across six screened deployments q runs from 0.81 to 0.99, and one model pair has effective retention 0.97 on its vendor's API but 0.81 and 0.83 on two re-hosts. With each deployment's cache block and minimum cacheable prefix, 95% of predicted trajectory bills land within $\pm 10\%$ of the billed cost, against 11% under full rebuild. From the tax we derive a closed-form break-even horizon and an output-aware diagnostic $\Psi(q)$ that chooses between committing to one model and following demand, and hands cases inside a band to a tax-greedy rule. In live replays the prescribed policy is the cheaper one at every horizon outside the band, and inside it tax-greedy stays within $1.05\times$ of the better fixed policy. The price sheet alone picks the costlier policy on one of these pools at 9 of 10 horizons. We prove a K-independent downgrade threshold for ordered tiers and worst-case linear-in-T regret for per-step learners when $q < 1$. Re-billing 1,785 production sessions with the router's own decisions, the switching tax is 20% of the median session's bill.

1 Introduction

An agent harness runs a loop of T steps: propose an action, execute it, observe the result, and a multi-turn router picks the model for each step. Published results disagree on whether this choice pays (Zhang et al., 2026c; 2025a; Yang et al., 2026). The answer depends on quantities the router does not see: the pool's price sheet, the step budget, and the cache.

The switching tax. Providers cache prompt prefixes and bill cached tokens at a discount $\delta = p^{\mathrm{cr}}/p^{\mathrm{in}}$ (0.10 on Anthropic and most OpenAI models, 0.02–0.03 on DeepSeek's API in 2026). The cache is keyed to the serving model, so a switch bills again for context that another model has already read. Let $c_{\mathrm{seen}}(m')$ be the longest prefix the current request shares with the destination's previous call (0 if none) and $q \in [0, 1]$ the fraction of it billed as a cache read, the effective retention. The router knows the shared prefix exactly, since it sent both requests. While context only grows it is the destination's previous prompt. Summarising, truncating or swapping retrieved text makes it shorter. The switching tax is

$$d_{m'}(t) \;=\; \big(p^{\mathrm{cw}}_{m'} - p^{\mathrm{cr}}_{m'}\big) \cdot \big(c_{t-1} - q\, c_{\mathrm{seen}}(m')\big). \tag{1}$$

A first visit pays for the whole context. A return pays only for the context added since the destination's last call. Measured q lies between 0.81 and 0.99 on six deployments (§4).

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.

Break-even horizon. A downgrade from tier i to a cheaper tier j saves $p_i^{\mathrm{cr}} - p_j^{\mathrm{cr}}$ per cached token per step and pays the tax on the way down and on the way back. At fixed context, with j visited for the first time, the round trip repays after

$$N^*_{i\to j}(q) \;=\; \frac{(p_j^{\mathrm{cw}} - p_j^{\mathrm{cr}}) + (1-q)(p_i^{\mathrm{cw}} - p_i^{\mathrm{cr}})}{p_i^{\mathrm{cr}} - p_j^{\mathrm{cr}}}, \qquad p_i^{\mathrm{cr}} > p_j^{\mathrm{cr}}, \tag{2}$$

steps. When $N^*(q)/T$ is well below 1, following demand is cheaper; when it is well above 1, committing to one model is cheaper (§6); §5.1 adds output prices. At q=0 both legs are rebuilt; at q=1 only the outbound leg is.

Three deployment choices. The harness designer sets T, the serving stack sets q, and whoever assembles the pool sets the price sheet; each moves a deployment between regimes (§5). At q=0 a production pool sits at $N^*/T = 0.92$ and six published harnesses span 0.13–0.92; at the measured retention the same pool sits at 0.12–0.27.

Contributions. Table 1 lists each claim with its evidence.

Table 1: Claims and evidence.

claim	evidence	basis
Switching to an unused model re-bills the prefix at the write rate, and charging every switch a full rebuild overstates the tax	315/322 such switches read nothing; 1.03× Eq. 1 in a growing loop; overstated 3.0× [2.5, 3.5] on agent runs and 2.2–2.7× on replays (§4)	measured, six deployments
A return re-reads most of the destination's last prefix	fitted q from 0.81 to 0.99; one pair 0.97 on its vendor's API, 0.81 and 0.83 on two re-hosts; with each deployment's cache block and minimum prefix the bill error is 0.007–0.053 (§4)	measured, six deployments
$\Psi(q)$ picks the regime outside $[0.2, 3]$	closed form for N^* exact on 1,636 pairs; outside the band right on all 206 and 104 held-out pools at q=1 and q=0 (§3, §5)	computed, 178 price sheets
The diagnostic names the cheaper fixed policy	on trajectories with $T \geq 9$, peer pool: commit $1.027 \times$ OPT and nothing cheaper; ladder pool: demand-following $1.003 \times$ OPT, with the lower bill at every $T \in [4, 44]$ and cheaper at every $T \in [3, 12]$ in live replay (§4, §8)	re-billed with per-step requirements; replayed
Guarantees	holding the session maximum optimal on price-flat pools for every q (Thm. 2); competitive threshold (Thm. 1); prediction-error identity (Prop. 1); worst-case linear regret for $q < 1$ (Thm. 3)	proof, ordered tiers
Noise-scaled gate	Prop. 4; held-out $\theta^\star/\hat{\sigma} = 1.04 \pm 0.18$ on five routers (§7)	proof and 12k queries
Deployed sessions pay the tax	20% of the median bill at q=1 and 47% at q=0, with the router's own logged decisions (§8)	re-billed, 1,785 sessions

OPT is the cheapest feasible plan in hindsight, computed exactly under each billing semantics (App. B).

2 Related Work

Multi-turn and agentic routing. Turn-level routers choose a model at each step (Zhang et al., 2026c; 2025a; Xu et al., 2026b; Zhou et al., 2026; Zhang et al., 2026b; Liu et al., 2026), step-level benchmarks exist (Yang et al., 2026; Li et al., 2026a), and Feng et al. (2026) unify router development and evaluation. All of them price a switch at zero. Si et al. (2026); Zhang et al. (2026a); Bhatti et al. (2026); Huang et al. (2026) learn stepwise, budget- or SLA-aware routers by RL; others match queries to logs, learn from sparse feedback or

pick the smallest feasible model (Lv et al., 2026; Woisetschläger et al., 2026; Wei et al., 2026; Jitkrittum et al., 2025; Wang et al., 2025), as surveyed by Moslem et al. (2026). Son et al. (2026) escalates at most once per trajectory, and Raj et al. (2026) pins one model at admission, as our peer-regime rule does. Ganz et al. (2026) measure the quality and cost of letting a second model continue a trajectory. Their tax comes from the non-native history and ours from cache billing; the two add. Stage-wise and batched routing (Wheeler et al., 2025; Xu et al., 2026a) change what a decision costs without tying it to the destination's cache. Sequential orchestration (Amin, 2026; Cheng et al., 2026; Chen et al., 2026), routing under budgets (Panda et al., 2025; Wu & Silwal, 2025; Li et al., 2026b; Mei et al., 2025) and preference-conditioned routing (Zeng et al., 2026) use per-decision costs without cache state. Single-turn routing (Hu et al., 2024; Ong et al., 2025; Lai et al., 2026; Shirkavand et al., 2025; Yu et al., 2026; Pulishetty et al., 2025; Kotte, 2026) optimises a memoryless objective, and Garg & Sagtani (2026) show how evaluation artifacts distort routing headroom. Cache-aware serving (Kwon et al., 2023; Chen et al., 2025; Woo et al., 2026b; Ojewale, 2026; Song, 2026; Lumer et al., 2026; Shen et al., 2025; Gao et al., 2024; Gim et al., 2024; Li et al., 2025b; Cheng, 2026) maximises hit rates for a fixed model. Zhang et al. (2025b) optimise eviction and Yao et al. (2026) derive LRU hit ratios for multi-turn conversations under finite memory, of which our scalar q is a per-deployment summary; Dai et al. (2025); Tiwari et al. (2026); Jin et al. (2026); Kabir et al. (2026); Zha et al. (2026) schedule, profile, place or delegate agent work. Khailo (2026) derive a keepalive break-even from the same read/write ratio for a single model, and Zhu et al. (2023) jointly optimise a query cache and a model multiplexer under one shared cache. Cross-model cache reuse (Li et al., 2025a; Woo et al., 2026a; Liu et al., 2024; Heo et al., 2026) lowers the tax within an architecture family. Online algorithms. We build on ski rental (Karlin et al., 1988), metrical task systems (Borodin et al., 1992) and algorithms with predictions (Purohit et al., 2018); Mukherjee et al. (2025) switch online between a custom model and an LLM under drift. Our movement cost depends only on the destination and its history, and grows over time. Constant switching costs give $\tilde{\Theta}(T^{2/3})$ bandit regret (Dekel et al., 2014; Arora et al., 2012); growth makes it linear (Thm. 3).

3 The Switching Tax and Its Closed Form

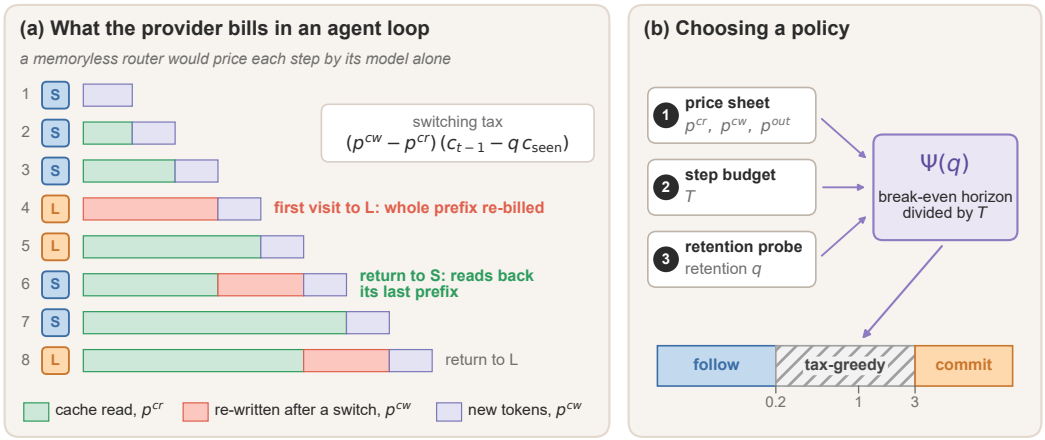

Figure 1: What a switch costs, and how to pick a policy. (a) Each row is a loop step and bar length is the billed context. Green is prefix read back from the serving model's cache at p^{cr}, lavender is new tokens at p^{cw}, and coral is prefix that model has not cached and is re-billed at p^{cw}. The first visit to a model re-bills the whole prefix (step 4, Eq. 1 with c_{seen}=0); a later return reads back what that model last cached and pays only for the growth since (step 6). At the median read discount 0.10 and the median write premium 1.25 of models that list one (both relative to p^{in}), the coral tokens cost 12.5× the green ones. Output is billed the same way in both cases and omitted. (b) The prescription takes a price sheet, the step budget T and a retention probe, forms $\Psi(q)$, and reads it against the band $[0.2, 3]$: follow demand below, commit above, tax-greedy inside.

Four-bucket billing. At step t a request carries context c_{t-1}, adds g_t new tokens and emits o_t (notation in App. A). With an append-only context, holding model m costs $p_m^{\mathrm{cr}} c_{t-1} + p_m^{\mathrm{cw}} g_t + p_m^{\mathrm{out}} o_t$. Switching to m' re-reads $w = q\, c_{\mathrm{seen}}(m')$ and costs $p_{m'}^{\mathrm{cr}} w + p_{m'}^{\mathrm{cw}}(c_{t-1} + g_t - w) + p_{m'}^{\mathrm{out}} o_t$. Its excess over a warm call to m', which reads all of c_{t-1}, is Eq. 1. The price gap between m and m' comes on top. The tax depends only on the destination, grows with context, and is charged separately on the way down and on the way back. Eq. 1 describes models with separate prefix caches; cross-model KV transfer (Heo et al., 2026) and shared backends lower it. We report policy results at q=0, the literal reading of a price sheet, and at q=1. Every measured q gives the same cheapest policy (§8).

Closed form. When two tiers share a discount δ and a write premium $\gamma = p^{\mathrm{cw}}/p^{\mathrm{in}}$, Eq. 2 depends only on their price ratio $r = p_i^{\mathrm{in}}/p_j^{\mathrm{in}} > 1$:

$$N^*(q) \;=\; \left(\frac{\gamma}{\delta} - 1\right) \cdot \frac{1 + (1-q)\, r}{r - 1}. \tag{3}$$

It matches the four-price form to 2×10^{-14} on the 1,636 of 15,248 census pairs that share both δ and γ. The regime therefore depends on the shape of the pool, not on its price level. Without retention a shared sheet has a floor, $N^*(0) \to \gamma/\delta - 1$ as $r \to \infty$: at γ=1, δ=0.1 no downgrade repays in fewer than 9 steps. Retention lowers the floor to $(1-q)(\gamma/\delta - 1)$, and at q=1 the horizon shrinks by a factor $1+r$.

Most pairs do not share a discount: δ is 0.10 for 88 models, 0.20 for 22, 0.50 for 17 and 0.25 for 14. Since $p_i^{\mathrm{cr}} - p_j^{\mathrm{cr}} = p_j^{\mathrm{in}}(\delta_i r - \delta_j)$, a shallow discount on the upper tier makes holding expensive and a downgrade repay fast. On the 73.4% of pairs with different discounts the median $N^*(0)$ is 10.0 and the median $N^*(1)$ is 1.4. Eq. 3 gives a design rule for building pools (App. F).

4 Retention, Measured on Real Agents

Providers report the cache buckets of every call, so Eq. 1 can be checked directly. We ran a tool-using agent on SWE-bench Verified instances in their official containers and routed between two models per pool with a demand-following and a commit policy. For every call we recorded the tokens billed as cache hit, cache miss and output. Each trajectory starts with a unique nonce, so no trajectory inherits another's cache. Costs are provider token counts times list prices. The data comprise 20 runs, 211 trajectories and 3,473 calls ($6.49).

A pool is measurable when both caches hit on consecutive same-model calls and a prefix sent to one model does not hit on the other. Fourteen of twenty-one probed models pass (App. I). The two agent pools are DeepSeek's chat→reasoner and inkling→glm-5-turbo on OpenRouter; the third row of Table 2 adds four pools run before this check.

First visits and returns. Of 206 switches to an unused model, 199 read nothing. In a controlled growing loop the first-visit tax matches Eq. 1 to 1.03× (App. H). Returns re-read the prefix the destination last cached. The hit ratio is 0.995 on DeepSeek and 0.974 on OpenRouter, every hit ending on a 64-token block boundary, and the DeepSeek shortfall sits on prefixes under 1.7k tokens (App. I). Charging every switch a full rebuild overstates the billed tax 3.0×.

Bill prediction. For each trajectory with a return we predict the total bill from its model and context sequence under a given q. The error at q=0 is 3.4–18.7× the error at q=1. The q that best fits costs (0.94–0.99) agrees with the q read from cache counts (0.97–0.995). Fig. 2 plots every trajectory.

Retention differs by deployment. We replay the switch schedules and context growth of the 29 DeepSeek trajectories that contain a return on four more deployments: DeepSeek's API (deepseek-flash→deepseek-v4-pro), the same pair on Alibaba Cloud Bailian and on Volcano Engine Ark, and Qwen3.8 Flash→Max on Bailian. Tool outputs are files from each instance's repository (App. I). All 116 first visits read nothing, and full-rebuild billing

Table 2: Eq. 1 against the bill. Overstated: tax at q=0 divided by billed tax, $(p^{\mathrm{cw}}-p^{\mathrm{cr}})$ times the part of c_{t-1} not read, over all switches / returns. Hit ratio: tokens read on a return divided by the destination's previous prompt. (B, M): cache block and minimum cacheable prefix. B is the provider's documented block where it matches the reads and otherwise the granularity every read shares; M is the shortest prefix that ever reads back, taken from the step in hit rate against c_{seen} in the same replay. Cold miss: first visits that read nothing. Bill error: mean $|\ln(\widehat{\mathrm{bill}}/\mathrm{bill})|$ per trajectory with a return, predicting at q=0, at q=1, and at q=1 with reads floored to B and dropped below M. $\hat{q}$: the scalar retention that minimises bill error. Replayed rows re-issue the switch schedules of 29 DeepSeek trajectories on each deployment. Trajectory-bootstrap 95% intervals and Wilcoxon tests are in App. I.

pool	traj.	returns	overstated	hit ratio	cold miss	(B, M)	bill error q=0/q=1/(B, M)	$\hat{q}$
DeepSeek chat→reasoner	150	348	3.1/6.0×	0.995	145/145	—	0.315/0.017/—	0.99
OpenRouter inkling→glm	23	64	3.6/4.1×	0.974	22/23	—	0.242/0.071/—	0.94
all agent runs (six pools)	211	472	3.0/4.9×	0.963	199/206	—		
replayed schedules								
DeepSeek flash→v4-pro	29	120	2.7/3.7×	0.949	29/29	(64, 0)	0.516/0.051/**0.028**	0.97
Bailian DeepSeek flash→v4-pro	29	120	2.2/2.8×	0.703	29/29	(1024, 1024)	0.403/0.268/**0.007**	0.81
Bailian Qwen3.8 flash→max	29	120	2.2/2.8×	0.632	29/29	(1024, 1024)	0.376/0.233/**0.008**	0.81
Ark DeepSeek flash→v4-pro	29	120	2.3/2.8×	0.527	29/29	(256, 2048)	0.478/0.333/**0.053**	0.83

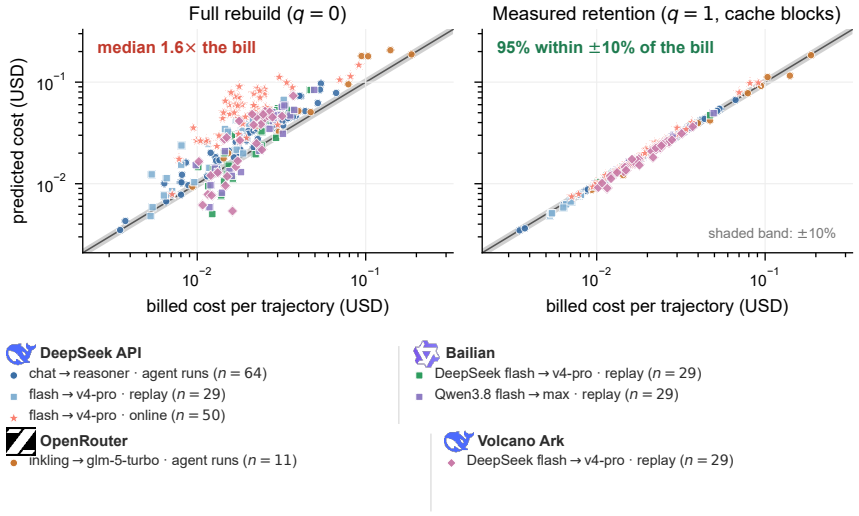

Figure 2: Predicted against billed cost. One point per trajectory that returns to a model it used before (241 in all). Left: charging every switch a full rebuild (q=0) overstates the bill by a median 1.6×, and 11% of trajectories land within ±10%. Right: reading the prefix back (q=1), floored to each deployment's block B and dropped below its minimum M, lands 95% within ±10% (mean $|\log|$ 0.025).

overstates the tax 2.2–2.7×. The same model pair has $\hat{q} = 0.97$ on DeepSeek, 0.81 on Bailian and 0.83 on Ark, and the Ark and DeepSeek price sheets for the strong tier are identical. It is steady in time: batches a day apart read back 0.989 and 0.986 on DeepSeek, an hour apart 0.970 and 0.961 on Ark.

Two deployment constants explain the gap. Plotting hit rate against c_{seen} gives a step. Returns read back nothing below a minimum prefix M and the whole prefix above it, floored to a block B. The step sits at M=0 on DeepSeek, 1,024 on Bailian and 2,048 on Ark, with B=64, 1,024 and 256. Predicting with q=1 and that pair, the bill error falls to 0.007–0.053 on all four deployments. On Ark the scalar q alone cuts the error only 1.4× against q=0 (p=0.06), while (B, M) cuts it 9.1×. Fitted on half the trajectories and scored on the rest, the error at the fitted q is 14× lower than at q=0 on DeepSeek and 2.0–2.1× lower on Bailian ($p \leq 0.003$). The two constants move the bill but not the policy. With the retention they imply in place of q, every deployment stays in the same band segment at both horizons (App. I).

When the context is not append-only. Every step of the 295 trajectories we measure extends the previous context, so the shared prefix is the previous prompt throughout. Loops that summarise or swap retrieved text break that. We replay the same schedules while compacting the context every four steps, and while swapping one retrieved block per step. Reads follow the shared prefix, not the prompt. On the 57 switches whose shared prefix is empty the provider reads back nothing, every time. Predicting bills from the shared prefix puts 87–100% of trajectories within ±10%, and predicting them from the previous prompt puts 0–7%. On a pool where the diagnostic calls for commit, commit stays cheaper at every horizon from 3 to 12 steps (0.63–0.88×) with the context compacted (App. I).

Replay on the live API. A teacher-forced replay appends each canonical assistant message, so both policies bill identical contexts. On the DeepSeek pool blind-follow is cheaper at every horizon from 3 to 12 steps, significantly from T=6 (App. I); §5 repeats the comparison on four more pools and §8 bills logged trajectories with per-step labels.

5 The Deployment Sets the Regime

5.1 Price sheets

We compute Eq. 2 for all 15,248 orderable pairs among the 178 models on a commercial aggregator that publish a cache rate: the median N^* is 11.8 steps at q=0 and 1.5 at q=1, with 58% and 88% of pairs repaying within 13 steps. Commit remains the rule where read prices nearly coincide, the setting of Thm. 2.

We read the break-even horizon over the step budget with a band. A downgrade also changes the output price, so the diagnostic adds that saving to Eq. 2,

$$\Psi(q) = \frac{1}{T} \cdot \frac{(p_j^{\mathrm{cw}} - p_j^{\mathrm{cr}}) + (1-q)(p_i^{\mathrm{cw}} - p_i^{\mathrm{cr}})}{(p_i^{\mathrm{cr}} - p_j^{\mathrm{cr}}) + \omega\,(p_i^{\mathrm{out}} - p_j^{\mathrm{out}})}, \tag{4}$$

where ω is the harness's output-to-context token ratio, taken from its logs. A pool's Ψ is the median over its orderable pairs. It equals $N^*(q)/T$ when output prices match. Every rank exchange we observe falls in $[0.2, 3]$, whether we sweep T or prices, weight sessions by bill or by optimum, or bill at q=0 or q=1 (Fig. 3, App. J). Outside the band these measurements agree on the cheaper policy, and inside it the trajectory decides. The band is the envelope of four flip points measured elsewhere. On 250 simulated pools that did not set it, the rule picks the cheaper policy outside the band on all 206 such pools at q=1 and all 104 at q=0 (App. J). Against the exact optimum the whole rule costs $1.01 \times \mathrm{OPT}$ at the median at q=1 and 1.12× at q=0, the same as the better fixed policy chosen in hindsight. Most of the gap at q=0 lies inside the band, where tax-greedy pays $1.22 \times \mathrm{OPT}$ (Table 4). On the re-billed trajectories with $T \geq 9$ tax-greedy stays within 9% of OPT on both pools. On three live pools inside the band the winning fixed policy differs, demand-follow on one and commit on two, and tax-greedy is 0.99–1.05× the winner at every T from 3 to 12 (App. I). On the first, the price sheet alone calls commit at 9 of the 10 horizons. Without the output term, balanced accuracy on 250 pools falls from 82% to 64% at q=0; at q=1 the input-only ratio already reaches 90% (App. G).

Harness horizons. Reported trajectory lengths vary 7.2× across harnesses (Table 7), which at q=0 moves N^*/T from 0.92 to 0.13; the production harness sits 8% below 1, where a small change of loop length flips the cheaper policy. At q=1 every harness sits far below 1 and demand-following bills less at every $T \in [4, 44]$ (64–92% of commit's; App. J).

5.2 A production anchor

The corpus (JetBrains Research, 2026) holds 1,785 logged SWE-agent trajectories with the model chosen at every turn between gpt-5.2 and gpt-5-mini (Table 3). 99.3% of them use more than one model, at a median switch rate of 0.500. Multi-turn benchmarks usually build sessions from single-turn queries. Re-ordering the same queries under one real router, a semantically coherent order matches the observed switch rate to 1.8%, and a random order overstates it by 36%. We use the coherent construction.

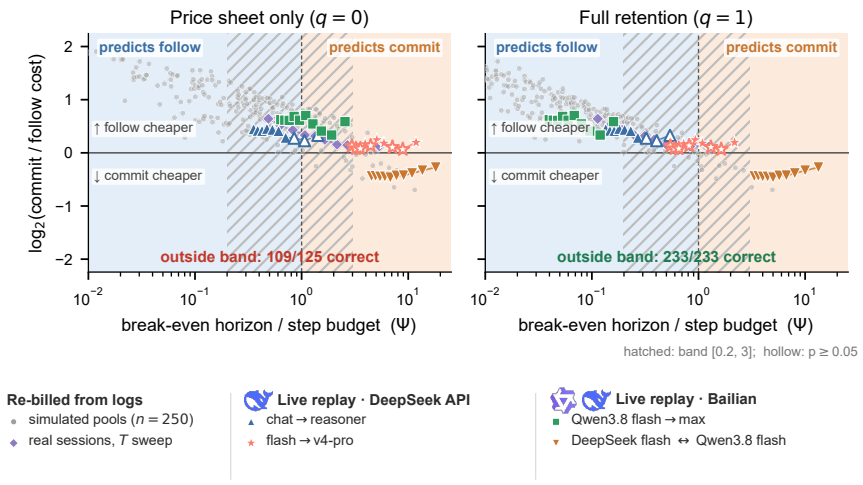

Figure 3: The diagnostic picks the cheaper policy outside the band. Horizontal: $\Psi(q)$. Vertical: $\log_2$ of the commit bill over the demand-follow bill, so points above the line favour demand-follow. Left: price sheet alone (q=0); right: full retention (q=1). Outside the hatched band $[0.2, 3]$ the price sheet calls 109 of 125 points correctly (100 of 107 pools), full retention 233 of 233 (210 pools). Points are 250 simulated pools and four real pools swept over T. Hollow: live-replay points with paired Wilcoxon $p \geq 0.05$ over trajectories.

Table 3: Left: a deployed-agent corpus. Right: session constructions used in the literature, compared with it.

Production corpus		Construction	x	vs. real
Trajectories	1,785	Single query, no history	0.000	−100%
Turns T	med. 13	Semantically coherent	0.509	+**1.8**%
Using >1 model	99.3%	Random shuffle	0.681	+36%
Switch rate x	0.500	Production (measured)	**0.500**	—
N^*/T at q=0 / 1	0.92 / 0.12			

5.3 Real pools

On the ladder pool at q=0, $N^*/T = 2.20$ lies inside the band, and re-billing shows that blind-follow is cheaper (Table 5). The ratio Ψ of App. G, which adds the output saving of a downgrade, also picks follow. At q=1, $N^*(1)/T = 0.35$ points to follow directly. On the peer pool commit is cheaper under both semantics. Sweeping price spread at fixed T, the two policies exchange rank at $N^*(0)/T \in [0.19, 0.39]$ and $N^*(1)/T \in [0.28, 0.45]$. On 250 random pools at T=13 billed at q=1, $N^*(1)/T$ picks the cheaper policy with 90% balanced accuracy, and $N^*(0)/T$ with 63%.

6 Policies with Guarantees

Setting. Tiers are totally ordered by capability, so each step has a lowest sufficient tier ℓ_t. Our labelled corpus records that tier and not each tier's outcome, so the ordering is an assumption here rather than something we test; pools whose models are incomparable fall outside this analysis. A step that requires ℓ_t forces every policy up; coming back down is optional and costs a tax. Choosing when to come down is a ski-rental problem in which both the rent and the purchase price grow. Retention changes only the purchase price. Proofs for every q are in App. M.

Theorem 1 (Growing-cost ski rental, K-independent). With K tiers ordered by capability, prices non-decreasing in tier, forced-up/optional-down dynamics and a context that grows monotonically, measure cost in excess of running warm at the requirement, and compare with the best policy that never upgrades before a requirement forces it. Inside a downgrade

window let D_t be the downgrade tax of Eq. 1, ρ bound its growth over the window, and κ bound, relative to it, the upgrade taxes paid after the downgrade until the requirement returns to the held tier. The threshold policy, which holds while accumulated rent stays below $\sqrt{1+\kappa}\, D_t$ and then drops, is $(1+\sqrt{1+\kappa})\rho$-competitive for every q. The constant does not depend on K, and as $\rho \to 1$ no deterministic online policy achieves a smaller one. With a free return (κ=0) the ratio is the classical 2ρ.

The bound holds on 3,000 two-tier and 2,694 three-tier random instances billed exactly at $q \in \{0, 0.5, 1\}$, reaching 0.95 and 0.98 of its value (App. M). With two tiers, κ can be set online at each step to $\hat{\kappa}_t = (p_h^{\mathrm{cw}} - p_h^{\mathrm{cr}})(\chi - q)\, c_{t-1}/D_t$, where χ bounds context growth to the end of the session; the guarantee then holds with $\max_t \hat{\kappa}_t$ in place of κ (App. M). Both inputs are available before the call. The harness fixes the step limit, and χ can be the growth so far carried over the remaining steps. The constant is independent of K because the session splits into holding episodes under forced-up, optional-down dynamics. When $N^*(q) \gg T$ the threshold never triggers. The policy that remains is commit: jump to the predicted session-maximum tier, escalate only when forced, and never come down.

Theorem 2 (Commit is optimal on price-flat pools). On a price-flat pool ($p_m^{\mathrm{cr}} \equiv p^{\mathrm{cr}}$, $p_m^{\mathrm{cw}} \equiv p^{\mathrm{cw}}$, $p_m^{\mathrm{out}} \equiv p^{\mathrm{out}}$), for every retention q: (i) the offline optimum makes zero switches and holds $L^* = \max_t \ell_t$ throughout; (ii) commit to L^* with perfect prediction equals OPT.

Proposition 1 (Prediction-error identity). On a price-flat pool, commit with a session-maximum predictor satisfies cost(commit)/OPT $= 1+\eta\bar{\kappa}$, where η is the fraction of sessions the predictor under-shoots and $\bar{\kappa}$ is the mean corrective tax of under-shot sessions divided by the mean OPT of all sessions. Both sides are independent of q. On 81 multi-step trajectories a first-step predictor (η=0.54) gives $1 + \eta\bar{\kappa} = 1.0783$ against a measured 1.0783 (App. J).

Near-flat pools. We perturb p_m^{cr} and p_m^{cw} by $\pm\varepsilon$ around a common level, with equal output prices (20 pools per ε). OPT makes zero switches on every trajectory up to ε=0.50, and commit/OPT is 1.0076 at ε=0.10 and 1.0156 at ε=0.20, at both q=0 and q=1.

Proposition 2 (Peer-pool lower bound). For every ε-spread peer pair and every q there are request sequences on which any deterministic online algorithm pays at least $(2 - o(1)) \cdot \mathrm{OPT}$ and any randomised algorithm at least $(\frac{e}{e-1} - o(1)) \cdot \mathrm{OPT}$, with $o(1) \to 0$ as $N^*(q)$ grows.

Per-step learning. Model peer-pool quality routing as a K-armed bandit in which a switch at round t to an arm last played at round t_{seen} costs $\lambda(t - q\, t_{\mathrm{seen}})$. This is Eq. 1 when context grows by $\bar{g}$ tokens per round, with $\lambda = (p^{\mathrm{cw}} - p^{\mathrm{cr}})\bar{g}$ in dollars. With a constant switching cost c the minimax regret is $\tilde{\Theta}(c^{1/3}T^{2/3})$ (Dekel et al., 2014).

Theorem 3 (Impossibility). For K-armed adversarial bandits with this switching cost, $\lambda \le 1$, $q < 1$ and $T \ge 2/((1-q)\lambda)$, every algorithm suffers worst-case policy regret $\tilde{\Omega}(((1-q)\lambda)^{1/3}T)$, linear in T. Mini-batched EXP3 attains $\tilde{O}(\lambda^{1/3}K^{1/3}T)$ for every q.

Proposition 3 (Explore then commit). If losses are i.i.d., explore-then-commit with round-robin exploration of length $T_0 = (T\sqrt{K}/\lambda)^{2/5}$ achieves $\tilde{O}(\lambda^{1/5}K^{2/5}T^{4/5})$ for every q. At q=0 every offline-optimal switch lies in $\{t : \lambda t \le \Delta_{\max}(T-t+1)\}$, and the best delayed-commit cutoff is $t^*_{\mathrm{cut}} \approx v/\lambda$ for a per-round gain v.

At $q = 0.81$–0.99 the constant is smaller by $(1-q)^{1/3} \approx 0.2$–0.6. The bound concerns policy regret against adversarial losses; it does not say that a learner facing structured stochastic demands, bounded windows or session resets incurs linear regret, and Prop. 3 is the stochastic counterpart. In simulation explore-then-commit reaches a log–log slope of 0.79 against the predicted 0.80 (App. M).

7 A Noise-Scaled Gate

On a peer pool a downgrade never repays, so a router switches only for quality. Near-tie margins flip the per-step argmax (Lai et al., 2026), and Lee et al. (2025) route each reasoning step on the small model's confidence without pricing the switch. A gate switches only when

the predicted gain exceeds a threshold θ, scored by per-step accuracy minus λ_{tax}=0.06 times the switch rate.

Under Gaussian margin noise $\hat{G}_t = G_t + \varepsilon_t$ and a density condition near the optimum, every interior minimiser of the expected error cost satisfies

$$\theta^\star = \sigma\sqrt{2\ln\Lambda}\,\big(1 + O(1/\ln\Lambda)\big) \qquad (\Lambda \to \infty), \tag{5}$$

where $\Lambda = \bar{d}\bar{x}/(V\bar{G})$ collects the mean tax, the rate of zero-gain steps and the value of a unit of quality (Prop. 4, App. L).

A gate tuned on one router must therefore be rescaled by the noise of another, and retention enters only through $\bar{d}$ inside the logarithm.

Result. We estimate σ by the spread of predicted top-2 margins on true ties and keep the five routers whose margins predict the sign of the gain (Tables 10, 11 and Fig. 8 in App. L). Selecting $\theta^\star$ on half of the sessions and scoring the other half (5 splits) gives $\theta^\star/\hat{\sigma} = 1.04 \pm 0.18$ with Spearman correlation 0.90, and setting $\theta = a\hat{\sigma}$ with a fitted on the other four routers recovers 81–100% of each router's gating gain. Gating raises the objective for all five. Switch rates fall from 0.40–0.67 to 0.04–0.25 and accuracy changes by -1.4 to $+1.1$ points. But one router supplies 91% of the spread in $\hat{\sigma}$; without it the other four span 1.8×, the held-out ratio is 0.96 and the Pearson correlation loses significance (App. L).

8 System Audit Under Both Billing Semantics

Re-billing two routing benchmarks. RouterBench and LLMRouterBench are single-turn. In a counterfactual simulation we place their per-query argmax router in a T-step loop and bill it with Eq. 1. Over $T \in [4, 44]$ the four-bucket bill exceeds the memoryless bill by 1.5–9.3× at q=0 and 1.5–5.5× at q=1 on RouterBench, and by 1.3–5.2× and 1.3–3.7× on LLMRouterBench, where live prices cover 4 of its 20 models. Switching tax is 9–79% of the bill. At T=13 and q=1 the factors are 2.7× (47% tax) and 1.9× (26%) (App. H).

Logged sessions. We re-bill the 1,785 production sessions of §5.2 with the deployed router's own per-turn choices, estimating tokens as characters/4. Switching tax is 47% of the median session's bill at q=0 and 20% at q=1. At q=1 every session costs less than holding the stronger model (median 0.70×); at q=0 the median session costs 1.06× as much.

Eight policies. We bill the eight policies of App. J, from blind-follow to lookahead(4), on the same 77 trajectories against an exact optimum under each semantics. Every policy meets the requirement at every step. For every q from 0.8 to 1 the cheapest policy on each pool is the one in the q=1 column and costs move by at most 0.07; Bailian's blocks and minimum prefix, 64-token rounding, and caches expiring after 3, 5 or 10 idle steps all leave every ranking unchanged (App. J).

On the peer pool no policy is cheaper than commit under either semantics (Fig. 7); retention cuts blind-follow's excess from 54% to 30%, because every first visit still pays the full tax. On the ladder pool at q=1 blind-follow and tax-greedy reach 1.003 × OPT with the same schedule on 17 of the 21 trajectories; every damped or committed policy costs more.

In App. J a gate cuts a predict-argmax router's switching cost by 56%, and nine router families pay 1.00–2.98× a never-switch bill on a real 11-model pool, the cost-oriented ones most.

9 Conclusion

Provider caches make each routing decision change the price of later steps. A first visit pays for the whole context, and a return pays for what was added since that model's last call. Outside $\Psi(q) \in [0.2, 3]$ the ratio picks the policy, commit near price-flat pools and follow demand on ladders. Effective retention varies with the deployment. One model pair has 0.97 on its vendor's API and 0.81 and 0.83 on two re-hosts.

Reproducibility statement

Every number is produced by a released script from a stored result file. We release the price snapshot behind the census (178 models with four prices, provider and timestamp), $N^*(q)$ for all 15,248 pairs, the billing module with its self-test (it reproduces the single-slot dynamic program at q=0 and brute force at $q > 0$), and the per-call cache records of all 211 agent trajectories and 87 replayed schedules with the harness and probes. The production corpus is a public release; we pin the split. No API key is included.

Ethics statement

The work uses public benchmarks, a public trajectory corpus and paid API access. It involves no human subjects or personal data.

AI use statement

In this work, we used generative AI tools to implement the measurement and analysis code (including the code that samples the simulated price pools). The experiments were planned by the authors. We have not used generative AI tools to generate synthetic data sets or for translation, and qualitative or thematic data analysis is not applicable to this work. Additionally, we used generative AI tools to edit the paper for readability, search for and summarize related literature, create figures, write and edit code, and format references. We have reviewed all AI-assisted work. Reported numbers are regenerated by released scripts from stored result files, and key numbers are checked against the text by an automated consistency script. Every cited work was checked to exist. The models routed in our experiments are objects of study, not assistants. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Wasi Uddin Ahmad, Nikolai Ludwig, Somshubra Majumdar, and Boris Ginsburg. OpenSWE-Traces: Advancing dual-mode multilingual distillation for software engineering agents. arXiv preprint arXiv:2606.16038, 2026.

Danial Amin. Bayesian orchestration of multi-LLM agents for cost-aware sequential decision-making. arXiv preprint arXiv:2601.01522, 2026.

Raman Arora, Ofer Dekel, and Ambuj Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. In ICML, 2012.

Amit Singh Bhatti, Vishal Vaddina, and Dagnachew Birru. PROTEUS: SLA-aware routing via Lagrangian RL for multi-LLM serving systems. arXiv preprint arXiv:2601.19402, 2026.

Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task systems. Journal of the ACM, 39(4):745–763, 1992.

Chaoran Chen, Vy Nguyen, Ziji Zhang, Abhinav Gullapalli, et al. Retry, switch, or abstain? Learning strategy-aware tool-use policies via controlled error injection. arXiv preprint arXiv:2608.11977, 2026.

Peng Chen, Jiaji Zhang, Hailiang Zhao, Yirong Zhang, Shenyao Chen, Jiahong Yu, Xueyan Tang, Yixuan Wang, Hao Li, Jianping Zou, Gang Xiong, Kingsum Chow, Shuibing He, and Shuiguang Deng. Toward robust and efficient ml-based gpu caching for modern inference. arXiv preprint arXiv:2509.20979, 2025.

Huang Cheng. CacheRoute: Planned prefix-affinity routing for large-scale LLM serving. arXiv preprint arXiv:2608.19677, 2026.

Yujun Cheng, Enfang Cui, Hao Qin, Zhiyuan Liang, and Qi Xu. AgentGate: A lightweight structured routing engine for the internet of agents. arXiv preprint arXiv:2604.06696, 2026.

J. G. Dai, Tianze Deng, Yueying Li, and Tianyi Peng. Throughput-optimal scheduling algorithms for LLM inference and AI agents. arXiv preprint arXiv:2504.07347, 2025.

Ofer Dekel, Jian Ding, Tomer Koren, and Yuval Peres. Bandits with switching costs: $t^{2/3}$ regret. In STOC, 2014.

Tao Feng, Fangxu Yu, Haozhen Zhang, Zhongjie Dai, Liangqi Yuan, Zijie Lei, Weizhi Zhang, Kunlun Zhu, Haodong Yue, Keyang Xuan, Ge Liu, and Jiaxuan You. LLMRouter: Unified infrastructure for developing, evaluating, and deploying LLM routers. arXiv preprint arXiv:2608.06867, 2026.

Roy Ganz, Mor Shpigel Nacson, Adi Kalyanpur, and Ron Litman. The handoff tax: Continuing non-native trajectories in LLM agents. arXiv preprint arXiv:2608.24358, 2026.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. Cost-efficient large language model serving for multi-turn conversations with CachedAttention. In USENIX Annual Technical Conference (ATC), 2024.

Saloni Garg and Amit Sagtani. Unsolvability ceiling in multi-LLM routing: An empirical study of evaluation artifacts. arXiv preprint arXiv:2605.07395, 2026.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In Proceedings of Machine Learning and Systems (MLSys), 2024.

Taekyung Heo, Rasoul Shafipour, Ritchie Zhao, Maximilian Golub, Mohammad Mahdi Kamani, Ritika Borkar, Makesh Tarun Chandran, Pantea Zardoshti, and Bita Darvish Rouhani. Cross-model KV cache transfer in LLM families: A closed-form linear mapping for prefill reuse. arXiv preprint arXiv:2608.03893, 2026.

Qitian Jason Hu et al. Routerbench: A benchmark for multi-llm routing systems. In ICML, 2024.

Kaiyu Huang, Xingyu Wang, Mingze Kong, Zhubo Shi, Yuqian Hou, Hong Xu, Zhongxiang Dai, Minchen Yu, and Qingjiang Shi. Uniscale: Adaptive unified inference scaling via online joint optimization of model routing and test-time scaling. In International Conference on Machine Learning (ICML), 2026.

JetBrains Research. agent-trajectories-swe-bench-test-minus-verified. Hugging Face dataset, https://huggingface.co/datasets/JetBrains-Research/agent-trajectories-swe-bench-test-minus-verified, 2026.

Yongjia Jin et al. Boute: Cost-efficient llm serving with heterogeneous llms and gpus via multi-objective bayesian optimization. In Proceedings of Machine Learning and Systems (MLSys), 2026.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient LLM inference. arXiv preprint arXiv:2502.08773, 2025.

Ahasan Kabir, Jiaqi Xue, Mengxin Zheng, and Qian Lou. HW-Router: Hardware-aware routing for scalable multi-LLM serving. arXiv preprint arXiv:2608.14575, 2026.

Anna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator. Competitive snoopy caching. Algorithmica, 3:79–119, 1988.

Maxim Khailo. Keeping the cache warm pays: Keepalive economics for agentic workloads. arXiv preprint arXiv:2607.19214, 2026.

Varun Kotte. UCCI: Calibrated uncertainty for cost-optimal LLM cascade routing. arXiv preprint arXiv:2605.18796, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In SOSP, 2023.

Han Lai et al. When routing collapses: On the degenerate convergence of llm routers. arXiv preprint arXiv:2602.03478, 2026.

Sangmook Lee, Dohyung Kim, Hyukhun Koh, Nakyeong Yang, and Kyomin Jung. Confidence-guided stepwise model routing for cost-efficient reasoning. arXiv preprint arXiv:2511.06190, 2025.

Allison Li, Kristjan Greenewald, Thomas Parnell, and Navid Azizan. Efficient multi-adapter LLM serving via cross-model KV-cache reuse with activated LoRA. arXiv preprint arXiv:2512.17910, 2025a.

Hanchen Li, Runyuan He, Qiuyang Mang, Qizheng Zhang, et al. Continuum: Efficient and robust multi-turn LLM agent scheduling with KV cache time-to-live. arXiv preprint arXiv:2511.02230, 2025b.

Siyuan Li et al. Llmrouterbench: A unified benchmark for multi-llm routing. arXiv preprint arXiv:2601.07206, 2026a.

Yifei Li, Zihui Gao, and Laks V. S. Lakshmanan. WISERouter: LLM routing with workload budget constraint. arXiv preprint arXiv:2607.23765, 2026b.

Xinchen Liu, Hang Zhou, Yingjie Zong, Yuchuan Tian, et al. Agentic routing: The harness-native data flywheel. arXiv preprint arXiv:2607.11399, 2026.

Yuhan Liu, Yuyang Huang, Jiayi Yao, Shaoting Feng, Zhuohan Gu, Kuntai Du, Hanchen Li, Yihua Cheng, Junchen Jiang, Shan Lu, Madan Musuvathi, and Esha Choukse. DroidSpeak: KV cache sharing for cross-LLM communication and multi-LLM serving. arXiv preprint arXiv:2411.02820, 2024.

Elias Lumer, Faheem Nizar, Akshaya Jangiti, Kevin Frank, Anmol Gulati, Mandar Phadate, and Vamse Kumar Subbiah. Don't break the cache: An evaluation of prompt caching for long-horizon agentic tasks. arXiv preprint arXiv:2601.06007, 2026.

Bo Lv, Jingbo Sun, Jianwei Lv, Chen Tang, Shaojie Zhang, Nayu Liu, Guoxin Yu, Zihao Li, Qichao Zhang, Dongbin Zhao, Ping Luo, and Yue Yu. Beyond query memorization: Large language model routing with query decomposition and historical matching. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics, 2026.

Kai Mei, Wujiang Xu, Minghao Guo, Shuhang Lin, and Yongfeng Zhang. OmniRouter: Budget and performance controllable multi-LLM routing. arXiv preprint arXiv:2502.20576, 2025.

Yasmin Moslem et al. Dynamic model routing and cascading for efficient LLM inference: A survey. arXiv preprint arXiv:2603.04445, 2026.

Kalpan Mukherjee, Vikramank Singh, Abishek Sankararaman, Murali Narayanaswamy, and Tim Kraska. Robust online inference using adaptive model switching. In ICLR Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning, 2025.

Mubarak Adetunji Ojewale. Netkv: Network-aware decode instance selection for disaggregated llm inference. arXiv preprint arXiv:2606.03910, 2026.

Isaac Ong et al. Routellm: Learning to route llms with preference data. In ICLR, 2025.

Pranoy Panda, Raghav Magazine, Chaitanya Devaguptapu, Sho Takemori, and Vishal Sharma. Adaptive LLM routing under budget constraints. In Findings of the Association for Computational Linguistics: EMNLP 2025, pp. 23934–23949, 2025.

Roshini Pulishetty, Mani Kishan Ghantasala, Keerthy Kaushik Dasoju, et al. One head, many models: Cross-attention routing for cost-aware LLM selection. arXiv preprint arXiv:2509.09782, 2025.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In NeurIPS, 2018.

Ritik Raj, Souvik Kundu, Sarbartha Banerjee, Dheemanth Joshi, Ishita Vohra, and Tushar Krishna. Trace-router: Task-consistent and adaptive online routing for agentic ai. arXiv preprint arXiv:2607.22465, 2026.

Yu Shen et al. Learned prefix caching for efficient llm inference. In Advances in Neural Information Processing Systems (NeurIPS), 2025.

Reza Shirkavand, Shangqian Gao, Peiran Yu, and Heng Huang. Cost-aware contrastive routing for llms. In NeurIPS, 2025.

Wenwen Si, Insup Lee, and Osbert Bastani. Policy-guided stepwise model routing for cost-effective reasoning. arXiv preprint arXiv:2605.06116, 2026.

Seongho Son, Sangwoong Yoon, Jiahua Tang, Shuhan Wang, Lorenz Wolf, and Ilija Bogunovic. SWE-Router: Routing in multi-turn agentic software engineering tasks. arXiv preprint arXiv:2607.00053, 2026.

Yan Song. Cache-aware prompt compression: A two-tier cost model for llm api caching. arXiv preprint arXiv:2607.15516, 2026.

Devesh Tiwari et al. Cachewise: Understanding workloads and optimizing kvcache management for efficiently serving llm coding agents. arXiv preprint arXiv:2606.16824, 2026.

Chenxu Wang et al. ICL-Router: In-context learned model representations for LLM routing. arXiv preprint arXiv:2510.09719, 2025.

Jiaming Wei, Zekun Wu, Adriano Koshiyama, and Maria Perez-Ortiz. Routing is least learnable where it is most valuable: Bounds on representation routing for web agents. arXiv preprint arXiv:2608.06171, 2026.

S. Wheeler, M. Torgbi, and H. T. Madabushi. Reducing environmental costs whilst maintaining operational effectiveness in large language models through chain routing. In Proceedings of the 21st Conference on Natural Language Processing (KONVENS), pp. 250–273, 2025.

Herbert Woisetschläger et al. Cost-optimal LLM routing with limited user feedback under user satisfaction guarantees. arXiv preprint arXiv:2606.19376, 2026.

Sunghyeon Woo, Jaeeun Kil, Hoseung Kim, et al. ICaRus: Identical cache reuse for efficient multi-model inference. arXiv preprint arXiv:2603.13281, 2026a.

Sunghyeon Woo, Hoseung Kim, Sunghwan Shim, Minjung Jo, Hyunjoon Jeong, Jeongtae Lee, Joonghoon Kim, Sungjae Lee, Baeseong Park, Se Jung Kwon, and Dongsoo Lee. Prefillshare: A shared prefill module for kv reuse in multi-llm disaggregated serving. arXiv preprint arXiv:2602.12029, 2026b.

Fangzhou Wu and Sandeep Silwal. Efficient training-free online routing for high-volume multi-LLM serving. In Advances in Neural Information Processing Systems (NeurIPS), 2025.

Haotian Xu et al. Towards cost-effective LLMs routing with batch prompting. arXiv preprint arXiv:2605.28268, 2026a.

Zhongling Xu, Shunan Zheng, and Wei Wang. Seqroute: Global budget-aware sequential llm routing via offline reinforcement learning. arXiv preprint arXiv:2605.25424, 2026b.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In Advances in Neural Information Processing Systems (NeurIPS), 2024. arXiv:2405.15793.

John Yang, Kilian Leret, Carlos E. Jimenez, et al. SWE-smith: Scaling data for software engineering agents. arXiv preprint arXiv:2504.21798, 2025.

Pei Yang, Wanyi Chen, Tongyun Yang, Pengbin Feng, Jiarong Xing, Wentao Guo, Yuhang Yao, Yuhang Han, Hanchen Li, Xu Wang, Zeyu Wang, Jie Xiao, Anjie Yang, Liang Tian, Lynn Ai, Eric Yang, and Tianyu Shi. TwinRouterBench: Fast static and live dynamic evaluation for realistic agentic LLM routing. arXiv preprint arXiv:2605.18859, 2026.

Heyuan Yao, Chutong Gao, Yuan Lyu, Izzy Grosof, and David Simchi-Levi. Multi-turn LLM conversations under the least-recently-used policy: Mean-field asymptotics and hit ratio approximation. arXiv preprint arXiv:2609.02027, 2026.

Tao Yu, Yifei Qu, Zhiqing Cui, Pengfei Zhou, et al. SCOPE-Router: Cost-aware open-set VLM routing for execution-oriented tasks. arXiv preprint arXiv:2608.12127, 2026.

Jiahao Zeng, Ming Tang, and Ningning Ding. Learning to route LLMs from implicit cost-performance preferences via meta-learning. arXiv preprint arXiv:2606.06178, 2026.

Donghui Zha, Lingwei Xu, Linxiao Wu, Yixue Dong, and Haochen Li. CacheRouter: A dual-path tool routing architecture with cache-preserving main-model isolation for long-tail tool discovery. arXiv preprint arXiv:2608.22708, 2026.

Caiqi Zhang, Menglin Xia, Xuchao Zhang, Daniel Madrigal, Ankur Mallick, Samuel Kessler, Victor Ruehle, and Saravan Rajmohan. Budget-aware agentic routing via boundary-guided training. arXiv preprint arXiv:2602.21227, 2026a.

Haozhen Zhang et al. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. arXiv preprint arXiv:2506.09033, 2025a.

Jiarui Zhang, Xiangyu Liu, Yong Hu, Chaoyue Niu, Hang Zeng, Shaojie Tang, Fan Wu, and Guihai Chen. From myopic selection to long-horizon awareness: Sequential LLM routing for multi-turn dialogue. arXiv preprint arXiv:2604.12385, 2026b.

Wenxin Zhang, Yueying Li, Ciamac C. Moallemi, and Tianyi Peng. Tail-optimized caching for LLM inference. In Advances in Neural Information Processing Systems (NeurIPS), 2025b.

Yiqun Zhang et al. Mtrouter: Cost-aware multi-turn llm routing with history-model joint embeddings. arXiv preprint arXiv:2604.23530, 2026c.

Pengfei Zhou, Zhiwei Tang, Yixing Ma, Jiasheng Tang, Yizeng Han, Zhenglin Wan, Fanqing Meng, Wei Wang, Bohan Zhuang, Wangbo Zhao, and Yang You. Agent-as-a-router: Agentic model routing for coding tasks. arXiv preprint arXiv:2606.22902, 2026.

Banghua Zhu, Ying Sheng, Lianmin Zheng, Clark Barrett, Michael I. Jordan, and Jiantao Jiao. On optimal caching and model multiplexing for large model inference. arXiv preprint arXiv:2306.02003, 2023.

A Notation

prices (per 10^6 tokens)		workload and harness	
$p_m^{\text{in}},\ p_m^{\text{out}}$	input, output rate	c_{t-1}	context entering step t
p_m^{cr}	cache-read rate (warm prefix)	$g_t,\ o_t$	new input, output tokens
p_m^{cw}	cache-write rate (cold prefix)	T	harness step budget
$\delta = p^{\text{cr}}/p^{\text{in}}$	read discount (median 0.10)	ℓ_t	tier demanded at t
$\gamma = p^{\text{cw}}/p^{\text{in}}$	write premium (median 1.25)	$L^* = \max_t \ell_t$	session peak demand
$r = p_i^{\text{in}}/p_j^{\text{in}} > 1$	tier price ratio	$K,\ k$	pool size; cache slots
q	cache retention (Eq. 1)	$c_{\text{seen}}(m)$	prefix shared with m's last call
regime diagnostics		**policy and gate**	
$d_{m'}(t)$	switching tax (Eq. 1)	x	per-step switch rate
N^*	break-even horizon (Eq. 2)	$\theta^\star,\ \hat{\sigma}$	optimal gate; router noise
$N^*_{\text{full}},\ \Psi$	output-aware form (Eq. 4)	η	peak-prediction error rate
$\omega = o/c$	output-to-context ratio	$\bar{\kappa}$	tax-to-OPT ratio (Prop. 1)
ρ	context growth over a session	λ_{tax}	tax rate in the gate sweep

B Computing OPT

Given realised $\ell_{1:T}$, $c_{1:T}$, $g_{1:T}$, $o_{1:T}$, define $D[t, m]$ as the minimum cost of serving steps $1..t$ ending on model m. The transition uses the four-bucket costs of §3, always at the prices of the model m that serves step t. Holding m charges $p_m^{\text{cr}} c_{t-1} + p_m^{\text{cw}} g_t + p_m^{\text{out}} o_t$. Switching into m from another model charges $p_m^{\text{cr}} w + p_m^{\text{cw}}(c_{t-1} + g_t - w) + p_m^{\text{out}} o_t$ with $w = q\, c_{\text{seen}}(m)$. Infeasible states ($m < \ell_t$) are $+\infty$. At q=0 the state is the current model and the complexity is $O(TK^2)$. At $q > 0$ a step's cost depends on every model's last call, so the state becomes the current model together with each model's retained prefix; we prune it exactly, discarding a state when another with the same current model costs no more and retains at least as long a prefix on every model (a longer retained prefix can never raise a later cost). The implementation reproduces the q=0 program to the last digit and matches brute-force enumeration on random instances. On the 77 real trajectories at q=1 the pruned program keeps a median of 40 (ladder) and 63 (peer) states at its widest step, and at most 642. It takes a median of under 1 ms and at most 47 ms per trajectory. It matches brute-force enumeration on all 51 trajectories with $T \le 7$, for both pools and both semantics.

C Multi-slot cache billing, and the k sweep

The single-slot model assumes a switch destroys the cache. Under a k-slot model, returning to a model within k distinct intervening models re-reads at p^{cr}. Retention q=1 with unbounded slots is the k=∞ case, which is what the deployments of §4 approach. Under retention the return leg of a round trip is not re-charged, so $N^*(q)$ carries the factor $1 - q$. The two sweeps below apply to bounded caches.

How much k buys. On the real peer pool and the 77 trajectories of §5.3, billing demand-following against staying on the session-modal model gives 1.185 at k=1, 1.088 at k=2, 1.065 at k=3 and 1.044 at k=∞. Extra slots remove about three-quarters of the follow penalty. The rest is the first visit to each model, which is billed at p^{cw} for any k.

Keep-warm hedging. A policy can pay a small fee each step to keep a spare model warm. We price it with a one-step oracle that always keeps warm the model used next. It wins only at k=1 (1.139 vs 1.185). At k=$2, 3, \infty$ it costs more (1.116/1.101/1.095 vs 1.088/1.065/1.044), because the cache is already warm.

Router ranking across k. We re-ran the nine-router flagship audit over $k \in \{1, 2, 3, 5, 8, 11\} \times \{\text{LRU, FIFO}\}$, billing each router against staying on its own first pick. Ranges across the grid: FrugalGPT 1.37–2.13, Avengers-Pro 1.77–3.12, IRT 1.23–1.67, Em-

bedLLM 1.19–1.66, GraphRouter 1.21–1.63, RouteLLM 1.13–1.62, Avengers 1.10–1.35, RouterDC 1.11–1.12, HybridLLM 1.00–1.00. Levels move by up to 76%. In all 12 cells the two cost-oriented routers pay the most tax. LRU and FIFO differ by at most 0.12.

D Cache TTL

A TTL and the break-even. At q=0 the numerator of Eq. 2 charges both legs of a round trip. When the return lands inside the origin's cache window, only the outbound leg rebuilds, which is q=1: $N^*(1) = (p_j^{\mathrm{cw}} - p_j^{\mathrm{cr}})/(p_i^{\mathrm{cr}} - p_j^{\mathrm{cr}})$. The reduction factor is $(p_j^{\mathrm{cw}} - p_j^{\mathrm{cr}})/[(p_i^{\mathrm{cw}} - p_i^{\mathrm{cr}}) + (p_j^{\mathrm{cw}} - p_j^{\mathrm{cr}})]$. It depends on the two tiers, not on the TTL length, and is smallest for a downgrade to a cheap tier. A return that lands after expiry is billed at q=0.

Turn gaps. Provider caches expire after roughly 300 s of inactivity. If every turn gap exceeded that, a switch would cost no more than a hold. In the production corpus the median inter-turn gap is well inside the TTL, and in our replays a return reaches the destination 8–12 s after its previous call (median).

E Non-monotone context

Our billing treats context as append-only, the standing assumption of App. M. Real agents truncate, summarise and swap retrieved passages, so c_t need not grow. The tax of Eq. 1 is $(p^{\mathrm{cw}} - p^{\mathrm{cr}})$ times the part of the context the destination has not cached. Replacing part of the prefix shortens the shared prefix, so a switch pays more; but a model that stays also re-writes the replaced tokens, so the gap between switching and staying narrows. On 20 trajectories with a fraction f of the carried prefix replaced each step:

replaced fraction f	0.0	0.1	0.3	0.5	0.7	0.9
follow / stay	1.458	1.366	1.223	1.118	1.036	0.972
commit / OPT	1.024	1.029	1.045	1.073	1.114	1.174
follow / OPT	1.388	1.309	1.194	1.123	1.082	1.071

The follow-over-stay ratio falls monotonically and reaches 1 near $f \approx 0.9$. Commit and follow exchange ranks between f=0.5 and f=0.7, so a heavily retrieval-driven loop favours following even on a peer price sheet. Of the 81 production-style trajectories 2 are natively non-monotone, both mildly ($c_t/c_{t-1} \geq 0.93$).

F The shared-sheet design rule

Within a shared price sheet write $G = \gamma/\delta - 1$. Setting $N^*(q) \leq T$ in Eq. 3 shows a downgrade is worth attempting only when $r \geq (T + G)/(T - (1 - q)G)$, which requires $T > (1-q)G$ before any ratio suffices. At the common γ=1, δ=0.1 (G=9): without retention loops of nine steps or fewer are in the peer regime no matter how wide the pool; with full retention every loop admits a downgrade once $r \geq 1 + 9/T$.

G The output-aware diagnostic Ψ

Definition. We write $N^*_{\mathrm{full}}(q) = T\,\Psi(q)$ for the output-aware horizon of Eq. 4, with $\omega = o/c$. The protocol table below uses the numerator $p_i^{\mathrm{cw}} + p_j^{\mathrm{cw}}$; the retention table that follows reports Eq. 4 alongside it.

A non-positive denominator means the output side consumes the prefix saving, so a downgrade never repays; we set N^*_{full}=∞, which reads as commit.

Protocol. Six falsification conditions were fixed in the script header before any run. Ground truth is the cheaper of blind-follow and commit in mean cost/OPT, with OPT by exact DP,

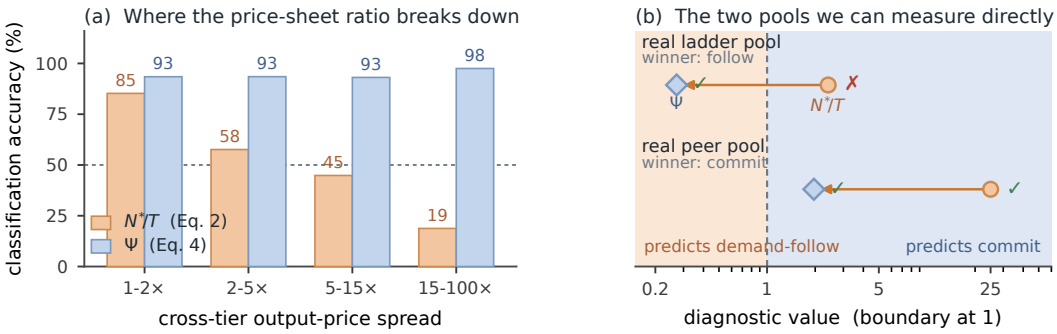

Figure 4: Restoring the output term keeps the screen accurate as output prices spread (q=0). (a) Accuracy against the DP-optimal policy by cross-tier output-price spread (n=400 sampled pools; dashed line is chance). (b) The two pools whose winner we measure directly, without retention: on the real ladder pool N^*/T=2.20 lands on the wrong side of the boundary and Ψ=0.27 on the right one (0.25 with the corrected numerator); on the peer pool both agree. At q=1 the plain screen already classifies both (Table 5). Arrows run $N^*/T \to \Psi$.

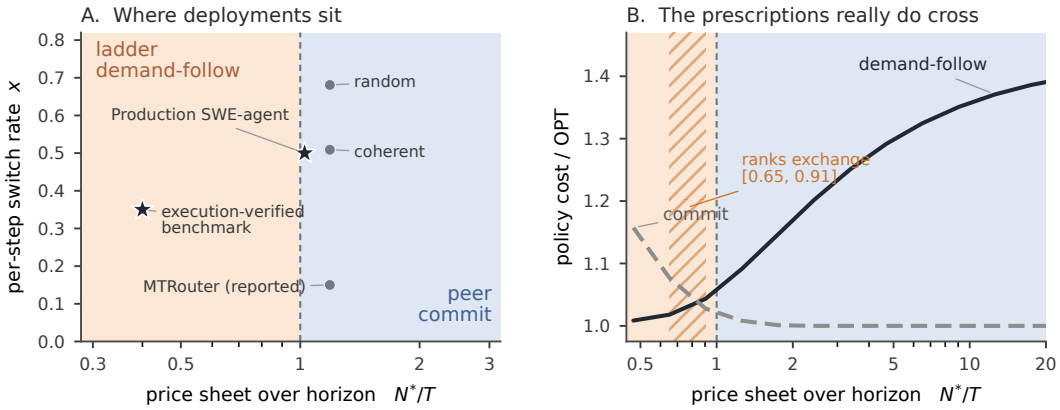

Figure 5: The phase plane without retention (q=0). (A) Price-sheet ratio against measured switch rate; production sits at 0.92. (B) Sweeping price spread at fixed output prices, the two policies exchange ranks at $N^*/T \in [0.19, 0.39]$. At q=1 the same production pool sits at 0.12 and the crossing at $N^*(1)/T \in [0.28, 0.45]$ (§5.3).

on the trajectories and four-bucket billing of §5.3. The classifiers see only the price sheet, T and ω. We report two ensembles. In ensemble A (n=400, T=6), market-realistic $\gamma/\delta \in [4, 40]$ gives $N^*/T > 1$ on every pool, so N^*/T is a constant predictor there. Ensemble B (n=250, T=13, prefix spread to 200×, coherent sessions) has $N^*/T > 1$ on 78% of pools. Because B is imbalanced we report balanced accuracy, for which a constant predictor scores 50%.

	ensemble A (T=6)		ensemble B (T=13)	
	N^*/T	Ψ	N^*/T	Ψ
balanced accuracy	50.0%	**94.6**%	63.7%	**83.3**%
sensitivity (commit)	100%	91.4%	100%	77.6%
specificity (follow)	0.0%	97.8%	27.4%	89.1%

Outcome. Ψ adds output prices at the cost of one estimated harness scalar ω. The peer pool's verdict holds for any ω up to 2.0× the estimate.

Both ensembles at full retention. Re-billing the same pools at q=1 (exact optimum per pool) changes the ground truth: commit is cheaper on 21.5% of ensemble A and 5.6% of ensemble B, against 43.5% and 19.6% at q=0. Balanced accuracy, ensemble A / B:

ground truth	$N^*(0)/T$	$N^*(1)/T$	$\Psi(0)$	$\Psi(1)$
q=0	51.3 / 63.9	55.5 / 62.7	94.3 / 82.0	63.5 / 63.3
q=1	51.0 / 62.9	69.3 / **90.0**	89.2 / 91.3	77.3 / 85.1

With the numerator of the protocol table the q=0 row of ensemble B is 63.9 / 83.3; Eq. 4 gives 82.0. The price-sheet ratio $N^*(0)/T$ is at chance on ensemble A's q=1 truth (51.0) and at 62.9 on ensemble B's. On ensemble B the output term adds about 18 points at q=0. At q=1 the retention-aware ratio reaches 90.0, and the best output-aware form reaches 91.3.

H Audit of released cost accounting

Provider microbenchmark. Eq. 1 states that a model change re-bills the accumulated prefix at the destination's cache-write rate, so the prefix cache is keyed per model. Providers report the cache buckets per call, which makes this directly measurable. Holding a fixed ≈2.7k-token prefix we issue four calls: model X cold, X again, model Y on the identical prefix, then X again. We repeat the sequence three times per direction, swapping X and Y; the table gives the two directions as forward/reverse.

	DeepSeek			OpenRouter			
step	write	read	result	write	read	result	note
1. cold	2666/2746	0	both	2665/2994	0	—	billed in full
2. repeat	106/58	2560/2688	both	2665/2994	0	—	no hit
3. switch	2745/2667	0	both	2994/2666	0	—	re-written
4. back	106/58	2560/2688	survives	2666/2994	0	evicted	design

Twelve sequences in total, six per provider, are byte-identical within each group. Row 3 is Eq. 1: a prefix warm one call earlier is billed at the destination's write rate with zero read credit, in 6/6 sequences per provider and both directions. Row 4 differs. On DeepSeek the return re-reads the origin's prefix, the retention measured at scale in §4. On the OpenRouter model the repeat call of row 2 did not hit either, so that configuration served no prompt cache. §4 therefore screens every model for cache stability before measuring retention.

The same text tokenises to 2666 / 2745 on DeepSeek (3.0% apart) and 2665 / 2994 on OpenRouter (12.4% apart), so the two models' cache keys differ. Warm reads land on multiples of 64 (2560=40×64, 2688=42×64), so even a model that never switches recovers slightly less than its full prefix.

The same test inside a growing loop. We repeat the design inside a 10-step loop whose context grows append-only. One arm holds a single model and the other switches at step 5; each starts cold with its own nonce. Steps 0–4 bill identically in the two arms (read/write 0/563, 512/615, 1024/667, 1664/591, 2176/643). At step 5 the switching arm reads 0 and writes 3462. Against Eq. 1 at the previous context (c_{t-1}=2819), the measured step-5 excess is 1.03× the prediction. Evaluating the tax at the current prompt instead would add the g_t tokens that every policy pays for.

The re-billing factor as a function of T. The loop length is our choice, so we sweep it. The argmax router's switch rate is nearly flat in T (0.52 on RouterBench, 0.22 on LLMRouterBench), so the growth below comes from the tax accumulating over a longer prefix.

T	4	8	13	20	34	44
RouterBench, undercount	1.51	2.24	**3.21**	4.62	7.41	9.32
of which tax	23%	42%	**55**%	66%	76%	79%
LLMRouterBench, undercount	1.28	1.66	**2.14**	2.85	4.29	5.22
of which tax	10%	22%	**32**%	42%	55%	60%
at full retention (q=1)						
RouterBench, undercount	1.48		**2.74**			5.46
of which tax	21%		**47**%			63%
LLMRouterBench, undercount	1.26		**1.91**			3.69
of which tax	9%		**26**%			45%

The gap is present at every T in this range, and its size grows with T. Only 4 of LLM-RouterBench's 20 models matched a live price sheet at our snapshot. One released multi-turn routing system enables provider-side prompt caching in its agent loop, while its cost tracker bills all input tokens at the flat input rate.

I End-to-end measurement: design and admissibility

Harness. SWE-bench Verified instances from a fixed repository whitelist, ordered by gold-patch length, each run inside its official container. The agent has four tools (view, search, edit, run tests); the test patch is never applied and test names are redacted from tool output, so the agent cannot read the answer. Tier demand is outcome-driven (a failed test run escalates); the two policies are blind-follow and commit with a first-step predictor of the session maximum. The step budget is 12–20 and output is capped at 900 tokens. Every call's cache-hit, cache-miss and output token counts are taken from the provider's usage field and billed at the serving tier's own prices; a hard spend cap aborts a run before it can overspend. Each trajectory's system message opens with a unique nonce, and every arm reads zero tokens at its first step.

Admissibility. A probe issues a short prefix, then a call whose context has grown by thousands of tokens (big jump, 8 trials, requiring 8/8 hits), and counts violations on consecutive same-model steps of a realistic trajectory (steady), excluding steps whose predecessor was below that deployment's minimum cacheable prefix M (Table 2). The probe prefix must itself exceed M: on Ark a 1,400-token prefix gives 6/8 and 5/8, a 3,000-token prefix 8/8 on both models. Independence is checked by sending a fresh prefix to one model and the same prefix to the other.

model	platform	big jump	steady	verdict
deepseek-chat, deepseek-reasoner	DeepSeek	—	0/40	admissible
thinkingmachines/inkling	OpenRouter	8/8	0/11	admissible
z-ai/glm-5-turbo	OpenRouter	8/8	—	admissible
z-ai/glm-4.7	OpenRouter	8/8	0/9	admissible
zai-org/GLM-5.2	SiliconFlow	8/8	0/9	admissible
deepseek-flash	DeepSeek	8/8	0/74	admissible
deepseek-v4-pro	DeepSeek	8/8	0/71	admissible
deepseek-v4-flash	Bailian	8/8	0/74	admissible
deepseek-v4-pro	Bailian	8/8	0/69	admissible
qwen3.8-flash	Bailian	8/8	0/74	admissible
qwen3.8-max	Bailian	8/8	0/71	admissible
deepseek-v4-flash-ga	Ark	8/8	0/68	admissible
deepseek-v4-pro-ga	Ark	8/8	0/58	admissible
deepseek-ai/DeepSeek-V4-Pro	SiliconFlow	6/8	1/9	rejected
openai/gpt-4.1-mini	OpenRouter	5/8	1/9	rejected
moonshotai/Kimi-K2.7-Code	SiliconFlow	4/8	1/9	rejected
qwen/qwen3-max	OpenRouter	4/8	1/9	rejected
qwen/qwen3-coder-flash	OpenRouter	—	36%	rejected
doubao-seed-2.1-turbo	Ark	4/8	—	rejected
doubao-seed-2.1-pro	Ark	4/8	—	rejected

For the DeepSeek and Bailian replays a big-jump trial starts from a 1.1–4.7k-token prefix, and independence holds in 3/3 trials per model. DeepSeek models are clean on DeepSeek's API, not measurable on SiliconFlow, and measurable on Bailian. Many exactly price-flat census pairs are aliases, such as a model and its "-latest" pointer; on the one we tested, a fresh prefix sent to the first model hit on the second.

Replayed schedules. Of the 36 DeepSeek trajectories in the teacher-forced runs, 29 contain a return within 12 steps. For each we keep the model sequence and grow an append-only context by the recorded per-step prompt increments (at least 60 tokens), filled with source files from the instance's repository. The assistant message is a fixed line, output is capped at 16 tokens, and thinking is off on Bailian and Ark. The cheap and strong tiers map to

deepseek-flash / deepseek-v4-flash / qwen3.8-flash / deepseek-v4-flash-ga and deepseek-v4-pro / qwen3.8-max / deepseek-v4-pro-ga. Replayed prompts have a median of 0.90–0.93 of the recorded lengths. Prices are list prices on 14 September 2026 in CNY per 10^6 tokens (input / cache hit / output): DeepSeek flash 2/0.04/8 and v4-pro 9/0.3/27 at peak, with off-peak prices half on all three rates; Bailian v4-flash 1/0.2/2, v4-pro 12/1/24, Qwen3.8 Flash 0.8/0.1/2.7 and Max 12/1.5/36; Ark v4-flash 3/0.1/9 and v4-pro 9/0.3/27 (16 September 2026), the same strong-tier sheet as the vendor's own API.

For DeepSeek, Bailian DeepSeek, Bailian Qwen and Ark in that order, the trajectory-bootstrap 95% intervals of the overstatement are $[2.4, 3.1]$, $[1.9, 2.5]$, $[1.9, 2.6]$ and $[2.0, 2.6]$, and those of the $q{=}0/q{=}1$ error ratio are $[6.7, 15.0]$, $[1.1, 2.1]$, $[1.2, 2.3]$ and $[1.0, 2.3]$. On the 85 returns per deployment whose stale prefix has at least 1,024 tokens the hit ratio is 0.98, 0.89, 0.89 and 0.74. Reads are multiples of 128 tokens on the first three deployments and of 256 on Ark. On DeepSeek a return reads at most 128 tokens less than the stale prefix; on Bailian the shortfall has a median of 350 tokens and stays below 1,024; on Ark returns below 2,048 tokens read nothing at all and the rest read the whole prefix. In the two-fold fit the per-fold $\hat{q}$ is 0.97/0.98, 0.78/0.83, 0.74/0.84 and 0.82/0.91. The out-of-sample error at $q{=}0$, $q{=}1$ and the fitted q is 0.516/0.051/0.037, 0.403/0.268/0.197, 0.376/0.233/0.182 and 0.480/0.335/0.281.

Block-level retention. A scalar q ignores cache granularity. We also fit reads rounded down to blocks of $B \in \{1, 64, 128, 256, 512, 1024\}$ tokens, with a minimum prefix $M \in \{0, 512, 1024, 2048\}$ below which holds and returns read nothing; a return reads $q\,B\lfloor \min(c_{\text{seen}}, c_{t-1})/B \rfloor$. Parameters are fitted on half of each deployment's trajectories and scored on the other half. The mean $|\log|$ bill error with a scalar q against the block model is 0.015/0.017 on DeepSeek chat→reasoner, 0.106/0.082 on OpenRouter, 0.034/0.004 on DeepSeek flash→v4-pro, 0.193/0.007 and 0.175/0.008 on the two Bailian pools, and 0.264/0.030 and 0.253/0.035 on the two Ark folds. The loss-minimising block model has $q{=}1$ on all four replayed deployments, with $B{=}128$ on DeepSeek's API, $B{=}1{,}024$ on Bailian, and $B{=}512$ with $M{=}2{,}048$ on Ark, the only deployment where the minimum binds. Table 2 uses the documented block where it matches the reads. On these deployments a return loses only what block rounding and the minimum lose.

Contexts that are not append-only. No step of the 295 trajectories of §4 shrinks the context, so we construct the violation. The context is a list of blocks, one user message each: compact replaces the middle with a 200-token summary every four steps, and rotate swaps the block in position 1 at every step. We replay 13–15 schedules per mode and policy on DeepSeek. Per trajectory the shared prefix predicts the bill to a mean $|\log|$ of 0.021–0.064 (87–100% within ±10%); the previous prompt gives 0.37–1.53 (0–7%). Per switch, all 57 switches with an empty shared prefix read back nothing, and where it is non-empty the read differs from $B\lfloor \text{shared}/B \rfloor$ by a median of 0 tokens and a mean of −30, the deviations sitting on prefixes of a few hundred tokens. On deepseek-flash↔Bailian qwen3.8-flash, where $\Psi(1)$ is 3.3–13.3 and the diagnostic calls commit, commit is cheaper at every T from 3 to 12 under compaction (0.63–0.88×, $p \le 0.005$ for $T \ge 5$), as in the append-only replay (0.73–0.83×). Switching tax is 2.2% of the bill append-only, 3.4% under compaction and 0.4% under rotation, where the prefix is invalidated every step in any case.

Reading M and B off a deployment. M is bracketed by bisecting the stale prefix length. On the replays the largest c_{seen} that reads nothing and the smallest that reads something bracket the value we use on three of the four deployments: DeepSeek never misses, so $M{=}0$; Bailian Qwen gives $[991, 1136]$ around 1,024; Ark gives $[2031, 2134]$ around 2,048. Three to seven calls suffice at block resolution. Bailian DeepSeek has no sharp threshold: prefixes of 563 tokens hit while others of 947 do not, and the hit rate over 512–1,024 is 0.61, so a single M is an approximation there. B is not identifiable from a handful of reads: every read is a multiple of it, so the greatest common divisor of a few reads is itself a multiple of B and overshoots (128, 512 and 2,048 from three reads on DeepSeek, Bailian DeepSeek and Ark). Over all reads the shared granularity is 128, 512, 128 and 256 on the four deployments. The (B, M) pairs of Table 2 are therefore the values that predict held-out bills best, not a claim about the provider's internal block size.

Block structure moves the bill, not the prescription. Weighting each return by c_{seen}, the effective retention $B\lfloor c_{\text{seen}}/B\rfloor \mathbf{1}\{c_{\text{seen}} \geq M\}/c_{\text{seen}}$ is 0.99, 0.88, 0.89 and 0.89 on the four deployments. Substituting it for q=1 moves Ψ by up to 0.3 but leaves every deployment in the same band segment at T=6 and T=13 (8 of 8). Block rounding and the minimum prefix matter for predicting the bill and not for choosing the policy, because the band is wide next to the shift they cause.

Where the parameters come from. Fitting (q, B) to the replayed bills on DeepSeek selects B=128 and predicts those bills to $|\log|$=0.004, but predicts the free-running bills of Fig. 6 to 0.070 (70% within ±10%). The documented B=64 gives 0.028 in distribution and 0.048 out of it (85%). We therefore read B and M off the provider and the reads rather than fitting them to bills.

Tax-greedy inside the band on live APIs. We replay the 29 switch schedules on three pools whose $\Psi(1)$ falls inside $[0.2, 3]$, under demand-follow, commit and tax-greedy, sweeping T by truncation. On deepseek-flash→deepseek-v4-pro demand-follow is cheaper at every T (commit costs 1.05–1.18×); on deepseek-flash→ Bailian deepseek-v4-flash and on Bailian qwen3.8-flash→deepseek-v4-flash commit is cheaper (0.78–0.85× with paired Wilcoxon $p \leq 0.003$ at every T, and 0.92–0.96× with $p < 0.05$ at eight of the ten T). Tax-greedy costs 0.987–1.009, 0.996–1.004 and 0.994–1.051 times the cheaper fixed policy over $T \in [3, 12]$. The three runs billed CNY 9.2 in total at list prices.

Where the retained prefix falls short. All 412 returns on the two pools read a multiple of 64 tokens. On DeepSeek the hit ratio is 0.968 for stale prefixes of 310–1,734 tokens and 1.000–1.007 for longer ones (rank correlation with prefix length +0.59). On OpenRouter it is 0.73 and 0.83 in the two shortest quartiles (below 1,443 tokens) and 0.98 above 12k. Short prefixes lose a larger share to block rounding.

Teacher-forced replay. Two free-running arms diverge: on one instance their contexts ended at 39,021 and 3,639 tokens. We therefore record one canonical trajectory and replay its message sequence under the other policy, issuing real calls but appending the recorded assistant message, so the two arms' contexts are identical up to the nonce. This cuts the log-ratio standard deviation from 0.798 to 0.227, and one instance in which both arms chose the same model at every step billed a ratio of 1.011, bounding measurement precision near 1%. Because neither policy reads T and cost is causal, the first k steps of a 12-step replay bill exactly as a k-step run.

On the DeepSeek pool (20 instances; all 90 returns in the canonical arms hit) blind-follow is cheaper than commit at every horizon from T=3 to 12 (commit 1.17–1.37×), significantly so for every $T \geq 6$ ($p \leq 0.005$, paired Wilcoxon), as $N^*(1)/T < 1$ predicts at each of these horizons.

Quality when the policy drives the trajectory. The replays above hold the trajectory fixed, which isolates billing but says nothing about the work. We also let three policies drive 28 SWE-bench instances free-running on deepseek-flash→deepseek-v4-pro, choosing the tier from the previous step's outcome rather than from a label, and scored them with the official harness (Fig. 6). Demand-follow resolved 10 instances, tax-greedy 9 and commit 5; commit against demand-follow is not significant (McNemar 1:6, p=0.12). Mean bills per instance fall within 3% of each other and are not separable either (0.97×, p=0.26): once each policy drives its own trajectory, the spread between trajectories swamps the billing difference the replays isolate. Switching tax is 35% of the bill under demand-follow, 25% under tax-greedy and 7% under commit.

Market consequence. On the 14,370 census pairs with admissible prices, taking q=0.95 and demand sequences from real trajectories, the full-rebuild and retention rules prescribe different deployments on 17.8% of pairs; the full-rebuild prescription is the costlier one on 25.4% of pairs against 10.8% for the retention rule.

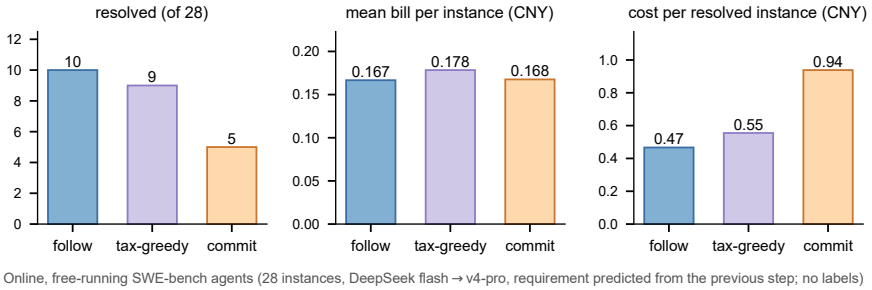

Figure 6: Free-running on 28 SWE-bench instances, three policies, requirement predicted from the previous step and no labels. Resolved counts, mean bill per instance and bill per resolved instance. The third panel divides by the resolved count, so it moves sharply with one instance; we report it for completeness and draw no cost conclusion from these runs.

J System, router audit, and wrapper

Held-out check on the band. The band $[0.2, 3]$ is the envelope of four flip points: price sweeps at q=0 and q=1, a same-corpus price sweep, and a T sweep. Applied to the 250-pool ensemble, which played no part in setting it, the rule outside the band is right on all 206 pools outside it at q=1 and all 104 at q=0 (exact binomial 95% lower bounds 98.2% and 96.5%). The tightest band that makes no error on a disjoint 400-pool ensemble is $[0.44, 0.71]$ at q=1 and $[0.58, 0.94]$ at q=0; on the held-out pools it scores 98.4% and 88.8%. $[0.2, 3]$ is wider, and its width is what carries it to new pools.

The whole rule against the optimum. On the same 250 held-out pools we bill the full prescription (commit above the band, blind-follow below it, tax-greedy inside) against the exact optimum under each semantics. Table 4 gives the median and 90th percentile of cost/OPT. Outside the band the rule never picks the costlier fixed policy. The rule is at or below the better fixed policy in hindsight on 91% of pools at q=0 and 94% at q=1; its excess over OPT is concentrated inside the band.

Table 4: Cost/OPT on 250 held-out pools (T=13), median / 90th percentile.

	q=0	q=1
rule (whole prescription)	1.116 / 1.335	1.011 / 1.087
blind-follow	1.162 / 1.522	1.011 / 1.112
commit	1.610 / 2.547	1.774 / 2.828
tax-greedy everywhere	1.121 / 1.352	1.013 / 1.095
better fixed policy (hindsight)	1.123 / 1.357	1.011 / 1.081
rule, below band	1.012 / 1.072 (n=90)	1.005 / 1.043 (n=204)
rule, inside band	1.216 / 1.377 (n=146)	1.101 / 1.182 (n=44)
rule, above band	1.131 / 1.220 (n=14)	1.023 / 1.025 (n=2)

Which pairs the median is taken over. Table 7 uses the all-pair median N^*, while a deployment traverses only the pairs its router visits. On the same 178-model sheet we recompute the median under seven price-side conventions. The census has no comparable quality scores, so the conventions use prices only. Sampling four-model pools from the same sheet, the pool median N^* is below 20 steps in 97% of cost ladders at q=0 and 100% at q=1; for four frontier peers the shares are 45% and 99%. At T=13 and q=1, 62% of census pairs lie below the band of §5.1 and 3.5% above it.

convention	pairs	median N^*	N^*/T at T=13
all orderable pairs with $N^* > 0$	15,209	11.84	0.91
same provider	1,455	11.71	0.90
cross provider	13,754	11.85	0.91
adjacent price ranks	3,038	15.50	**1.19**
frontier peers (top quartile)	1,052	19.25	**1.48**
top quartile → bottom quartile	2,450	10.71	0.82
above-median pairs	3,615	18.51	**1.42**

The pair the deployed router of §5.2 traverses gives $N^* = 12.00$, within 1.3% of the all-pair median, and the same- and cross-provider splits agree to 1%. Across conventions the median spans 1.8×, and at T=13 it ranges over 0.82–1.48. The further a downgrade reaches, the faster it repays: pools of adjacent tiers sit on the peer side, and pools that drop from the top to the bottom quartile sit on the ladder side. For a given router, N^* is computed over the pairs that router selects between.

Table 5: Blind-follow and commit on 77 real multi-step trajectories (median T=6, 500 output tokens per step), mean cost/OPT. Requirements ℓ_t are execution-verified per-step labels (App. J).

pool	q	$N^*(q)/T$	blind-follow	commit	cheaper
ladder (execution-verified tiers)	0	2.20	1.038	1.287	follow
	1	0.35	1.009	1.377	follow
peer (flagship prices)	0	23.0	1.456	1.162	commit
	1	10.6	1.337	1.171	commit

A wider policy comparison under one protocol. The eight policies are: blind-follow, which follows the demanded tier; commit, which never drops once raised; threshold, which makes one sustained downgrade while more than $N^*(q)$ steps remain; hysteresis(k) for $k \in \{2, 3\}$, which drops only after k consecutive lower-demand steps; tax-greedy, which compares staying with dropping at each step using q; segment(4), which re-decides every four steps from the previous window; and lookahead(4), which fixes each four-step segment to its future maximum demand and so uses future information.



Figure 7: Policy cost/OPT on the 21 trajectories with $T \geq 9$, under both billing semantics on both pools; the cheapest policy in each column is outlined. Commit and threshold are optimal on the peer pool, tax-greedy and blind-follow on the ladder pool, and hysteresis is dominated everywhere. Lookahead(4) uses future demands. The full numbers, including all 77 trajectories, are in the table below.

The eight policies of §8, on all 77 trajectories and on the 21 with $T \geq 9$, under both billing semantics, with 400 output tokens per step (so the all-trajectory columns differ slightly from Table 5). Hyperparameters are fixed across pools and semantics: hysteresis $k \in \{2, 3\}$, segment and lookahead length 4; threshold uses the $N^*(q)$ of the billing semantics.

	peer pool				ladder pool			
	q=0		q=1		q=0		q=1	
policy	all	$T \geq 9$	all	$T \geq 9$	all	$T \geq 9$	all	$T \geq 9$
commit	1.154	1.012	1.161	1.027	1.263	1.418	1.350	1.584
threshold	1.154	1.012	1.161	1.027	1.263	1.418	1.222	1.377
tax-greedy	1.234	1.058	1.233	1.058	1.042	1.085	1.011	1.003
segment(4)	1.226	1.050	1.233	1.066	1.148	1.229	1.227	1.377
hysteresis(2)	1.311	1.211	1.285	1.151	1.107	1.150	1.148	1.224
hysteresis(3)	1.252	1.118	1.249	1.101	1.142	1.226	1.209	1.338
lookahead(4)*	1.223	1.111	1.231	1.127	1.176	1.238	1.258	1.387
blind-follow	1.494	1.542	1.354	1.296	1.051	1.085	1.011	1.003

*uses future demands.

On all 77 trajectories the ordering matches the $T \geq 9$ subset: commit and threshold lead on the peer pool, and tax-greedy and blind-follow lead on the ladder pool. The segment policies behave like holding the running maximum, because requirements in these trajectories are close to monotone.

Block rounding and cache expiry. We repeat the $T \geq 9$ comparison at q=1 with cache reads rounded down to 64-token blocks and with caches expiring after k idle steps (at 30–60 s per step, a 300 s TTL corresponds to $k \approx 5$–10). The exact optimum tracks each model's last use step. Every ranking is unchanged. Billing instead at $q \in \{0.8, 0.85, 0.9, 0.94\}$, the range of fitted retention in §4, every $T \geq 9$ cost stays within 0.07 of the q=1 column and the cheapest policy on each pool is unchanged. At q=0.8 segment(4) and tax-greedy trade third place on the peer pool (1.055 and 1.056). Under the block model fitted to Bailian (q=1, 1,024-token blocks and minimum prefix), tax-greedy remains cheapest on the ladder pool (1.002) and commit on the peer pool (1.0446, against 1.0451 for tax-greedy). The largest cost change from the q=1 column is 0.077, for commit on the ladder pool. With 128-token blocks at q=1, and with 128- or 1,024-token blocks at q=0.81, the cheapest policies are also unchanged.

	peer pool			ladder pool		
billing	commit	tax-greedy	blind-follow	blind-follow	tax-greedy	commit
q=1	1.027	1.058	1.296	1.003	1.003	1.584
+ 64-token blocks	1.028	1.057	1.292	1.003	1.003	1.589
+ expiry after 10 steps	1.028	1.057	1.292	1.003	1.003	1.589
+ expiry after 5 steps	1.028	1.057	1.318	1.004	1.003	1.588
+ expiry after 3 steps	1.019	1.051	1.365	1.016	1.015	1.564

Encoder choice. Tail-of-prefix MiniLM (384-d, last 2k characters): tier accuracy 73.4%, under-prediction 7.5%. Full-prefix gte-Qwen2-1.5B (1536-d, 8k-token window, GPU): accuracy 71.8%, under-prediction 17.5%. The difficulty signal of a step sits at the end of the prefix, and whole-context mean-pooling dilutes it. The sentence encoder is frozen; a downstream classifier is trained on its embeddings with instance-level 5-fold splits.

Measuring the under-prediction rate η. η is the rate at which a t=1 predictor under-shoots the session maximum $L^* = \max_t \ell_t$. On all 520 static-track trajectories with instance-level 5-fold splits: a naive predictor (commit to ℓ_1) gives $\eta = 0.085$; the trained MiniLM$\to L^*$ classifier with argmax decisions gives $\eta = 0.048 \pm 0.016$; the cost-sensitive rule (round up to the smallest tier whose cumulative probability reaches $1 - \text{risk}$, risk $= 0.15$) gives $\eta = 0.013 \pm 0.010$.

Two corpora. 439 of the 520 are single-step, where a first-step predictor cannot under-shoot. On the 81 multi-step trajectories the naive η is 0.54, the figure quoted in §6. The identity of Prop. 1 is exact on both corpora ($|1 + \eta\bar{\kappa} - \text{measured}| < 10^{-15}$), with $\bar{\kappa}$ the mean forced-rebuild tax of an under-shot session divided by the mean per-session OPT over all sessions, the $\bar{\kappa}$ of Prop. 1.

Concentration of L^*. 444/520 trajectories have L^* at the lowest tier, but 439 of them are single-step, where $\ell_1 = L^*$. Among multi-step trajectories the lowest tier holds 39.5% and the entropy of L^* rises from 0.79 to 1.47 bits. On the longest trajectories ($T \geq 6$) L^* sits at the highest tier in 65.5% of cases and at the lowest in 29.1%. Commit is cheap at both ends of the length distribution: short sessions rarely escalate and long ones start at the ceiling. The middle stratum costs most ($T \in [3, 5]$, entropy 1.55 bits, commit/OPT = 1.136).

Resampling multi-step trajectories to a uniform L^* raises η from 0.54 to 0.69 and commit/OPT from 1.078 to 1.226. Since $\eta \leq 1$, the identity gives commit/OPT $\leq 1 + \bar{\kappa}$, which is 1.33 on this workload.

Requirement labels. Every policy table uses per-step requirement labels fixed before any policy runs. On the 77 TwinRouterBench trajectories ℓ_t is the benchmark's execution-verified tier label, the cheapest of four tiers whose model solves that step; on constructed sessions it is the cheapest of four tiers answering the query correctly, with queries no model answers excluded. For example, SWE-bench django-10914 has $\ell_{1:8} = (2, 1, 2, 3, 3, 0, 0, 0)$ with context growing from 1.4k to 4.4k tokens: commit holds tier 3 for all eight steps, blind-follow switches four times, and OPT is computed on exactly this sequence.

Audit settings. The router audits use four settings, listed below with the setting named alongside each number.

Table 6: Router audit settings. The rows differ in prices, context growth and anchor. The main text quotes the last two rows.

Setting	Corpus	Prices	Ctx. growth	Anchor	Range
Uniform testbed	20 open models	flat	fixed	modal	1.26–4.38
Flagship, naive	11 models, 12k q.	per-model	model-dep.	modal	1.00–4.77
Flagship, ctrl.	11 models, 12k q.	per-model	pool-mean	modal	1.00–2.98
↪ pure tax	”	”	”	warm-cache	1.00–2.29

Two confounds separate the rows. Growing context by the chosen model's own token counts charges a policy for picking verbose models; holding growth at the pool average removes this. Comparing against the session's modal model charges a policy for picking dearer models; the warm-cache counterfactual below removes this too. Each removal lowers the ratio.

Decomposing the audited ratio. A router's bill relative to a never-switch anchor mixes two effects: the switching tax proper, and the fact that its chosen models may be intrinsically dearer than the anchor. We separate them with a third bill in which every step is charged as if that model's cache were warm, giving $B_{\text{actual}}/B_{\text{modal}} = (B_{\text{actual}}/B_{\text{warm}}) \times (B_{\text{warm}}/B_{\text{modal}})$. On a price-flat pool the second factor is 1; on the real flagship sheet it ranges over 0.95–1.30. The pure-tax factor ranges over 1.00–2.29 (median 1.68). The two orderings agree at Spearman 0.72: the cost-oriented pair stays on top, but GraphRouter falls from second to sixth once its model mix is removed.

The imitation baseline, in full. Five seeds per arm on the price-flat peer sheet. Commit reaches $1.113 \pm 0.032 \times$ OPT at $0.20 \pm 0.25\%$ step-level violations; imitation trained on hindsight-optimal labels and constrained by the same quality floor reaches 1.133 ± 0.058 at $1.43 \pm 1.56\%$. Two unconstrained variants reach 0.99 ± 0.02 and 1.03 ± 0.04 and violate the tier requirement on 5.3% and 6.8% of steps (2 and 3 seeds). Blind demand-following is 1.262 ± 0.047 at 2.8%.

Varying T directly. Table 7 divides a measured N^* by cited values of T; here we vary T itself. Sessions are built by the coherent construction of §5.2 from the 9,634 queries of

Table 7: Trajectory lengths reported for published agent harnesses, on the production pool of §5.2 ($N^*(0)$=12.0, $N^*(1)$=1.5). †turn counts.

harness	reported quantity	T	$N^*(0)/T$	$N^*(1)/T$
Production SWE-agent (ours)	median turns†	13.0	**0.92**	0.115
SWE-agent + GPT-4 (Yang et al., 2024)	mean turns†	14.7	0.82	0.102
SWE-agent-LM-32B (Yang et al., 2025)	mean steps	24.9	0.48	0.060
SWE-agent + Claude 3.7 (Yang et al., 2025)	mean steps	29.1	0.41	0.052
OpenHands, thinking (Ahmad et al., 2026)	mean turns†	58.2	0.21	0.026
OpenHands, no thinking (Ahmad et al., 2026)	mean turns†	94.1	0.13	0.016

the 11-model corpus that some model answers, with ℓ the cheapest tier that answers each. Holding one pool fixed and sweeping $T \in [4, 44]$, commit is cheaper at large N^*/T and a single sustained downgrade (the policy of Thm. 1) is cheaper below; they separate at N^*/T=1.27. On a pool whose output prices spread 60×, the ranking does not change at any T.

Under retention, and the band. The exact optimum at $q > 0$ becomes expensive on long constructed sessions, so we also compare the two policies' bills directly, which needs no normaliser (600 sessions per T, fewer where the corpus runs out). At q=1 blind-follow's total bill is 0.92× commit's at T=4 falling to 0.64× at T=44, and it is cheaper on 60–93% of sessions for every $T \geq 13$; on the wide-output pool it is 0.41–0.59×. At q=0 the total-bill comparison exchanges rank near $N^*/T \approx 3$, the optimum-weighted one at 1.27, and price sweeps at 0.2–0.45: together these define the band $[0.2, 3]$ of §5.1. The q=1 verdict is the same under either weighting.

Sweeping prices at fixed T on the same corpus puts the exchange at N^*/T=0.40, a factor of 3.2 from the T sweep; §5.1 therefore reads the ratio with a band. Holding the L^* distribution fixed leaves the gap (1.265 → 1.238). It comes from the constant-context assumption of Eq. 2.

Pool construction with the discount. §3 identifies discount heterogeneity as a lever on N^*. From the 178-model census we form every ordered pair whose price ratio matches a target r and whose expensive tier lies in a fixed price band ($1–$3/Mtok), so that (δ, γ) is the only free variable. At r=2, N^*/T ranges from 86 down to 0.18 within this set.

Remaining material. Per-router audit tables, the wrapper's per-family recovery rates, the imitation-learning baseline's setup, and the full operating-point sweeps accompany the SwitchTax source in the code release.

K Gate sensitivity: the full tax-rate sweep

Table 8: $\theta^\star/\hat{\sigma}$ across tax rates, with $\hat{\sigma}$ from the default tie-std estimator. The law holds for $\lambda_{\text{tax}} \in [0.06, 0.15]$; at 0.02 the bracket Λ falls below 1, outside the proposition's regime.

λ_{tax}	$\theta^\star/\hat{\sigma}$	CV	Spearman
0.02 (bracket < 1)	0.318	0.95	0.10
0.04	0.737	0.28	0.40
0.06	0.997	0.17	1.00
0.10	1.237	0.22	0.90
0.15	1.791	0.26	0.90
0.25	2.461	0.40	0.70

L Gate regression and leverage

Proposition 4 (Optimal gate scales with the router's noise). At step t let $G_t \geq 0$ be the true quality gain of switching to the router's preferred model, observed as $\hat{G}_t = G_t + \varepsilon_t$ with $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ independent of G_t; the gate switches iff $\hat{G}_t > \theta$. Let $\bar{d}$ be the mean

tax (Eq. 1), $\bar{x}$ the rate of zero-gain steps, V the value of a unit of quality, $\bar{G}f(\theta)$ the gain-weighted density of $\hat{G}$ on genuine-gain steps, and $\Lambda = \bar{d}\bar{x}/(V\bar{G})$. If $\sqrt{2\pi}\,\sigma f(\theta)$ stays in a fixed interval $[s_-, s_+] \subset (0, \infty)$ near the optimum, every interior minimiser of the expected error cost satisfies $\sqrt{2\ln(\Lambda/s_+)} \le \theta^\star/\sigma \le \sqrt{2\ln(\Lambda/s_-)}$ whenever $\Lambda > s_+$, and hence Eq. 5.

Leverage of one router. GraphRouter spans 91% of the $\hat{\sigma}$ range. Dropping it leaves four routers over a 1.8× range: the Pearson correlation falls to 0.938 (p=0.06) while Spearman stays at 1, and the in-sample $\theta^\star/\hat{\sigma}$ tightens from 0.997 (CV 0.17) to 0.918 (CV 0.08). Selected on held-out halves, the four give 0.96 (CV 0.08). The correlation rests on that router's leverage; the proportionality constant does not.

Table 9: $\theta^\star \propto \hat{\sigma}$ across published routers (in-sample).

	Avengers	Avengers-Pro	EmbedLLM	GraphRouter	IRT-Router	mean / CV
$\hat{\sigma}$	0.0416	0.0325	0.0501	0.3236	0.0580	
$\theta^\star$	0.0342	0.0326	0.0432	0.4254	0.0572	
ratio	0.82	1.00	0.86	1.31	0.99	**1.00** / 0.17

Table 10: Sensitivity of $\theta^\star \propto \hat{\sigma}$ to the noise estimator (default tax rate); the tax-rate sweep follows.

	$\theta^\star/\hat{\sigma}$	CV	Spearman
tie-std (default)	0.997	0.17	1.00
tie-MAD	1.326	0.25	1.00
tie-IQR/2	1.055	0.15	1.00

Table 11: Gate outcomes per router on 20-step sessions (λ_{tax}=0.06): accuracy and switch rate without a gate, with the in-sample optimal gate, and without any switch, and the held-out $\theta^\star/\hat{\sigma}$.

	no gate		gate at $\theta^\star$		never switch	held-out
router	acc.	switch rate	acc.	switch rate	acc.	$\theta^\star/\hat{\sigma}$
Avengers	0.619	0.441	0.614	0.161	0.594	0.89
Avengers-Pro	0.527	0.403	0.538	0.038	0.516	0.98
EmbedLLM	0.630	0.628	0.621	0.254	0.580	1.08
GraphRouter	0.627	0.447	0.618	0.107	0.587	1.37
IRT-Router	0.635	0.672	0.621	0.232	0.579	0.87

The two sensitivity sweeps. The three tie-based estimators of $\hat{\sigma}$ all give Spearman 1.00 and CV 0.15–0.25. Across tax rates the law holds for $\lambda_{\text{tax}} \in [0.06, 0.15]$ (Spearman ≥ 0.90); at λ_{tax}=0.02 the bracket Λ falls below 1, outside the proposition's regime. The coefficient grows with λ_{tax} (0.997 → 2.461), the direction $\sqrt{2\ln\Lambda}$ predicts.

A through-origin fit $\theta^\star = k\hat{\sigma}$ over the five admissible routers gives $R^2 = 0.989$, and the Pearson correlation is $r = 1.000$. GraphRouter's $\hat{\sigma}$ is 6.5× the median and spans 91% of the observed range, and without it $r = 0.938$. The statistics we use instead are $\theta^\star/\hat{\sigma} = 1.00 \pm 0.17$, Spearman 1.00 ($p \approx 0.008$), and the curves of Fig. 8.

Transfer across routers. For each router we set $\theta = a\hat{\sigma}$, with a the mean $\theta^\star/\hat{\sigma}$ of the other four (0.92–1.04). On all sessions this recovers 95–100% of the objective gain of gating at the router's own $\theta^\star$, and 81% on GraphRouter; the objective is at most 0.0021 below its optimum. Session-bootstrap 95% intervals of $\theta^\star/\hat{\sigma}$ are $[0.66, 1.67]$ (Avengers), $[0.70, 1.00]$ (Avengers-Pro), $[0.69, 1.69]$ (EmbedLLM), $[0.67, 1.47]$ (GraphRouter) and $[0.45, 1.41]$ (IRT-Router). Without rescaling, gating at $\theta^\star$ raises the objective over no gate by 0.011–0.033, and the transferred threshold by 0.009–0.033; for every router the session-bootstrap 95% interval of both gains excludes zero.

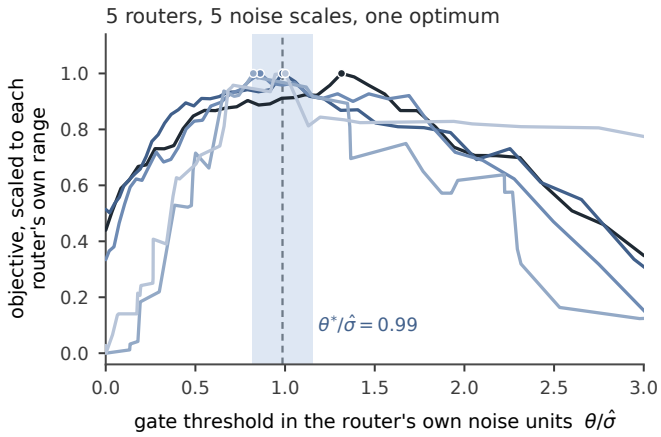

Figure 8: Objective curves of the five admissible routers against the gate threshold in units of each router's prediction noise. The peaks sit at $\theta/\hat{\sigma} \approx 1$ (markers; median 0.99). Each curve is scaled to its own range; the five objective levels differ by up to 0.07.

M Proofs

Standing assumption (monotone growth). Throughout this appendix, within a downgrade window the per-step excess premium π_t and the rebuild price B_t are non-decreasing in t. Both are proportional to the accumulated context c_{t-1}, so the assumption is that context grows monotonically inside a window. Append-only agent loops satisfy it; prefix replacement is treated in App. E.

M.1 Retention: what changes, proof by proof

Under Eq. 1 a switch to m' at step t re-reads $w = q\,\min(c_{\text{seen}}(m'), c_{t-1})$ at $p^{\text{cr}}_{m'}$ and writes the rest; under the standing assumption $c_{\text{seen}}(m') \le c_{t-1}$. Every argument below at $q > 0$ rests on two facts.

(R1) On a price-flat pool a switch is never cheaper than holding. The switch costs $p^{\text{cr}}w + p^{\text{cw}}(c_t - w)$ against the hold's $p^{\text{cr}}c_{t-1} + p^{\text{cw}}(c_t - c_{t-1})$; the difference is $(p^{\text{cw}} - p^{\text{cr}})(c_{t-1} - w) \ge 0$.

(R2) Rebuild prices stay monotone inside a window. While a tier m is not played its retained prefix w_m is fixed, so its rebuild price $p^{\text{cr}}_m w_m + p^{\text{cw}}_m(c_t - w_m) = p^{\text{cw}}_m c_t - (p^{\text{cw}}_m - p^{\text{cr}}_m)w_m$ is an affine, non-decreasing function of c_t, and so is any minimum over such tiers.

Theorem 1. The proof below is written for every q: retention enters only through the retained prefixes in D_t and U_e, which stay fixed while the corresponding tier is not played, i.e. (R2). □

Theorem 2. Replace the q=0 step difference in the proof below by (R1); the rest is unchanged. □

Proposition 1. Commit holds the running maximum of its prediction and the requirement, so every tier change moves to a tier it has never played: $w = 0$ and the forced tax is $(p^{\text{cw}} - p^{\text{cr}})c_{t-1}$ at every q. By Thm. 2 OPT makes no switch. Both sides of the identity, and $\eta, \bar{\kappa}$, are therefore independent of q. In the second claim, replace N^* by $N^*(q)$: the down-then-up pair now charges only the fraction $1 - q$ of the return leg's rebuild. □

Proposition 2. The construction uses only that a round trip costs $B + U > 0$ against a premium π per step. Under retention the return rebuild becomes $U_q \le U$; the two branches meet at $\pi(\tau{+}1) = B + U_q$ with ratio $2 - \pi/(B + U_q)$, and $o(1) \to 0$ as the round-trip horizon $(B + U_q)/\pi$ grows. The randomised bound follows by the same Yao argument. □

Theorem 3. A Phase-2 switch at round t to an arm last played at $t_{\text{seen}} < t$ costs $\lambda(t - q\,t_{\text{seen}}) \ge (1 - q)\lambda t \ge \Lambda_q := (1 - q)\lambda T/2$. Step (i) of the lower bound holds with Λ_q in place

of Λ, and step (ii) applies once $\Lambda_q \geq 1$, i.e. $T \geq 2/((1-q)\lambda)$, giving $\tilde{\Omega}(((1-q)\lambda)^{1/3}T)$. At q=1, $\Lambda_q = 0$ and the reduction says nothing. The upper bound never used the arm's history: a switch at round $b\tau$ costs at most $\lambda b\tau$ for every q, so the batched-EXP3 bound holds verbatim. □

Proposition 3. Exploration switches still cost at most λt each, so ETC's bound holds for every q. The commitment-horizon and stationary-cutoff claims use that a switch's tax depends only on the round index, which is exact only at q=0; we state them there. The simulations of this appendix and App. O are single-slot (q=0) by construction.

M.2 Theorem 1: growing-cost ski rental, and its K-independence

Accounting and benchmark. Costs are measured as an excess over the floor $F = \sum_t [p^{\mathrm{cr}}_{\ell_t} c_{t-1} + p^{\mathrm{cw}}_{\ell_t} g_t + p^{\mathrm{out}}_{\ell_t} o_t]$, the cost of running warm at the requirement. Prices are non-decreasing in tier, so every feasible step costs at least its floor term and all excesses are non-negative. The benchmark OPT is the cheapest policy that never occupies a tier above the running maximum requirement $\max_{s\leq t} \ell_s$. Anticipating a requirement before it appears is the job of a predictor (Prop. 1); against a benchmark allowed to anticipate, a long low-requirement prefix followed by one high step makes any non-anticipating policy pay a first-visit rebuild that the benchmark avoids at negligible rent on a flat pool, so no online policy is competitive against it.

Two tiers. A window opens at t_0 when the requirement drops to ℓ while the policy holds h. Holding step t costs the rent $\pi_t = (p^{\mathrm{cr}}_h - p^{\mathrm{cr}}_\ell)c_{t-1} + (p^{\mathrm{cw}}_h - p^{\mathrm{cw}}_\ell)g_t + (p^{\mathrm{out}}_h - p^{\mathrm{out}}_\ell)o_t \geq 0$. Dropping at step t costs the downgrade tax $D_t = (p^{\mathrm{cw}}_\ell - p^{\mathrm{cr}}_\ell)(c_{t-1} - w_\ell)$ of Eq. 1, after which steps at ℓ cost nothing in excess. Until the requirement returns to h at step e, a policy that dropped pays every upgrade tax the requirement forces on it, each of the form of Eq. 1; U_e is their sum. A policy still at h pays none of them. The threshold policy with parameter κ drops at the first step τ at which the rent accumulated in the window, including step τ, reaches $X_\tau = \sqrt{1+\kappa}\, D_\tau$.

Let $D^{\min}_{t_0}$ be the smallest downgrade tax any feasible history allows at t_0 (the largest prefix of a tier below h that can be retained, from the last step before the window at which such a tier was feasible). Assume $\rho \geq D_\tau / D^{\min}_{t_0}$ (with τ replaced by the last step considered if no drop occurs) and $\kappa \geq U_e / D_\tau$ for windows that close after the drop. Inside the window D_t is non-decreasing, because the retained prefix is fixed while ℓ is not played; so the game value is $\mathrm{OPT}_e = \min\big(\sum_{\text{window}} \pi,\ D^{\min}_{t_0} + U^{\mathrm{OPT}}_e \mathbf{1}[\text{closes}]\big)$. Write $s = \sqrt{1+\kappa}$.

(a) The window closes at e before any drop. The policy paid $\sum_{t<e} \pi_t < X_{e-1} \leq s\rho D^{\min}_{t_0}$; against the hold option the ratio is 1, against dropping it is below $s\rho$.

(b) Drop at τ, no return. The policy pays $\sum_{t<\tau} \pi_t + D_\tau < X_\tau + D_\tau$. If $\mathrm{OPT}_e = D^{\min}_{t_0}$ the ratio is at most $(1+s)\rho$; if $\mathrm{OPT}_e = \sum \pi \geq X_\tau$ it is at most $1 + 1/s \leq 2$.

(c) Drop at τ, return at e. The policy pays less than $X_\tau + D_\tau + U_e$. If OPT drops at or before τ, it retains a prefix of h no longer than the policy's, so $U^{\mathrm{OPT}}_e \geq U_e$ and the ratio is at most $(1+s)\rho$. If OPT holds past τ, its rent already reaches X_τ and the ratio is at most $1 + (1+\kappa)D_\tau/(sD_\tau) = 1 + s$.

In every case the window's excess is at most $(1 + \sqrt{1+\kappa})\rho\, \mathrm{OPT}_e$. Lower bound. With constant context ($\rho = 1$) a deterministic policy is a threshold X on accumulated rent. The adversary either ends the session immediately after the drop (cost $X + D$ against $\min(X, D)$) or returns the requirement immediately after it (cost $X + D + U$ against $\min(X, D + U)$). Equating the two ratios gives $X = D\sqrt{1+\kappa}$ with $\kappa = U/D$ and the value $1 + \sqrt{1+\kappa}$; any other X makes one of them larger. A free return ($\kappa = 0$) recovers the classical 2.

K tiers. The policy occupies one tier at a time, so the K-tier problem is a sequence of temporally disjoint binary games rather than K coupled boundary games. Call step t tight if the policy sits at the requirement and slack otherwise; a holding episode is a maximal run of slack steps, opened by a forced upgrade and closed by a downgrade or a further forced upgrade. Upgrade taxes the requirement forces after a drop are charged to the window that

the drop opened, as part of its U_e, so all optional cost lives in episodes. Within an episode the tier is fixed at h and the requirement is $\ell < h$, which is exactly the two-tier game above.

Lemma 1 (Episode decomposition). For every policy P feasible for the benchmark and every holding episode e, the excess P incurs inside the episode's window is at least OPT_e.

Proof. (i) The floor is common: by price monotonicity every feasible policy pays at least the floor at every step. (ii) Every policy enters high: an episode opens only after a step whose requirement equalled h, which forces every feasible policy to a tier $\geq h$. (iii) Any exit pays the game's price: let s^* be the first step of the window at which P is below h. Before s^*, P sits at a tier $\geq h$ and by monotonicity pays at least π_t. At s^*, moving to any tier $m \geq \ell$ costs, in excess of the floor, at least $p_m^{\mathrm{cr}} w + p_m^{\mathrm{cw}}(c_{s^*-1} - w) - p_\ell^{\mathrm{cr}} c_{s^*-1} \geq (p_\ell^{\mathrm{cw}} - p_\ell^{\mathrm{cr}})(c_{s^*-1} - w) \geq D_{s^*}^{\min} \geq D_{t_0}^{\min}$, since no tier below h can retain more than the prefix that defines $D^{\min}$. If the requirement rises while P is below h, P pays its own upgrade taxes. Hence P's excess is at least the minimum of holding and dropping, which is OPT_e. □

Retained caches do not couple episodes. A visit to an intermediate tier before an episode can only lengthen that tier's retained prefix, and $D_{t_0}^{\min}$ already uses the longest prefix any feasible history retains below h. The bound of step (iii) therefore holds whatever the policy did in earlier episodes.

Episodes are disjoint and excesses additive and non-negative, so $\mathrm{OPT} - F \geq \sum_e \mathrm{OPT}_e$ while $\mathrm{ALG} - F = \sum_e \mathrm{ALG}_e \leq (1 + \sqrt{1+\kappa})\rho \sum_e \mathrm{OPT}_e$; adding F to both sides gives the theorem, with K absent throughout. The constant is K-independent because the state is one-dimensional and every decision is the same binary rent-or-buy against the current requirement, which is how the ordered, forced-up/optional-down structure escapes the $O(\log K)$ that general metrical task systems pay for K mutually accessible states. □

Numerical check. e106_thm1_verify.py draws random two-tier instances (growing contexts, monotone random prices, $q \in \{0, 0.5, 1\}$, one or two windows, with and without a prior visit to the lower tier, closing or not), bills the threshold policy with the four-bucket billing of §3, computes the benchmark exactly, and compares their excesses with $(1 + \sqrt{1+\kappa})\rho$. Over 3,000 instances the bound holds without exception; the largest observed excess ratio is 4.36 and the largest ratio to the bound 0.95. e111_thm1_verify_K3.py repeats the check with three tiers, where the requirement can move between the two lower tiers during a window: over 2,694 instances with a positive benchmark excess the bound again holds without exception, reaching 0.98 of its value.

Setting κ online (two tiers). After a drop at τ the held tier h retains at least $c_{\tau-1}$, and if context grows by at most a factor χ until the requirement returns, the one upgrade back to h gives $U_e \leq (p_h^{\mathrm{cw}} - p_h^{\mathrm{cr}})(\chi - q)\, c_{\tau-1} = \hat{\kappa}_\tau D_\tau$. With more tiers the requirement can force several intermediate upgrades, and $\hat{\kappa}$ would have to bound their sum. The proof uses κ only through $U_e \leq \kappa D_\tau$ at the drop step and through $s = \sqrt{1+\kappa}$ at the drop or last step, so a policy that uses $\hat{\kappa}_t$ at every step is $(1 + \sqrt{1 + \max_t \hat{\kappa}_t})\rho$-competitive. e123_thm1_online_kappa.py runs this policy on the 3,000 two-tier instances above with $\chi = c_T / c_{t-1}$. The bound holds on every instance, reaching 0.91 of its value. $\hat{\kappa}$ is a median 2.1× the realised κ (90th percentile 5.6), and the policy's excess cost is a median 1.00× that of the threshold policy given the realised κ (90th percentile 1.46).

M.3 Theorem 2 and Proposition 1

Theorem 2 (structural optimality). Fix a price-flat pool. Under uniform prices the per-step cost depends only on (switch?, c_t, c_{t-1}) and not on which model is chosen: a stay step costs $p^{\mathrm{cr}} c_{t-1} + p^{\mathrm{cw}}(c_t - c_{t-1}) + p^{\mathrm{out}} o_t$ and a switch or cold step costs $p^{\mathrm{cw}} c_t + p^{\mathrm{out}} o_t$; their difference is $(p^{\mathrm{cr}} - p^{\mathrm{cw}}) c_{t-1} \leq 0$ since $p^{\mathrm{cr}} \leq p^{\mathrm{cw}}$. Hence every switch weakly increases cost relative to staying, and the output term is model-independent. Let π' hold $L^* = \max_t \ell_t$ throughout; it is feasible ($L^* \geq \ell_t$) and makes zero switches, so $\mathrm{cost}(\pi') \leq \mathrm{cost}(\pi)$ for any feasible π. Thus $\mathrm{OPT} = \pi'$, proving (i); commit-to-L^* is π', proving (ii). □

Proposition 1 (exact decomposition). On correctly predicted sessions ($\hat{L} \geq L^*$) commit holds a feasible tier with no switch and, by Thm. 2, equals OPT. On under-predicted sessions commit runs at lower tiers early, at identical per-token price, and pays one forced rebuild $(p^{\mathrm{cw}} - p^{\mathrm{cr}})c_{t-1}$ at each step where the requirement first exceeds the held tier. Therefore $\mathrm{cost}(\mathrm{commit}) = \mathrm{OPT} + \sum_{\text{under-shot}}(\text{forced taxes})$, and dividing by OPT gives the identity $1 + \eta\bar{\kappa}$ with $\eta, \bar{\kappa}$ as defined; it is a ratio of sums, hence exact. □

Proposition 1, second claim (the ratio tracks N^*). Commit-perfect holds L^*; OPT's only possible advantage is to serve steps with $\ell_t < L^*$ at a cheaper tier, saving $(p^{\mathrm{cr}}_{L^*} - p^{\mathrm{cr}}_{\ell_t})c_{t-1}$ per step at the price of a down-then-up rebuild pair. By the definition of N^* this pair is net-profitable only for a downgrade held longer than N^* steps. If $N^* \geq T$, no downgrade within a length-T trajectory can be held that long, so OPT never downgrades and equals commit-perfect; if $N^* < T$, profitable downgrades exist and OPT is strictly cheaper, by an amount increasing as $N^*/T \to 0$. Formally $\mathrm{commit}/\mathrm{OPT} \leq 1 + \big(\sum_{\text{intervals}}[\text{saving} - \text{round-trip tax}]^+\big)/\mathrm{OPT}$, the summand positive only on intervals of length $> N^*$. □

Numerical instantiation. The bound is nonzero only on downgrade pairs with $N^*_{\mathrm{pair}} < T$; on the flagship pool (real prices, T=20) 71% of pairs qualify (median $N^*_{\mathrm{pair}} = 11.7$), giving 39 eligible pairs. Evaluating the bracketed term per pair (typical session context 30k tokens) and summing gives a median per-session gap of \$0.11 and a 90th-percentile gap of \$0.30, against a typical commit bill of \$5–\$15, a gap of roughly 1–6%. The realised commit/OPT ratios in the main text are of the same order.

M.4 Proposition 2: peer-pool lower bound

Two peer models with $p^{\mathrm{cr}}_1 = r(1+\varepsilon)$, $p^{\mathrm{cr}}_0 = r$, constant context c (growth only worsens the online player). An opening pulse forces everyone to model 1; the adversary then controls when the next pulse closes the 0-window. Premium $\pi = \varepsilon r c$, downgrade rebuild $B = p^{\mathrm{cw}}_0 c$, return rebuild $U = p^{\mathrm{cw}}_1 c$. A deterministic policy is a drop time τ: never dropping lets the adversary run the window forever (ratio unbounded); dropping at τ triggers a pulse at $\tau{+}1$, costing $\pi\tau + B + U$ against $\mathrm{OPT} = \min(\pi(\tau{+}1),\, B + U)$. The two branches meet at $\pi(\tau{+}1) = B + U$ with ratio $2 - \pi/(B+U) = 2 - o(1)$, where $o(1) \to 0$ as the round-trip horizon $(B+U)/\pi \to \infty$. Yao's principle over exponentially distributed window lengths gives $e/(e-1) - o(1)$ for randomised policies. □

M.5 Theorem 3 and Proposition 3

Lower bound. Fix any algorithm. The adversary plays two phases. Phase 1 ($t \leq T/2$): every arm has loss $\frac{1}{2}$, so the phase carries no information and any switch only adds cost. Phase 2: the adversary embeds the Dekel et al. (2014) instance with horizon $T' = T/2$. Two ingredients complete the reduction. (i) Monotonicity: every Phase-2 switch costs $\lambda t \geq \Lambda := \lambda T/2$; running the same algorithm against per-switch cost exactly Λ can only lower its total cost while the best-fixed-arm benchmark is unchanged, so our regret dominates the constant-cost-Λ regret. (ii) The constant-cost bound: for per-switch cost $c \in [1, T']$ the minimax policy regret of a T'-round bandit is $\tilde{\Omega}(c^{1/3}T'^{2/3})$ (Dekel et al., 2014). Substituting $c = \Lambda = \lambda T/2$ (valid since $\lambda \leq 1$ gives $\Lambda \leq T'$) yields $\tilde{\Omega}((\lambda T)^{1/3}T^{2/3}) = \tilde{\Omega}(\lambda^{1/3}T)$. Phase 1 gave the algorithm nothing, so no head start reduces the Phase-2 bound. □

Upper bound. Batch rounds into blocks of length τ and run EXP3 over block-averaged losses, switching only at block boundaries (Arora et al., 2012). Switch cost: at most one switch per boundary, $\sum_b \lambda(b\tau) \leq \lambda T^2/\tau$. Learning cost: $\tilde{O}(\tau\sqrt{K(T/\tau)}) = \tilde{O}(\sqrt{KT\tau})$. Balancing at $\tau^* = \lambda^{2/3}T/K^{1/3}$ gives $\tilde{O}(\lambda^{1/3}K^{1/3}T)$. □

Stochastic ETC (Prop. 3, first claim). Round-robin the K arms for T_0 rounds; every switch happens at $t \leq T_0$, so exploration switching costs at most λT_0^2. Commit to the empirical best; a standard concentration argument prices the misidentification cost at $\tilde{O}(T\sqrt{K/T_0})$. Balancing the two, $\lambda T_0^2 = T\sqrt{K/T_0}$, gives $T_0 = (T\sqrt{K}/\lambda)^{2/5}$ and regret $\tilde{O}(\lambda^{1/5}K^{2/5}T^{4/5})$. □

Commitment horizon (Prop. 3, second claim). Let π^* be offline-optimal and suppose it switches at a round t with $\lambda t > \Delta_{\max}(T{-}t{+}1)$. Let s be its next switch round after t ($s = T{+}1$ if none). Modify π^* into π': on $[t, s{-}1]$ keep the arm held at $t{-}1$; from s on, coincide with π^*. (If π^* switches at s, π' makes the same switch; the tax λs depends only on the round index, not on the arms involved, so this costs π' exactly what it costs π^*; if π^*'s arm at s equals π''s current arm, π' saves a further tax.) π' saves the tax λt and loses per-round value at most $\Delta_{\max}$ on the $s - t \le T{-}t{+}1$ rounds of $[t, s{-}1]$, so its utility changes by at least $\lambda t - \Delta_{\max}(T{-}t{+}1) > 0$, contradicting optimality. The constraint $\lambda t \le \Delta_{\max}(T{-}t{+}1)$ has a non-increasing right-hand side and an increasing left-hand side, giving $t^* = \lfloor \Delta_{\max}(T{+}1)/(\lambda + \Delta_{\max}) \rfloor$. Tightness: fix $t \le t^*{-}1$ and an instance where one arm gains $\Delta_{\max}$ over all others on every round $\ge t$ and all arms tie before t; switching at t nets $\Delta_{\max}(T{-}t{+}1) - \lambda t > 0$, so the optimum does switch at t. □

Stationary cutoff. Let DC(k) free-route rounds $1..k$ and hold thereafter. Under the stationarity hypothesis (each round's argmax carries advantage v over any arm held from an earlier round, in expectation), extending the free phase by one round adds value v at round $k{+}1$ and one switch tax $\lambda(k{+}1)$. Charging a full switch at every free round is the conservative accounting, since the round's argmax sometimes coincides with the held arm and is then free; the resulting t^*_{cut} is correspondingly a lower bound. The held-phase expectation is unchanged, so $U(\text{DC}(k{+}1)) - U(\text{DC}(k)) = v - \lambda(k{+}1)$, positive exactly while $k{+}1 \le v/\lambda$, giving $t^*_{\text{cut}} = \lfloor v/\lambda \rfloor$ up to integer rounding. □

Simulation. $K{=}2$, $\lambda{=}0.02$, $T \in [2\text{k}, 64\text{k}]$, 10 seeds. Stochastic (gap 0.15): fitted log–log slopes are ETC 0.79 (theory 0.80), per-step EXP3 1.14, batched EXP3 1.01. On the two-phase lower-bound instance (gap $\varepsilon = \lambda^{1/3}$): never-switch 1.08, per-step EXP3 1.80, batched EXP3 1.01 at $0.17T$, against the theoretical scale $\lambda^{1/3}T = 0.27T$.

M.6 Proposition 4: the optimal gate

The expected error cost is $J(\theta) = \bar{d}\,\bar{x}\,\bar{Q}(\theta/\sigma) + V\bar{G}\,\Phi_G(\theta)$, where $\bar{Q}$ is the standard normal upper tail and $\Phi_G(\theta) = \int_{-\infty}^{\theta} f(u)\,du$ the gain-weighted mass of genuine-gain steps the gate blocks. An interior minimiser satisfies $J'(\theta) = 0$, i.e. $\bar{d}\bar{x}\,\phi(\theta/\sigma)/\sigma = V\bar{G}f(\theta)$ with ϕ the standard normal density. Taking logarithms, $\theta^2/(2\sigma^2) = \ln\Lambda - \ln s(\theta)$ with $s(\theta) = \sqrt{2\pi}\,\sigma f(\theta) \in [s_-, s_+]$, hence $\theta^\star/\sigma = \sqrt{2\ln\Lambda}\,\sqrt{1 - \ln s(\theta^\star)/\ln\Lambda} = \sqrt{2\ln\Lambda}\,\big(1 + O(1/\ln\Lambda)\big)$, uniformly over routers once $s_\pm$ are fixed. Since $s(\theta^\star) \in [s_-, s_+]$ the same identity gives the two-sided bound of the proposition without any limit. The bracket collects only $\bar{d}, \bar{x}, V, \bar{G}$; σ enters as a multiplicative scale. □

N Numerical verification of the bounds

On 77 real trajectories under peer prices: the lemma bound $L \le \text{OPT}$ holds 77/77; the covered-prediction bound holds on all monotone trajectories (the single violation is a non-monotone prefix-replacement trajectory, which motivates the standing assumption of App. M); worst-case under-prediction ($p{=}0$) realises $1.225 \times \text{OPT}$ (mean) against a record-instantiated bound of 1.317; forced escalations average 0.71 per trajectory (max 2), occurring at median context fraction $0.34\,c_T$.

O RL validation of the delayed-commit schedule

We compare the two-phase structure with trained policies on real traces: the 11-model flagship pool (LLMRouterBench queries; $T{=}20$ domain-grouped sessions; 400 sessions $\times 5$ evaluation seeds), utility $Q - \lambda_\$ \cdot \text{bill}$, with Q the quality, the bill in dollars per session-step and $\lambda_\${=}20, and single-slot billing.

(a) A free REINFORCE learner on the routing MDP converges to a commit-like policy (switch rate $x \approx 0$). (b) On the joint routing×keepalive MDP, 4/5 PPO seeds collapse to low-quality policies (≈ 0.35). The fifth reaches 0.488 ± 0.007 quality at $\$125.81 \pm 8.44$ per 400 sessions over 5 test seeds, by mixing models early. (c) The delayed-commit rule

dominates it. Its free phase uses the per-query hindsight argmax, so this compares the two-phase structure, not two deployable routers.

Table 12: Delayed commit versus a trained policy. DC(t_{cut}) for every $t_{\text{cut}} \in \{2, 3, 5, 8\}$ Pareto-dominates the trained policy on both axes; its free phase uses the hindsight argmax.

t_{cut}	quality	bill (\$/400 sess.)
1	0.482	95.94
2	0.500	90.77
3	0.511	92.83
5	0.543	96.42
8	0.596	104.05
12	0.663	128.20
20	0.791	202.34
PPO (best of 5 seeds)	0.488	125.81
value-commit + always-ping	0.485	161.06

At the learner's own objective ($\lambda_{\$}$=20) the best cutoff nearly doubles its utility (0.342 at t_{cut}=12 vs. 0.173).

The two ingredients, measured. Quality is linear in t_{cut} (slope 0.0163 per step, R^2=0.9995), so the per-round gain of free routing over holding is the constant $v = 0.0163$ in quality. The marginal bill of one more free round grows with the round index, from \$0.005 to \$0.023 per session per step. The fitted tax slope is $\hat{\beta} = \$0.00147$ per round^2 per session, which the utility charges at $\lambda_{\$}/T = 1$ per session dollar, so $\hat{\lambda} = 0.00147$. The closed form predicts $t^*_{\text{cut}} = v/\hat{\lambda} = 11.1$; the observed utility argmax is 12, with utility flat over $[8, 12]$. At $\lambda_{\$}$=60 it predicts 3.7, against a flat optimum over $[2, 8]$ with argmax 2. The ratio $\hat{\beta}/\bar{n}$ ($\bar{n} \approx 7.2$k tokens per round) gives an effective write–read spread of \$0.20/Mtok, the spread of the cheap arms the free phase holds (pool mean \$0.69). The dip in the bill column from t_{cut}=1 to 2 comes from the same cheap-arm mixing. On heterogeneous pools the commitment horizon is therefore computed with the spread of the arms a policy holds, not the pool mean.

P Assumptions of Prop. 4

The derivation assumes Gaussian noise and a margin density bounded away from zero and infinity near the optimum; the five routers measured here have near-Gaussian margin kernels. It also assumes $\hat{G} = G + \text{noise}$. If a router's margin is uninformative (AUC ≈ 0.5), $\theta^\star$ is undefined; this is one of the two conditions of §7.