WHEN STATE INFORMATION PAYS: COST SEPARATION IN SEQUENTIAL MULTI-MODEL ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Errors in agent trajectories can change the value of later model calls. We isolate this coupling in a two-state transmission model and compare state-conditioned policies with both deterministic and randomized open-loop schedules. Under capability ordering, the best schedule with a fixed number of strong calls has a strong suffix. This yields an exact deterministic cost in an explicit target window and reduces the randomized optimum to the lower convex envelope of $K+1$ suffix points. A feasible state policy then gives a matched- class cost-separation certificate. In a controlled instance, randomization reduces the open-loop cost from 12 to 5.32. The exact randomized state-history optimum costs 3.56, leaving a matched information gap of 1.76. A noisy-signal certificate vanishes under an uninformative signal, separating information from exogenous randomization. We evaluate a gold-free detector cascade on a real-execution data workflow. After calibrating a state-independent random trigger on costs alone, we freeze it before testing on 200 disjoint tasks with three fresh repetitions per policy (1,200 episodes). StopRoute reaches 79.50% success versus 74.50%, a 5.00-point paired gain on average, while the nominal-cost gap is 2.67%, inside the prespecified 5% point-estimate tolerance. Together, the exact model analysis and controlled online comparison isolate the value of where additional model calls are allocated, beyond randomization or heterogeneous model assignment.

1 INTRODUCTION

A growing share of LLM inference is spent inside *agent trajectories*: multi-step sequences of planning, tool calls, code execution, and verification. To control cost, a router can choose from a pool of models at each step. Its information can range from task features and the step index to the entire execution history. Sequential routers already exploit uncertainty, progress, and budget signals (Section 2). We isolate the cost of withholding information about execution state from an otherwise optimized routing policy.

Our question is: *under what conditions does observing execution state reduce the cost of achieving a terminal success target?* In the model, failures can contaminate subsequent inputs, while successful calls restore a clean state. Comparing deterministic and randomized schedules with state-conditioned policies separates the value of information from policy-learning performance. The API case study then examines observable failure checks in a fork–join data workflow, whose execution graph and observation process differ from the transmission model.

A valid comparison must give open-loop routing its optimal strong-call placement and episode-level randomization. Our theory supplies that control, while the online study fixes primary assignments and calibrates random fallback placement on cost alone.

Our main contributions are:

(1) **Exact open-loop structure.** In a two-state transmission chain, we prove that strong calls are optimally placed as a suffix. This reduces randomized open-loop optimization from arbitrary schedules to the lower convex envelope of $K+1$ suffix points.

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.

(2) **An information gap after randomization.** A feasible state policy certifies residual cost separation against that optimized randomized baseline. Exact enumeration gives a gap of 1.76 in a controlled instance. The noisy-signal certificate is zero for an uninformative signal. Multi-tier witnesses and a soft-budget reservation rule provide supporting extensions.

(3) **A cost-calibrated online test.** A frozen, gold-free detector cascade improves success by 5.00 points on 200 held-out DataFlow tasks, with three fresh repetitions per policy, after independent cost-only calibration of a random-trigger control. The 2.67% nominal-cost gap meets the prespecified 5% tolerance. Four earlier prospective cohorts document both replication and the importance of checking this cost condition.

2 Related Work

Query-level routing and cascading. RouteLLM (Ong et al., 2025), Hybrid LLM (Ding et al., 2024), RouterDC (Chen et al., 2024b), GraphRouter (Feng et al., 2025), and TensorOpera Router (Stripelis et al., 2024) select a model before generation. FrugalGPT (Chen et al., 2024a), AutoMix (Aggarwal et al., 2024), and token-uncertainty cascades (Gupta et al., 2024) may run a candidate before escalating. Select-then-Route (Shah and Shridhar, 2025) combines taxonomy-based pool selection with a confidence cascade, and LLMSelector (Chen et al., 2025) assigns models to fixed modules in compound systems. RouterBench (Hu et al., 2024) and RouterEval (Huang et al., 2025) benchmark query routing. PILOT (Panda et al., 2025) learns it online under a workload budget. These settings optimize individual queries or fixed modules, rather than transmission state within one trajectory.

Step-level and agentic routing. AgentRouter (Paul and Nandy, 2026), TRIM (Kapoor et al., 2026), and STEER (Lee et al., 2026) make per-step decisions from task or confidence signals. Boundary-guided routing uses path-dependent per-task budgets (Zhang et al., 2026). SWE-Router (Son et al., 2026) conditions escalation on a partial trajectory, and STAR (Yang et al., 2026) routes from typed execution states. ProgRouter (Li et al., 2026), harness-native routing (Liu et al., 2026), and TACIT-Switch (Lei and Huang, 2026) use progress, execution traces, or accumulated risk. Router-R1 (Zhang et al., 2025) and MasRouter (Yue et al., 2025) address multi-round or multi-agent routing. SWE-Router's Bayes comparison formalizes the benefit of informative partial trajectories. Our complementary result makes each model choice alter the state presented to later calls and derives the resulting suffix and cost structure. We add a gold-free detector cascade evaluated by recorded-tree replay and an online fresh-call replication. The trajectories follow the ReAct (Yao et al., 2023) paradigm.

Process feedback and stepwise benchmarks. RoRo (Ye et al., 2026) combines rubric-guided process rewards with outcome rewards to train a stepwise router. Policy-guided routing (Si et al., 2026) uses reinforcement learning and threshold calibration for constrained reasoning cost. These methods learn decisions from intermediate reasoning states. Our analysis supplies a fixed-model information comparison and a terminal-reward scope condition. TwinRouterBench (Yang et al., 2026) combines static router-visible prefixes with live software-agent evaluation. Its dynamic track is a relevant test of external validity for future detector-cascade studies. The present experiments use structured data workflows.

Theory of routing and stopping. Classical cascade construction optimizes pretrained prediction models subject to accuracy constraints (Streeter, 2018). Optimal stopping and Pandora's-box search (Weitzman, 1979) and constrained MDPs (Altman, 1999) provide related search and budget

Closest trajectory-routing work.

	Step/turn signal	Failure state	Formal result
TRIM (Kapoor et al., 2026)	uncertainty	erroneous step	routing policies
Budget-aware (Zhang et al., 2026)	budget/history	implicit	sequential objective
SWE-Router (Son et al., 2026)	partial trajectory	implicit	Bayes comparison
STAR (Yang et al., 2026)	execution trace	typed	support result
TACIT-Switch (Lei and Huang, 2026)	cumulative risk	handoff state	survival model
Ours: theory	observed state	binary state	cost certificate
Ours: case study	output checks	memory proxy	online comparison

machinery. Dekoninck et al. (Dekoninck et al., 2025) unify routing and cascading under a quality estimator. Bouchard (Bouchard, 2026) characterizes threshold-cascade frontiers and the structural cost of paying for a weak response before escalation, while RLCascadeRouter (Huang et al., 2026) learns stop-or-route actions from returns. Our coupling is across steps: each model choice changes the state faced by later calls. Theorem 1 and Proposition 1 give matched deterministic and randomized cost-separation results. Theorem 2 extends them by feasible multi-tier witnesses and exact small-horizon enumeration.

3 Problem Setup

Definition 1 (Transmission chain). *A trajectory has K steps, starts at $X_1 = 0$, and has time-homogeneous success probabilities and per-call costs. The state is $X_t \in \{0, 1\}$ ($0 =$ clean, $1 =$ corrupted). Choosing model m at step t costs $c(m)$ and succeeds with probability $p(m, X_t)$. A successful step sets $X_{t+1} = 0$. After a failed step, $X_{t+1} = 1$ with probability γ and $X_{t+1} = 0$ with probability $1 - \gamma$. Thus γ is the probability that a step failure corrupts the downstream state. The terminal reward is $R \cdot \mathbf{1}\{S\}$ where S is the event that the final step is correct.*

Assumption 1 (Monotonicity). *$p(m, 0) \geq p(m, 1)$ for all m (clean states are easier), and tiers are capability-ordered. For the two-model case $\{w, s\}$ with $c_w < c_s$, write $a_x = p(w, x)$, $b_x = p(s, x)$, with $a_0 \geq a_1$, $b_0 \geq b_1$, $b_x \geq a_x$.*

Definition 2 (Policy classes). *For each fixed task/parameter instance, Π_{marg} consists of deterministic open-loop schedules $m_{1:K}$. Task features may index a schedule, but within-task external randomization and early stopping are excluded. Π_{state} consists of deterministic policies observing the state history $X_{1:t}$ and past actions. It includes Markov policies $m_t = \pi(t, X_t)$ and all fixed schedules. Every policy makes exactly K calls. $\Pi^{\text{rand}}_{\text{marg}}$ contains episode-level mixtures over Π_{marg} whose random seed is independent of the chain, and $\Pi^{\text{rand}}_{\text{state}}$ is defined analogously over Π_{state}. Thus $\Pi^{\text{rand}}_{\text{marg}} \subseteq \Pi^{\text{rand}}_{\text{state}}$.*

We study $V^\star = \min_{\pi \in \Pi_{\text{state}}} \mathbb{E}[C(\pi)]$ s.t. $\Pr(S) \geq s^*$, and V_{marg} defined identically over Π_{marg}. $V^{\text{rand}}_{\text{marg}}$ and $V^{\text{rand}}_{\text{state}}$ denote the corresponding optima over the randomized classes. Here X_t is observed by the ideal state-policy class. Section 4.5 introduces noisy observations. Parameters are known in the analysis. In the API study, the binary correlation ϕ measures adjacent errors, while a separately defined memory proxy supports conditional descriptive rates. Neither identifies γ or a homogeneous $p(m, x)$. Randomized schedule mixtures convexify the success–cost set. Proposition 1 characterizes the resulting optimum. History-dependent and Markov constrained optima may also differ. The soft-budget optimum has a Markov representative.

4 The Structural Cost of Stepwise Decisions

4.1 Suffix optimality of open-loop schedules

Lemma 1 (Suffix optimality). *In the two-model transmission chain with terminal reward, fix the number of strong steps $j \geq 1$. Under Assumption 1, $\Pr(S)$ over all schedules containing j strong steps is maximized by placing the strong steps as a contiguous suffix.*

The proof repeatedly swaps an adjacent (s, w) pair to (w, s). Write $d_x = b_x - a_x \geq 0$ and $\alpha_w = 1 - \gamma(1 - a_1)$. For an internal pair, the swap increases the outgoing clean-state probability by $\gamma[\alpha_w d_0 + \gamma(1 - a_0) d_1] \geq 0$. For the final pair it increases terminal success by the bracketed quantity. Exhaustive enumeration checks all $j \in \{1, \ldots, K - 1\}$ on 60 random instances (Appendix B).

Remark 1. *A natural conjecture that the optimal marginal schedule is always "all-weak or all-strong" is false: under terminal reward, a single strong step is optimally placed at the end under Assumption 1. This suffix structure gives the exact open-loop cost used in the bound below.*

4.2 The separation bound

Let v^w_t be the clean-state occupancy under the all-weak policy. With $\alpha_w = 1 - \gamma(1 - a_1)$ and $\beta_w = \gamma(a_0 - a_1)$, $v^w_1 = 1$ and $v^w_{t+1} = \alpha_w + \beta_w v^w_t$. Define $g_0 = \Pr(\text{all } w) = a_1 + (a_0 - a_1) v^w_K$

and $g_1 = \Pr(\text{strong last, weak elsewhere}) = b_0 v_K^w + b_1(1 - v_K^w)$. Let $\pi^{\dagger\prime}$ be the state-conditioned policy "weak throughout, strong at step K iff $X_K = 1$", with $P^{\dagger\prime} = a_0 v_K^w + b_1(1 - v_K^w)$.

Theorem 1 (Deterministic open-loop cost and separation bound)**.** *If the success target lies in the window $s^* \in (g_0,\ \min(g_1, P^{\dagger\prime})]$, then*

(i) $V_{\text{marg}} = (K-1)c_w + c_s$ *exactly;*

(ii) $V_{\text{marg}} - V^\star \ \geq\ (c_s - c_w)\, v_K^w$.

Proof sketch. (i) By Lemma 1 the best 1-strong-step schedule places it last, achieving $g_1 \geq s^*$. The all-weak schedule is infeasible ($s^* > g_0$), and cost strictly increases in j. (ii) $\pi^{\dagger\prime}$ costs $C^{\dagger\prime} = Kc_w + (c_s - c_w)(1 - v_K^w)$ (the strong call happens only when the final state is corrupted, probability $1 - v_K^w$) and is feasible in the window, so $V^\star \leq C^{\dagger\prime}$. Subtracting yields the bound. Appendix B gives the full proof and exact-enumeration checks on 60 instances. □

Proposition 1 (Randomized open-loop optimum and matched certificate)**.** *For $j = 0, \ldots, K$, let $\sigma_j = (w^{K-j}, s^j)$ be the schedule with a j-step strong suffix, let $G_j = \Pr(S \mid \sigma_j)$, and let $C_j = (K-j)c_w + jc_s$. Whenever the target is feasible,*

$$V_{\text{marg}}^{\text{rand}} = \min_{q \in \Delta^K} \left\{ \sum_{j=0}^{K} q_j C_j : \sum_{j=0}^{K} q_j G_j \geq s^* \right\}.$$

Hence the randomized optimum is the lower convex envelope of the $K+1$ suffix points (G_j, C_j). If a state-conditioned witness with success at least s^* has expected cost $C^\dagger$, then

$$V_{\text{marg}}^{\text{rand}} - V_{\text{state}}^{\text{rand}} \geq \max\{0, V_{\text{marg}}^{\text{rand}} - C^\dagger\}.$$

Proof sketch. Every schedule with j strong calls has cost C_j and, by Lemma 1, success at most G_j. Replacing each component of any open-loop mixture by σ_j preserves cost and weakly increases success. The converse inclusion is immediate. The certificate follows because the randomized state class contains the randomized open-loop class and the stated witness. □

Proposition 1 therefore distinguishes an exact randomized open-loop optimum from a witness-based certificate. The API detector is only an empirical state proxy, so its results test the routing motivation rather than the latent-chain bound.

Remark 2 (The gap is an absolute quantity)**.** *For $\beta_w < 1$, as $K \to \infty$, $(c_s - c_w)v_K^w \to (c_s - c_w)\alpha_w/(1 - \beta_w)$: each trajectory saves "one strong call times the clean probability". The* relative certificate *decays with K against the one-strong schedule cost, when its target window is applicable. This concerns the certificate. the full optimal-policy gap can have a different scaling. The empirical heuristic–oracle statistic in Section 6.4 is a separate quantity.*

Where does the window fall on real domains? We replay an analogous final-step witness on each legacy trace realization (w = DeepSeek, s = Kimi-K3), using any earlier gold-labeled failure as a history proxy. The resulting target windows are empty in all four domains (Appendix J). Because this proxy and the workflow need not obey Definition 1, these are empirical witness checks, rather than estimates of the theorem's latent parameters. The controlled experiment below supplies the direct model test. The API experiment evaluates cascades.

4.3 Multi-tier pools

Theorem 2 (Computable multi-tier certificate)**.** *Let V_{marg} be computed over all L^K open-loop schedules. For each $i < L$, let $\pi^\dagger(i)$ use tier i in the clean state and tier L in the corrupted state, and let $F = \{i : \Pr(\pi^\dagger(i)) \geq s^*\}$. If F is non-empty, then*

$$V_{\text{marg}} - V^\star \ \geq\ \max\left\{0,\ V_{\text{marg}} - \min_{i \in F} C^\dagger(i)\right\}.$$

The certificate is computable from $v_1^\dagger = 1$ and $v_{t+1}^\dagger = 1 - \gamma(1 - p_{L,1}) + \gamma(p_{i,0} - p_{L,1})v_t^\dagger$, with $C^\dagger(i) = \sum_t [c_i v_t^\dagger + c_L(1 - v_t^\dagger)]$.

The proof uses only feasibility: every $\pi^\dagger(i)$ in F is a legal state-conditioned policy, hence $V^\star \le \min_{i\in F} C^\dagger(i)$, while $V^\star \le V_{\text{marg}}$. This formulation avoids assuming that the best open-loop schedule uses one tier uniformly. Exact enumeration, Pareto DP, and Monte Carlo tests are reported in Appendix B. The witness costs take $O(KL)$ operations in total. The stated exact open-loop enumeration takes $O(KL^K)$ time. This is a finite-horizon certificate from chosen feasible policies. It does not characterize the optimal multi-tier policy nor supplies a scalable routing algorithm.

4.4 Soft-budget routing as a reservation-value rule

For a soft (Lagrangian) budget, let $V_t(x)$ denote the optimal value from step t and state x, and let $\lambda \ge 0$ be the cost shadow price.

Theorem 3 (Reservation-value rule)**.** *For the soft-budget objective* $\max_\pi \mathbb{E}[R \cdot \mathbf{1}\{S\}] - \lambda\mathbb{E}[\sum_t c(m_t)]$*, there is an optimal deterministic policy with*

$$\pi_\lambda(t,x) = s \iff G_t(x) \ge \lambda(c_s - c_w),$$

where, for $t < K$, $G_t(x) = \gamma(b_x - a_x)[V_{t+1}(0) - V_{t+1}(1)]$*, and* $G_K(x) = R(b_x - a_x)$*. Moreover* $V_t(0) \ge V_t(1)$.

Remark 3 (Hard budgets)**.** *A remaining-budget formulation augments the state with* b_t *and is solved on the finite set of reachable budget sums. The soft-budget rule characterizes* λ*-indexed actions. Budget monotonicity requires the augmented DP.*

4.5 Noisy observations of corruption

A deployable router observes signals rather than an ideal state. To quantify this difference within the model, let $O_K \in \{0,1\}$ be a signal available *before* the final call, with $r = \Pr(O_K = 1 \mid X_K = 1)$ and $f = \Pr(O_K = 1 \mid X_K = 0)$ under the all-weak prefix. Conditional on X_K, the signal is independent of the final-call success randomness. Its acquisition has zero cost in this witness calculation.

Proposition 2 (Matched noisy-signal certificate)**.** *Use weak models up to step* $K-1$*, and choose* s *at* K *iff* $O_K = 1$*. Writing* $v = v_K^w$ *and* $d_x = b_x - a_x$*, its success and cost are*

$$P_O = g_0 + vfd_0 + (1-v)rd_1, \tag{1}$$

$$C_O = Kc_w + (c_s - c_w)[vf + (1-v)r]. \tag{2}$$

Let V_O^{rand} *be the optimum of a randomized observation-policy class that contains both* $\Pi_{\text{marg}}^{\text{rand}}$ *and this witness. If* $P_O \ge s^*$*, then*

$$V_{\text{marg}}^{\text{rand}} - V_O^{\text{rand}} \ge \max\{0, V_{\text{marg}}^{\text{rand}} - C_O\}.$$

If $r = f = q$*, the witness has the same success and expected cost as the open-loop mixture that chooses the one-strong suffix with probability* q *and all-weak otherwise, so the displayed certificate equals zero whenever the witness is feasible.*

Proof. Conditioning on (X_K, O_K) gives the displayed P_O and C_O. Feasibility implies $V_O^{\text{rand}} \le C_O$, while class inclusion gives $V_O^{\text{rand}} \le V_{\text{marg}}^{\text{rand}}$. Combining the two inequalities proves the certificate. If $r = f = q$, the final action is independent of X_K. Because final success and cost depend only on X_K and that action, an independent open-loop mixture attains the same (P_O, C_O). Therefore $V_{\text{marg}}^{\text{rand}} \le C_O$ and the bound is zero. □

A positive comparison with a deterministic schedule alone is therefore not an information certificate. Reducing recall saves calls but can make the target infeasible. False positives buy strong calls on clean inputs. A signal cost c_O adds to C_O. The API detector is a post-call observation process.

Controlled instance. Set $K = 3$, $\gamma = 0.8$, $(a_0, a_1) = (0.9, 0.2)$, $(b_0, b_1) = (0.98, 0.8)$, $(c_w, c_s) = (1, 10)$, and $s^* = 0.85$. At this target, deterministic and randomized open-loop costs are 12 and 5.321. Convexifying the state-history frontier gives cost 3.560 and an information gap of 1.760 (Figure 1a and Table 1). The uninformative signal $r = f \approx 0.2578$ attains the target at cost 5.321 and gives zero matched certificate. Panel (b) plots feasible witnesses only: at fixed false-positive rate, higher recall buys additional strong calls, lowering this particular cost certificate after

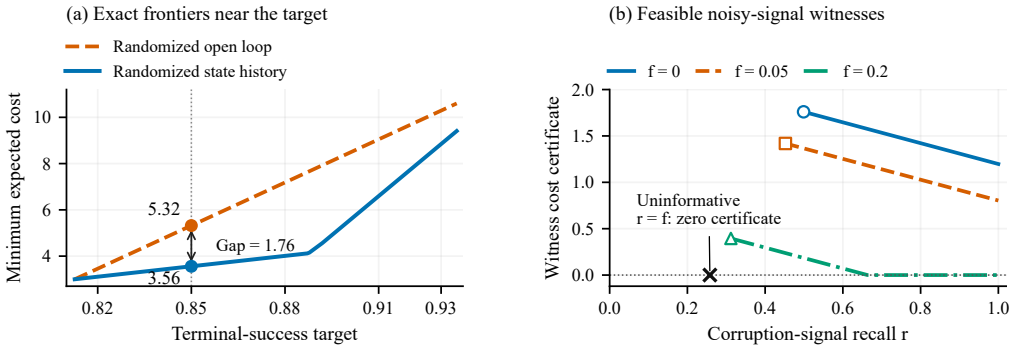

Figure 1: State-information value in the controlled model. Costs are normalized with $c_w = 1$, $c_s = 10$. **(a)** Exact randomized frontiers near target 0.85: state history reduces cost from 5.32 to 3.56. The full target range is in Appendix I. **(b)** Noisy-signal certificates for three false-positive rates f. Hollow markers indicate the minimum feasible recall. Infeasible portions are omitted. The cross marks an uninformative signal with zero certificate. These are model calculations, not API outcomes.

Table 1: Controlled comparison at target success 0.85. OL denotes open loop. Values are rounded to four significant digits. The deterministic row overshoots the target because suffix schedules are discrete. The randomized rows are exact optima before rounding. The first saving includes open-loop convexification, whereas the second is the matched-success saving from state information.

Policy class	Information	Randomization	Success	Min. cost	Saving
Deterministic OL	step index	none	0.9575	12.00	–
Randomized OL	step index	episode-level	0.8500	5.321	6.679 vs. det.
Randomized state history	state history	history-level	0.8500	3.560	1.760 vs. rand. OL

the success target is met. This does not mean that optimal information value decreases with recall. Independent simulation with 200,000 trajectories per each of six settings agrees within five Monte Carlo standard errors. Another 1,000 random algebra checks also pass. Appendix I gives the full-range frontiers, policy-class audit, and observation sweep.

Suffix optimality makes open-loop convexification finite and exact. Comparing the two randomized classes in Table 1 rules out randomization alone as the explanation and leaves a cost gap of 1.760 at the matched success target, using the unrounded values.

5 StopRoute

StopRoute is a finite detector-cascade policy family motivated by the value of conditioning expenditure on execution outcomes. Its actions include within-step retries and early termination, so its action space differs from the fixed-K model. It uses three mechanisms:

(1) **Development selection.** Enumerate candidate assignments or cascades. The original experiment maximizes development success, breaking ties by recorded cost. An alternative selects minimum cost at a development success target or maximum success under a development mean-cost budget.

(2) **Early stop.** When every model in the configured cascade fails the execution detector, terminate the trajectory rather than spend downstream cost on a missing intermediate result.

(3) **Escalation cascade.** When a detector rejects an output, retry the step using the next configured model on the same incoming prompt. The search allows both orders of each model pair, reflecting task-dependent strengths. The original policy is frozen before its test evaluation.

Gold-free failure detection. Escalation presumes that at least some failures are observable. In tool-use, code, and data agents, crashes and schema violations are detectable *without* gold labels via expression executability, output shape, and answer-format compliance. On held-out traces, the recorded detector has observed failure precision 1.00 and recall 0.765/0.760/0.683 at the three steps. No false rejection was observed, although this finite-sample result does not establish zero false-positive risk. Accepted outputs can remain semantically wrong (Section 6.4).

Router overhead. Calibration is offline. Online routing uses table lookups and local detector checks, without an additional model call.

Post-call decision interpretation. Let h denote the paid-for trace after a primary response and let o be its detector output. If A accepts the current response (or terminates after a missing result) and R_m retries with model m, a soft-budget controller retries exactly when

$$\mathbb{E}[U_t \mid h, o, R_m] - \mathbb{E}[U_t \mid h, o, A] \geq \lambda c(m),$$

where U_t is downstream success utility excluding the retry cost. StopRoute approximates this comparison with a deterministic eligibility check followed by development-set selection over a finite cascade family. This post-call rule connects detector-triggered fallback to the reservation-value view, while the pre-action signal in Proposition 2 quantifies a distinct observation channel.

6 EXPERIMENTS

Task and harness. The primary case study is DATAFLOW-V2, a real-execution data workflow on independently generated 1,200-row tables. Steps 1 and 2 write independent quarterly aggregation expressions, which are executed in pandas. Step 3 computes per-region growth from both resulting tables. This fork–join structure makes intermediate output validity relevant to downstream success. The model pool is Kimi, DeepSeek, and Qwen. Online experiments execute fresh calls for each policy and repetition. Replay experiments instead execute the full three-model assignment tree once per task and evaluate policies on stored nodes. A replay fallback uses a sibling node with the same incoming prompt, discards the rejected response, and follows the accepted prefix thereafter. This conditions on stored completions and assumes that the discarded response does not affect its sibling's distribution.

6.1 PRIMARY RESULT: COST-CALIBRATED ONLINE EVALUATION

Frozen policies and independent calibration. The selected StopRoute uses DeepSeek→Kimi at step 1, Kimi at step 2, and Qwen-Flash at step 3. Its original development selection is detailed in Section 6.3. The random-trigger control has the same primary assignments and detector stopping, but allocates step-1 Kimi fallbacks independently of detector output. On 40 calibration tasks, each repeated three times, we use costs alone to calibrate a state-independent step-1 fallback probability $q \approx 0.2307$. We then freeze 200 disjoint test tables and three fresh repetitions per policy, including exactly 46 randomly placed fallbacks per repetition for the control. Both the policy and random assignments are fixed before test execution. Calibration success outcomes do not set q. There are 1,200 episodes but *200 independent task units*: inference compares each task's policy means over three repetitions (Appendix H).

Table 2: Primary held-out online result: 200 tasks, three fresh repetitions per policy. Paired success gain: 5.00 points, 95% CI $[0.50, 9.33]$, $p = 0.0342$. Nominal-cost gap: 2.67%, within the pre-specified 5% point-estimate tolerance. Costs use frozen token weights. Wall time includes online execution and retry waiting.

Policy	Success	Cost/run	Calls/run	Wall (s)
StopRoute	**79.50%**	\$0.002044	3.042	35.04
Random trigger	74.50%	\$0.001989	2.978	35.41

Table 2 supports a success gain at nearby nominal cost, not simultaneous improvements in success and expenditure. The primary uncertainty and sign-flip test are reported in the caption, with full

Table 3: Prospective DataFlow-v2 evidence: StopRoute minus random triggering. Gains and 95% task-bootstrap CIs are in percentage points. Cost gap is the absolute mean-cost difference divided by StopRoute cost in every row. Repeats are per policy.

Study	Execution	Tasks × reps	Gain	95% CI	Cost gap
Replay-200	stored tree	200×1	8.33	$[4.81, 12.23]$	19.47%
Replay-300 v3	stored tree	300×1	7.25	$[4.19, 10.36]$	11.08%
Replay-300 v4	stored tree[†]	300×1	8.45	$[5.77, 11.22]$	4.89%
Online-120	fresh calls	120×3	6.67	$[1.11, 12.50]$	10.89%
Online-200	**fresh calls**	$\mathbf{200 \times 3}$	**5.00**	$[\mathbf{0.50}, \mathbf{9.33}]$	**2.67%**

[†]New task cohort. Evaluation still uses stored-tree replay, not fresh policy-specific online calls.

procedures in Appendix H. The 5% criterion compares the absolute mean-cost gap with StopRoute mean cost and is not an equivalence test. Estimated cost and wall-time differences are small, so no latency advantage is established. There are no operational failures.

6.2 Prospective replications and cost controls

Table 3 places the primary online result beside four earlier prospective cohorts, while Figure 2 shows the primary task-level effects and cohort intervals. All use new DataFlow-v2 task tables and are reported separately without pooling. The online studies collect three fresh repetitions per policy. Replay studies reuse stored completions. All five average gains are positive.

Replay-200 costs 63.1% less than Kimi-only, with an uncertain 4.0-point success advantage. Its advantage over random remains positive on 116 unseen semantic configurations. Appendices C, E, F, and G give the frozen controls, full tables, and configuration-cluster analyses.

6.3 Held-out operating point and selection sensitivity

Frozen held-out operating point. Before collection, we froze a 24-development/48-test split across 72 independently generated configurations. Development selection enumerates $9^3 = 729$ per-step single-model or ordered-cascade policies, maximizing success and breaking ties by cost. The same rule compares $3^3 = 27$ fixed routes with detector stopping. The retained policy succeeds on 42/48 test tasks versus 40/48 for its fixed route and 37/48 for Kimi-only. The two recoveries give exact McNemar $p = 0.5$, so this experiment motivates rather than establishes the primary claim. Its resampling analyses are exploratory (Appendix C).

Component and selection sensitivity. On the original 48 tasks, disabling stopping, removing fallback, and always falling back give $42/48$, $40/48$, and $37/48$ successes. Across 2,000 development resamples, 22 policies are selected and the original appears 34.6% of the time. Appendices L–M report the component, selection, and price sensitivities.

6.4 Legacy diagnostics and detector calibration

Legacy scope probes. Legacy probes cover retrieved-context HotpotQA (Yang et al., 2018), a synthetic tool chain, and DataFlow-v1/v2. In DataFlow-v1, adjacent-error association is high ($\phi = 0.94$) but retrospective routing saving is negligible, showing that association alone does not identify a routing opportunity. Legacy four-tier history-proxy witness has an empty feasible set, and final-step success after a wrong or missing table is nearly zero apart from one DeepSeek exception. Appendix J reports domains, definitions, and sensitivities. Appendix K reports the multi-tier analysis.

Detector calibration. Across held-out prefixes, detector failure precision is 1.00 and recall is 0.765/0.760/0.683 at steps 1–3. Accepted-output accuracy is 0.886/0.869/0.642. At step 3, rejection after a wrong or missing prior table is 0.863/0.679/0.615 for Kimi/DeepSeek/Qwen, versus zero after a correct history. Appendix K gives denominators, intervals, and proxy caveats.

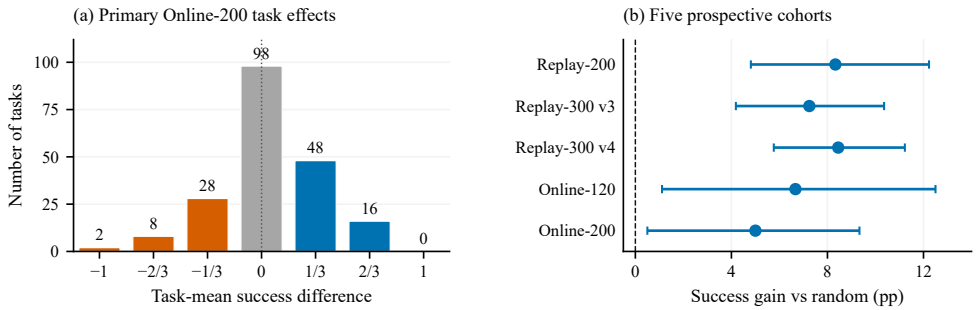

Figure 2: Primary task-level effects and prospective cohort evidence. **(a)** Online-200 task effects, computed after averaging each policy's three repetitions within task. Of 200 independent task units, 64 favor StopRoute, 98 tie, and 38 favor random triggering. The mean gain is 5.00 percentage points, with 95% task-bootstrap CI $[0.50, 9.33]$. **(b)** StopRoute-minus-random gains and 95% task-bootstrap intervals for the five prospective cohorts. All share the DataFlow-v2 task family, so we do not pool them. Table 3 reports execution modes, sample sizes, and exact cost gaps.

7 Discussion

Proposition 1 matches episode randomization: suffix optimality makes randomized open-loop routing exact, while an uninformative signal yields zero certificate. At success 0.85, Table 1 separates a 6.679 cost reduction from randomization and another 1.760 from state history. This matched-class result motivates randomized fallback placement. The multi-tier witness and reservation rule extend the analysis without claiming general router optimality.

The online study asks whether conditioning fallback placement on detected failure improves success when primary assignments are fixed and mean costs are nearby. Independent cost-only calibration and a held-out task-level test support that claim within DataFlow-v2. Because the control retains detector stopping, the comparison isolates conditioned fallback placement rather than all uses of state or the separate saving from heterogeneous model assignment.

The connection is mechanistic, not a fitted theorem test. The chain observes X_t before one of exactly K calls and resets after success. The workflow checks validity after paying for a response, can retry, and cannot repair an incorrect first table through a correct second aggregation. Detector outputs are therefore operational signals, not calibrated measurements of X_t or γ. The controlled model proves the separation. The online experiment tests its routing motivation under a different observation process.

8 Conclusion and Limitations

Suffix optimality reduces randomized open-loop routing to an exact convex program, exposing information value after randomization is matched. The controlled instance retains a 1.760 cost gap at target 0.85. In the primary online case study, cost-only calibration precedes testing on 200 held-out tasks with three fresh repetitions per policy. Conditioned fallback improves success by 5.00 points on average, with a 2.67% nominal-cost gap inside the prespecified tolerance. Information access and expenditure must therefore be compared separately: adding strong calls and placing them at observed failures are distinct sources of performance. Accordingly, evaluating an adaptive router requires first optimizing the information-blind policy class over model choices, call budgets, and randomization. Only the residual advantage should then be attributed to execution feedback, preventing stronger models or extra calls from being mistaken for informative routing.

The theory assumes a binary state, capability ordering, terminal reward, and fixed length. Intermediate credit can break suffix optimality (Appendix B.5). The API detector is post-call, unlike the free pre-action signal in Proposition 2. All five cohorts share one synthetic three-step family and cost gaps range from 2.67% to 19.47%. Frozen nominal prices are not invoices and do not establish cost equivalence.

Ethics Statement

Reducing expensive model calls can lower inference expenditure, but a cost-optimized router can also conceal quality degradation. Deployments should report outcome uncertainty with expenditure, audit semantic failures that pass format checks, and preserve a higher-quality fallback for consequential tasks. The study measures nominal API expenditure but does not estimate energy use.

AI Use Statement

Generative AI tools assisted with coding, editing, and reviewing. The authors conceived the original idea, built the research plan and theoretical framework, and performed the experiments and analysis. Statistical values were regenerated from the released records, and bibliographic metadata were checked against primary arXiv, OpenReview, or venue records. The authors reviewed all AI-assisted work and take responsibility for the final text, claims, code, and artifacts.

Reproducibility Statement

The supplement contains the recorded API traces, frozen task manifests, the v4 single-task recovery log, programmatic success checks, task-level statistical analysis, the fixed-count trigger permutation and graded degradation, relative-price sensitivities, independent audits, and scripts that reproduce every reported table and figure without new API calls. It also includes the SWE-bench dev diagnostic's frozen manifests, official scoring logs, and transition analysis, plus both online replications' frozen schedules, the second study's disjoint cost-only calibration, 2,640 test episode records, retry accounting, and task-level analyses. The complete proofs and executable theorem checks appear in the appendix. Appendix C records the model endpoints, collection protocol, token counts, prices, detector definitions, and split construction.

References

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. https://doi.org/10.52202/079017-4164.

Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.

Dylan Bouchard. Is escalation worth it? A decision-theoretic characterization of LLM cascades. *arXiv preprint arXiv:2605.06350*, 2026. https://doi.org/10.48550/arXiv.2605.06350.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research (TMLR)*, 2024. https://openreview.net/forum?id=cSimKw5p6R.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Matei Zaharia, James Zou, and Ion Stoica. Optimizing model selection for compound AI systems. *arXiv preprint arXiv:2502.14815*, 2025. https://doi.org/10.48550/arXiv.2502.14815.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 66305–66328, 2024. https://doi.org/10.52202/079017-2120.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 12987–13010, 2025. https://proceedings.mlr.press/v267/dekoninck25a.html.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations (ICLR)*, 2024.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A graph-based router for LLM selections. In *International Conference on Learning Representations (ICLR)*, 2025.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. In *International Conference on Learning Representations (ICLR)*, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world Github issues? In *International Conference on Learning Representations (ICLR)*, 2024. https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*, 2024. https://doi.org/10.48550/arXiv.2403.12031.

Shihong Huang, Shengjie Wang, Hong Ma, and Zhou Xu. RLCascadeRouter: Quality-estimator-free cascade routing via reinforcement learning. *arXiv preprint arXiv:2608.15817*, 2026. https://doi.org/10.48550/arXiv.2608.15817.

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. RouterEval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 3860–3887, 2025. https://doi.org/10.18653/v1/2025.findings-emnlp.208.

Vansh Kapoor, Aman Gupta, Hao Chen, Anurag Beniwal, Jing Huang, and Aviral Kumar. TRIM: Hybrid inference via targeted stepwise routing in multi-step reasoning tasks. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2601.10245. https://doi.org/10.48550/arXiv.2601.10245.

Sangmook Lee, Dohyung Kim, Hyukhun Koh, Nakyeong Yang, and Kyomin Jung. Confidence-guided stepwise model routing for cost-efficient reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(37):31483–31491, 2026. https://doi.org/10.1609/aaai.v40i37.40413.

Ji'an Lei and Jian Huang. TACIT-Switch: Cost-aware model escalation for LLM agents from censored supervision. *arXiv preprint arXiv:2608.27911*, 2026. https://doi.org/10.48550/arXiv.2608.27911.

Songyuan Li, Ahmed M. Abdelmoniem, and Shiqiang Wang. ProgRouter: Online progress-guided orchestration for multi-agent LLM workflows under quality-cost tradeoffs. *arXiv preprint arXiv:2608.25992*, 2026. https://doi.org/10.48550/arXiv.2608.25992.

Xinchen Liu, Hang Zhou, Yingjie Zong, Yuchuan Tian, Liuyang Song, Shuo Zhang, Yulong Li, Wei He, Mengyu Zheng, Runke Liu, Siyang Cheng, Xiang Kuang, Hailin Hu, Kai Han, and Yunhe Wang. Agentic routing: The harness-native data flywheel. *arXiv preprint arXiv:2607.11399*, 2026. https://doi.org/10.48550/arXiv.2607.11399.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations (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*, pages 23934–23949, 2025. https://doi.org/10.18653/v1/2025.findings-emnlp.1301.

Rudrendu Kumar Paul and Sourav Nandy. AgentRouter: Heterogeneous model routing for cost-optimal multi-step agentic workflows. *arXiv preprint arXiv:2609.22951*, 2026. https://doi.org/10.48550/arXiv.2609.22951.

Soham Shah and Kumar Shridhar. Select-then-Route: Taxonomy guided routing for LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 425–441, 2025. https://doi.org/10.18653/v1/2025.emnlp-industry.28.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021. https://openreview.net/forum?id=0IOX0YcCdTn.

Wenwen Si, Insup Lee, and Osbert Bastani. Policy-guided stepwise model routing for cost-effective reasoning. *arXiv preprint arXiv:2605.06116*, 2026. https://doi.org/10.48550/arXiv.2605.06116.

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. https://doi.org/10.48550/arXiv.2607.00053.

Matthew Streeter. Approximation algorithms for cascading prediction models. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 4752–4760, 2018. https://proceedings.mlr.press/v80/streeter18a.html.

Dimitris Stripelis, Zhaozhuo Xu, Zijian Hu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, and Chaoyang He. TensorOpera Router: A multi-model router for efficient LLM inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 452–462, 2024. https://doi.org/10.18653/v1/2024.emnlp-industry.34.

Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979. https://doi.org/10.2307/1910412.

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. https://doi.org/10.48550/arXiv.2605.18859.

Ruiyi Yang, Lihuan Li, Hao Xue, and Flora D. Salim. STAR: Failure-aware Markovian routing for multi-agent spatiotemporal reasoning. *arXiv preprint arXiv:2605.10057*, 2026. https://doi.org/10.48550/arXiv.2605.10057.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2369–2380, 2018. https://doi.org/10.18653/v1/D18-1259.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shenghao Ye, Yu Guo, Zhengheng Li, Shuangwu Chen, and Jian Yang. Rubric-guided process reward for stepwise model routing. *arXiv preprint arXiv:2605.29310*, 2026. https://doi.org/10.48550/arXiv.2605.29310.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. MasRouter: Learning to route LLMs for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15549–15572, 2025. https://doi.org/10.18653/v1/2025.acl-long.757.

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*, 2026. https://doi.org/10.48550/arXiv.2602.21227.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-R1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025. https://doi.org/10.52202/085713-4719.

A NOTATION

Symbol	Meaning
K	chain length (number of steps)
$X_t \in \{0,1\}$	latent trajectory state (0 = clean, 1 = corrupted)
γ	probability that a failed step makes $X_{t+1} = 1$
$p(m,x)$	success probability of model m in state x
$c(m)$	per-call cost of model m; c_w, c_s for weak/strong
$R \cdot \mathbf{1}\{S\}$	terminal reward; S = final step correct
s^*	target success probability
a_x, b_x	$p(w,x)$, $p(s,x)$ in the two-model case
v_t^w	clean-state occupancy at step t under the all-weak policy
g_0	$\Pr(S)$ of the all-weak schedule
g_1	$\Pr(S)$ of "strong last, weak elsewhere"
$\pi^{\dagger\prime}$, $P^{\dagger\prime}$	state-conditioned policy "strong at K iff $X_K = 1$" and its success
$V_{\text{marg}}, V^\star$	optimal cost over state-independent / state-conditioned policies
λ	shadow price of the soft (Lagrangian) budget
b_t	remaining budget at step t (hard-budget augmentation)
$G_t(x)$	expected value gain of the strong model before its extra cost
ϕ	adjacent-error Pearson correlation (descriptive, not γ)
O_K, r, f	pre-final-call corruption signal, sensitivity, false-positive rate
Z_t	audit-only proxy: an earlier required table is wrong or missing

Table 4: Notation used throughout.

B FULL PROOFS

B.1 PROOF OF LEMMA 1 (SUFFIX OPTIMALITY)

Setup and occupancy. Under any schedule $\boldsymbol{m} = (m_1, \ldots, m_K)$, let $v_t = \Pr(X_t = 0)$. Since success restores the clean state and a failure creates corruption with probability γ,

$$v_1 = 1, \qquad v_{t+1} = 1 - \gamma[(1 - p(m_t, 0))v_t + (1 - p(m_t, 1))(1 - v_t)]\,.$$

For a schedule ending in model m, terminal success is $p(m,0)v_K + p(m,1)(1 - v_K)$.

Exchange argument. Define the affine occupancy map $T_m(v) = \alpha_m + \beta_m v$, where $\alpha_m = 1 - \gamma(1 - p(m,1))$ and $\beta_m = \gamma[p(m,0) - p(m,1)]$. Let $d_0 = b_0 - a_0 \geq 0$, $d_1 = b_1 - a_1 \geq 0$, and $\alpha_w = 1 - \gamma(1 - a_1)$. Direct expansion gives, for every incoming occupancy v,

$$T_s(T_w(v)) - T_w(T_s(v)) = \gamma\,[\alpha_w d_0 + \gamma(1 - a_0)d_1] \geq 0.$$

Thus swapping an internal adjacent inversion (s,w) to (w,s) cannot reduce any later clean occupancy. If the pair occupies the final two steps, the same expansion applied to terminal success gives

$$\Pr(S \mid w,s) - \Pr(S \mid s,w) = \alpha_w d_0 + \gamma(1 - a_0)d_1 \geq 0.$$

Repeated adjacent swaps move all strong calls to a contiguous suffix without decreasing terminal success. □

Verification. Exhaustive comparison of all $\binom{K}{j}$ schedules against the suffix schedule over 60 random capability-ordered instances and every $j \in \{1, \ldots, K-1\}$ produces zero counterexamples (`verify_t1_fixed.py`, hypothesis H1).

B.2 PROOF OF THEOREM 1 (EXACT OPEN-LOOP COST AND SEPARATION BOUND)

(i) Exact value of V_{marg}. Cost is strictly increasing in the number of strong steps j. The all-weak schedule ($j = 0$) achieves $g_0 < s^*$ and is infeasible. By Lemma 1, the best schedule with $j = 1$ places the strong step last and achieves $g_1 \geq s^*$, hence feasible. Therefore the marginal optimum uses exactly one strong step, at the end: $V_{\text{marg}} = (K-1)c_w + c_s$.

(ii) The gap. The state-conditioned policy $\pi^{\dagger\prime}$ (weak throughout; strong at step K iff $X_K = 1$) succeeds with

$$P^{\dagger\prime} \;=\; a_0\, v_K^w + b_1\,(1 - v_K^w) \;\geq\; s^*$$

inside the window, hence is feasible for the state-conditioned problem. Its expected cost is

$$C^{\dagger\prime} \;=\; K\, c_w \;+\; (c_s - c_w)\,(1 - v_K^w),$$

since the strong model is called only when the final state is corrupted, an event of probability $1 - v_K^w$. Then $V^\star \leq C^{\dagger\prime}$, and

$$V_{\mathrm{marg}} - V^\star \;\geq\; (K-1)c_w + c_s - Kc_w - (c_s - c_w)(1 - v_K^w) \;=\; (c_s - c_w)\, v_K^w.$$

□

Why the window. The window $(g_0, \min(g_1, P^{\dagger\prime})]$ is exactly the region where the marginal class must buy one strong step while $\pi^{\dagger\prime}$ remains feasible: below g_0 all-weak is marginally optimal and the gap vanishes; above $\min(g_1, P^{\dagger\prime})$ the embedding witness is infeasible and the bound must be recomputed from the next feasible witness.

Verification. We enumerate all 2^K marginal schedules and use exact Pareto DP for deterministic $V^\star$; a Lagrangian sweep would recover only the convex hull and could miss deterministic feasible points. Across 60 random instances, hypothesis H2 has zero failures and the deterministic-class relative gap averages 77.4% (median 80.0%); hypothesis H3 (the exact V_{marg} formula) zero violations; $V^\star \leq V_{\mathrm{marg}}$ holds universally (`verify_t1_fixed.py`).

B.3 Proof of Proposition 1 (randomized open-loop optimum)

Let μ be any distribution over deterministic open-loop schedules. Group its support by the number j of strong calls and write q_j for the total probability assigned to group j. Every member of that group has cost C_j. By Lemma 1, its terminal success is at most G_j, so replacing the entire group by mass q_j on σ_j preserves expected cost and weakly increases success. Thus any feasible randomized open-loop policy induces a feasible point of the stated linear program with no greater cost. Conversely, every q in that program is itself an episode-level mixture over the suffix schedules and therefore belongs to $\Pi^{\mathrm{rand}}_{\mathrm{marg}}$. The two optimal values are equal, and the feasible mixtures trace the convex hull of the suffix points; minimizing cost above the target selects its lower envelope.

Finally, class inclusion gives $V^{\mathrm{rand}}_{\mathrm{state}} \leq V^{\mathrm{rand}}_{\mathrm{marg}}$. A feasible state-conditioned witness of cost $C^\dagger$ also gives $V^{\mathrm{rand}}_{\mathrm{state}} \leq C^\dagger$. Subtracting each upper bound from $V^{\mathrm{rand}}_{\mathrm{marg}}$ and taking the larger result proves the certificate. □

B.4 Proof of Theorem 2 (multi-tier certificate)

Under $\pi^\dagger(i)$, the clean occupancy follows

$$v_1^\dagger = 1, \qquad v_{t+1}^\dagger = 1 - \gamma(1 - p_{L,1}) + \gamma(p_{i,0} - p_{L,1})v_t^\dagger.$$

The terminal success and expected cost are therefore

$$P^\dagger(i) = p_{i,0}v_K^\dagger + p_{L,1}(1 - v_K^\dagger), \qquad C^\dagger(i) = \sum_{t=1}^{K}[c_i v_t^\dagger + c_L(1 - v_t^\dagger)].$$

For every $i \in F$, $\pi^\dagger(i)$ is feasible, so $V^\star \leq \min_{i \in F} C^\dagger(i)$. Because every open-loop policy is also a state-conditioned policy, $V^\star \leq V_{\mathrm{marg}}$. Taking the stronger of these two inequalities proves the stated certificate. □

Verification. The script `verify_t2.py` enumerates all L^K open-loop schedules, computes the state-policy Pareto frontier by exact finite-horizon DP, and compares the embedding recurrence with Monte Carlo. Across 50 random three-tier instances it reports zero formula failures and zero certificate-direction failures. The certificate is positive only when a feasible embedding is cheaper than the exact open-loop optimum.

B.5 Robustness to intermediate rewards

Our theory assumes terminal reward. To probe robustness under reward shaping, we mix terminal and per-step credit: $R_\eta = \eta \cdot \mathbf{1}\{S\} + (1-\eta) \cdot \frac{1}{K}\sum_t \mathbf{1}\{\text{step } t \text{ correct}\}$. Exhaustive schedule enumeration over 500 random instances per cell under capability ordering shows the suffix structure of Lemma 1 degrades as intermediate reward increases:

	$\eta = 1.0$	$\eta = 0.8$	$\eta = 0.5$	$\eta = 0.2$
$K = 3$	0.0%	0.0%	0.2%	1.0%
$K = 4$	0.0%	8.5%	19.3%	30.1%

Table 5: Fraction of (instance, j) pairs where the R_η-optimal schedule with j strong steps is *not* the contiguous suffix. The terminal-reward case ($\eta = 1$) is exact, as Lemma 1 guarantees; intermediate credit shifts value toward early strong steps, and the effect grows with K.

Suffix optimality is guaranteed here only at $\eta = 1$. At $K = 4$ it already fails for 8.5% of the tested (instance, j) pairs at $\eta = 0.8$. Intermediate credit can favor earlier strong calls; the present proof does not extend to a range of terminal-dominated mixtures. RoRo (Ye et al., 2026) uses process rewards as a training signal; such training shaping should also be distinguished from changing the terminal evaluation objective itself.

B.6 Proof of Theorem 3 (reservation-value rule)

Bellman equation. For the soft-budget objective with shadow price λ, the terminal value is $V_K(x) = \max_m\{Rp(m,x) - \lambda c(m)\}$. For $t < K$,

$$V_t(x) = \max_m \Big\{ -\lambda c(m) + [1 - \gamma(1 - p(m,x))]V_{t+1}(0) + \gamma(1-p(m,x))V_{t+1}(1) \Big\}.$$

At $t < K$, the strong-minus-weak action value is

$$\gamma(b_x - a_x)[V_{t+1}(0) - V_{t+1}(1)] - \lambda(c_s - c_w) = G_t(x) - \lambda(c_s - c_w).$$

At $t = K$ the same comparison is $R(b_x - a_x) - \lambda(c_s - c_w)$. This proves the reservation rule.

State monotonicity. The inequality $V_t(0) \geq V_t(1)$ follows by backward induction. At the terminal step, $p(m,0) \geq p(m,1)$ (Assumption 1) gives $V_K(0) \geq V_K(1)$. Inductive step: from a clean state, every action produces a weakly higher probability of the clean next state. The inductive hypothesis preserves the ordering through the max. □

Verification. Across 2,000 random parameter instances, `verify_t3.py` reports zero state-monotonicity failures and zero reservation-rule failures over 19,942 state–time comparisons.

B.7 Hard-budget scope

For a strict remaining budget b, the Bellman state is (t, x, b) and only actions with $c(m) \leq b$ are feasible. The continuation uses budget $b - c(m)$ and the same transition kernel as Definition 1. Consequently, Theorem 3's soft-budget comparison does not establish monotonicity in b. We leave structural analysis of the exact reachable-budget DP open and report no hard-budget threshold result.

C Experimental Details

C.1 Counterfactual harness

For each task we enumerate the assignment tree over the model pool (3 steps; 39 distinct prefix nodes for 3 tiers and 84 for 4 tiers), executing every node through the real provider APIs. Records are stored per node (`records_*.jsonl`) with tokens, cost, latency, and programmatic step-success labels. Offline policy evaluation walks recorded branches with zero additional API cost. The original 72-task record set contains 2,808 nodes; the new 200-task set contains 7,800. Legacy diagnostic sets retain repeated calls for sensitivity checks.

C.2 DataFlow task families

Each table contains `region`, `product`, `units`, `price`, `quarter`, `returned`. Legacy diagnostics share one fixed 600-row table. The original 72-task cohort and Replay-200 give each task an independently seeded 1,200-row table.

DataFlow-v1 (degenerate downstream). s_1: write one pandas expression computing per-region totals for a quarter (filtered to non-returned orders, optionally one product); s_2: read the resulting table and pick the argmax region; s_3: report that region's value. Steps s_2, s_3 are deliberately trivial given a correct table.

DataFlow-v2 (differentiated downstream). s_1/s_2: independent pandas expressions for quarters Q and Q' respectively; s_3: given both tables, compute per-region growth $(A - B)/B \times 100$ and report the argmax region and its growth. All three steps differentiate models.

For the original policy-selection study, 72 configurations are sampled without replacement from 12 ordered quarter pairs, two metrics, and seven product filters. The first 24 form the frozen development set and the remaining 48 the test set. Success is checked programmatically against answers computed from each task's table using absolute tolerance < 0.05 for table values and < 0.5 percentage points for the final growth value, together with the correct region.

C.3 Original held-out operating points

Table 6: Original DataFlow-v2 held-out results ($n = 48$). Success is a count, and costs are mean $\pm$ sample SD across tasks. StopRoute versus Kimi-only: 10 wins, 5 losses and 33 ties. Exact McNemar $p = 0.302$ and paired-bootstrap 95% success-difference CI $[-0.042, 0.250]$ do not establish superiority.

Policy	Success	Cost (10^{-3} USD)
Kimi-K3-only + stop	37/48	6.109 ± 2.410
mid-only + stop (DeepSeek)	33/48	0.2325 ± 0.09228
cheap-only + stop (Qwen-Flash)	3/48	0.07346 ± 0.1180
fixed route + stop (dev-selected)	40/48	1.907 ± 0.9357
StopRoute (dev-selected)	42/48	2.184 ± 1.037

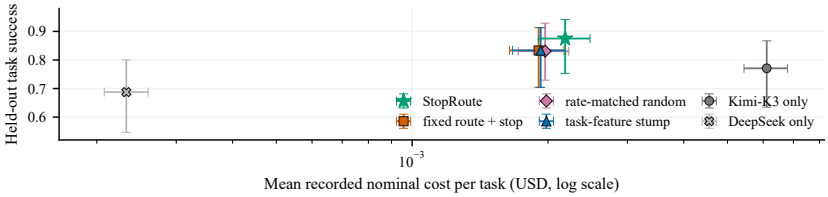

Figure 3: Original held-out operating points on 48 DataFlow-v2 tasks (log cost axis). Bars show 95% task-bootstrap cost and Wilson success intervals; random-control success uses task bootstrap.

C.4 Gold-free detector

s_1/s_2 (code steps): the EXPR line must execute against the task's table and yield a Series or a DataFrame whose selected values can be converted to floats. The recorded detector accepts whenever a table string is returned; it does not enforce the gold index set or numeric correctness. The latter checks belong to the evaluator only. s_3: the output must contain `REGION:` followed by a valid region and a `FINAL:` number. No gold answers are consulted by the detector. On held-out traces, $(\mathrm{TP}, \mathrm{FP}, \mathrm{FN}, \mathrm{TN})$ for detecting failure is $(39, 0, 12, 93)$, $(127, 0, 40, 265)$, and $(565, 0, 262, 469)$ at steps 1–3. Importantly, a detector-accepted but gold-incorrect intermediate is passed to the next prompt rather than replaced by the gold table.

C.5 Model pool and pricing

The primary pool is Kimi-K3, DeepSeek-Chat, and Qwen-Flash. The legacy four-tier diagnostic additionally includes Qwen-Plus, which is dominated by DeepSeek-Chat on that diagnostic sample. Token counts, latency, output text, programmatic success, and nominal cost are stored for every node.

C.6 Prospective follow-up and comparability

The 200-task manifest was fixed with seed 20270916 before follow-up calls. It contains every one of the 168 ordered-quarter/metric/product configurations once and 32 second instances with new 1,200-row tables; 84 tasks use a configuration seen in the original cohort and 116 use one of 96 previously unseen configurations. The separate 12-task pilot is excluded. Collection completed all $200 \times 39 = 7{,}800$ nodes, with zero failed tasks, eight HTTP retries, and two empty-response retries. Across the full tree, the three requested models have 2,600 nodes each. Their prompt/completion token totals are 509,491/905,640 (Kimi), 285,741/572,403 (DeepSeek), and 338,343/750,122 (Qwen). Summed full-tree nominal cost is $9.288; replayed policy costs in Table 7 are much lower because each policy visits only a few nodes per task.

The response model IDs were `kimi-k3`, `deepseek-flash`, and `qwen-flash`, respectively. Thus the requested `deepseek-chat` alias returned `deepseek-flash` in the new cohort; the historical collector did not retain served IDs. The new collector also checks generated pandas syntax and attributes before local execution. Applied to the original traces, this guard would reject 13 of 864 previously accepted code-step nodes. Exploratory old-tree replay with the new guard yields $40/48$ StopRoute, $38/48$ fixed-route, and $37/48$ Kimi-only successes, versus the original $42/48$, $40/48$, and $37/48$. This detector harmonization does not remove temporal API drift.

Table 7: Frozen-policy replay on 200 new DataFlow-v2 tasks. The random control's success count is an exact expectation. Costs use fixed nominal weights; latency sums successful-call durations under concurrent branch collection and is not deployment wall-clock time.

Policy	Success	Cost/task	Latency (s)	Calls
Kimi-only + stop	$145/200$	\$0.005679	44.68	2.810
Fixed route + stop	$136/200$	\$0.001626	26.44	2.670
Random trigger ($q = 1/24$)	$136.3/200$	\$0.001689	27.08	2.718
StopRoute	$\mathbf{153/200}$	\$0.002097	30.81	3.035

Table 8: State-independent trigger checks on the 200-task cohort. The first row is the frozen primary control; action- and nominal-cost-matched rates were computed after collection and are exploratory. Success values are exact expectations over the trigger, and intervals are paired task bootstraps for StopRoute minus control.

Control	Trigger q	Success	Cost/task	Success gap	95% CI
Frozen dev rate	0.0417	0.6817	\$0.001689	0.0833	$[0.0481, 0.1223]$
Observed action rate	0.1350	0.6854	\$0.001830	0.0796	$[0.0466, 0.1160]$
Matched nominal cost	0.3122	0.6925	\$0.002097	0.0725	$[0.0381, 0.1075]$

C.7 Reproducibility

Model endpoints and snapshot. Original calls were made on 2026-09-05–06 (UTC-7), and the independent follow-up was collected on 2026-09-16, through the providers' OpenAI-compatible endpoints: `kimi-k3` (`api.moonshot.cn/v1`, `reasoning_effort=low`), `deepseek-chat` and `deepseek-reasoner` (`api.deepseek.com/v1`), `qwen-flash` and `qwen-plus` (`dashscope.aliyuncs.com/compatible-mode/v1`). Generation used each client's defaults with `max_tokens` 4096 (8192 for `deepseek-reasoner`); temperature was left at provider default. The new traces store response model IDs and available fingerprints; the requested `deepseek-chat` returned `deepseek-flash`. The older traces did not store response IDs, so strict model-snapshot equivalence cannot be checked.

Cost accounting. Costs use fixed per-token accounting weights (USD per token, input/output): Kimi-K3 $2.0/8.0\times10^{-6}$, DeepSeek-Chat $0.27/1.10\times10^{-6}$, DeepSeek-Reasoner $0.55/2.19\times10^{-6}$, Qwen-Flash $0.05/0.40\times10^{-6}$, Qwen-Plus $0.40/1.20\times10^{-6}$. These weights were not independently verified as contemporaneous provider list prices and were held fixed for the follow-up, so dollar values are *nominal*. Raw prompt/completion token counts permit offline recomputation under any price table. The Replay-200 traces include the two empty-response retries in token and latency totals. The original stored-tree traces omit one such retry. Replay latency excludes HTTP-layer retry waiting time and local overhead. Online-120 and Online-200 retain retries in their call records, and their wall times include episode execution and retry waiting.

Policy selection. On 24 development tasks, StopRoute replays all $9^3 = 729$ combinations of three single-model choices and six ordered pairs per step. It maximizes observed task success and breaks ties by mean nominal cost. The fixed-route-with-stopping comparator is selected analogously over $3^3 = 27$ schedules, with detector rejection terminating execution. Table 6 evaluates these frozen policies on the original 48 test tasks; Table 7 evaluates the same assignments on 200 new tasks. Post-review analyses reusing the original tasks are exploratory (Appendix L.1). The new cohort does not reselect policies.

Theory verification. Lemma 1 and Theorem 1: all schedules and all non-trivial strong-call counts are enumerated for 60 random capability-ordered instances with $K \in \{2, 3, 4, 5\}$; zero suffix, bound-direction, or open-loop-cost failures. Theorem 2: across 50 random three-tier instances, 100 embedding formulas agree with Monte Carlo within tolerance; 27 instances have feasible certificates, 11 positive, with zero direction failures. Theorem 3: 2,000 random instances yield zero state-monotonicity failures and zero reservation-rule failures over 19,942 comparisons. All scripts and traces needed for these checks are included in the supplement.

D INTERACTIVE ALFWORLD TRANSFER CHECK

We evaluated a per-turn transfer analogue on the text-only ALFWorld environment (Shridhar et al., 2021). A salted SHA-256 rule first selected four games from each of six task types in the official `valid_unseen` split (24 games). After that study, we ran the same frozen four policies once each on all 110 remaining eligible games. The eligibility filter retains 134 solvable text games and excludes sliced and movable variants. The official environment `won` flag after at most 50 executed actions is the outcome. A separate `valid_seen` engineering pilot is excluded.

Every policy receives the same goal, current observation, six most recent action–feedback pairs, and the environment's admissible-command list. Responses must be a single JSON `action` from that list. DeepSeek self-retry and Kimi self-retry each get one same-model revision after a rejected candidate. The StopRoute analogue calls Kimi first after an exact unchanged observation and admissible list; otherwise it calls DeepSeek first and escalates to Kimi after a rejected response. The state-independent control selects Kimi first with probability $5/263$ and requests a same-state Kimi revision after DeepSeek with probability $5/258$, regardless of the state signal. These rates were frozen from the earlier `valid_seen` pilot. All second calls use the same generic revision prompt. The requested endpoints were `deepseek-chat` and `kimi-k3`; responses identified `deepseek-flash` and `kimi-k3`. Their completion ceilings were 256 and 1,024 tokens. The latter includes reasoning tokens when reported.

Table 9: Descriptive pooled view of 134 ALFWorld `valid_unseen` text games. Calls include same-state revisions; wall time is summed episode runtime divided by 134, not elapsed collection time.

Policy	Wins	Total calls	Kimi calls	Wall time/task (s)
DeepSeek self-retry	$85/134$	2,805	0	51.4
Kimi self-retry	$113/134$	2,405	2,405	175.2
StopRoute analogue	$97/134$	3,139	101	64.0
Random Kimi trigger	$91/134$	2,404	84	48.6

The first 24-game comparison gave StopRoute $17/24$ versus random $14/24$ (paired difference $+12.5$ points; 95% interval $[-4.2, 29.2]$; exact McNemar $p = 0.375$). The subsequent 110-game exten-

sion is the primary comparison for that cohort: StopRoute $80/110$ versus random $77/110$ ($+2.73$ points; paired task-bootstrap 95% interval $[-5.45, 10.91]$; exact McNemar $p = 0.664$). There are 12 StopRoute-only and nine random-only wins. StopRoute uses 5.95 more model calls per game (paired interval $[3.39, 8.67]$). Kimi self-retry wins $93/110$ and DeepSeek self-retry $69/110$. Pooling both cohorts descriptively gives StopRoute minus random $+4.48$ points (95% interval $[-2.99, 11.94]$; exact $p = 0.327$). The observed long-horizon advantage over random triggering is not reliable at this configuration, and StopRoute incurs more calls.

Across all 17 two-object placement games, Kimi wins 14 and StopRoute nine; six are won by Kimi alone and one by StopRoute alone. Four of StopRoute's eight failures in this category contain no Kimi call. The exact-stall and admissibility signals can miss semantic non-progress. In an exploratory development probe, two previously observed StopRoute prefixes were each continued three times per model from the same post-regression state: Kimi won four of six continuations and DeepSeek none. These are two selected states, not six independent games or a full-policy estimate.

A second exploratory probe froze all five previously unprobed two-object trajectories in the 110-game extension with an action-history semantic-regression event. From each identical post-event prefix, we ran three continuations per model. Kimi won $8/15$ and DeepSeek $4/15$, but the game-level advantage was concentrated in two originally failed games ($3/3$ versus $0/3$ in each). In another failed game DeepSeek won $1/3$ and Kimi $0/3$; the two originally successful games showed no Kimi gain. The five game-level differences average $+26.7$ points with a five-game bootstrap interval of $[-26.7, 80.0]$ points. These selected-prefix repeats show that some semantic regressions can be rescued, not that regression alone is a reliable escalation trigger or that a new full policy improves terminal success at matched cost.

As a task-type baseline diagnostic, we also form a post-hoc episode-level hybrid from the stored trajectories: use Kimi self-retry on two-object games and StopRoute on all other games. On the 110-game extension it wins $82/110$ versus StopRoute's $80/110$ (three hybrid-only and one StopRoute-only wins; exact McNemar $p = 0.625$), while raising Kimi calls from 81 to 414 and mean episode wall time from 65.8 to 84.5 seconds. The paired task-bootstrap interval for the $+1.82$-point difference is $[-1.82, 5.45]$ points. The type choice used observed results; the hybrid is an exploratory benchmark for a future, prospectively evaluated router.

The extension uses one stochastic rollout per policy and game. It tests a new per-turn heuristic with provided admissible commands, not the frozen three-step DataFlow assignment or an unrestricted code agent. Raw trajectories and frozen manifests are available in the research artifacts.

E Cost-calibrated DataFlow replication

We generated 50 calibration tables and 300 confirmatory tables before the first call in this replication. Table seeds do not overlap either the original study or the 200-task follow-up. The confirmation set covers all 168 semantic configurations, with 132 independently seeded repeats. The previously frozen StopRoute, fixed route, and Kimi-only policies retain their prompts, detector, model aliases, token limits, and stopping rules. The state-independent control always calls DeepSeek at step 1 and independently calls Kimi to replace that response with probability q, regardless of the detector output. Downstream calls follow the selected response. Every task records the complete 39-node model-assignment tree, allowing exact expectation over this single random decision while pairing policies on the same sampled responses.

Before the confirmatory calls, we used only the 50 calibration tasks' nominal costs to solve for the trigger probability that matches StopRoute's mean nominal cost on that cohort. This gave $q \approx 0.4177$ (calibration means: StopRoute \$0.002273; random $q = 0$ \$0.001516; random $q = 1$ \$0.003327). Calibration outcomes were not used for this choice or in the reported confidence intervals. We prespecified a 5% relative mean-cost tolerance on the confirmation set for an equal-cost interpretation; the denominator is StopRoute's mean nominal cost.

The prespecified StopRoute-minus-random success difference is 7.25 percentage points. A 20,000-draw paired task bootstrap gives a 95% interval $[4.19, 10.36]$ points; resampling the 168 semantic configurations gives $[3.93, 10.56]$. The paired nominal-cost difference is $-\$0.000235$ per task, with task-bootstrap interval $[-\$0.000356, -\$0.000109]$; the corresponding call-count difference is -0.122 per task, interval $[-0.199, -0.043]$. The 11.08% relative cost gap exceeds the prespecified

Table 10: Frozen-policy replay on 300 new DataFlow-v2 tables. The random control's success count is an exact expectation. Costs use the historical nominal weights, not provider invoices; latency sums call durations and is not online wall-clock time.

Policy	Success	Cost/task	Calls/task	Latency/task (s)
Kimi-only + stop	$222/300$	\$0.005749	2.760	25.57
Fixed route + stop	$206/300$	\$0.001648	2.677	12.39
Random trigger ($q \approx 0.4177$)	$212.3/300$	\$0.002354	3.129	16.14
StopRoute	$\mathbf{234/300}$	\$0.002119	3.007	15.07

5% tolerance, so this cohort confirms a frozen-q success advantage at lower observed nominal cost, not an equal-cost comparison. StopRoute exceeds the fixed route by 9.33 points ($[6.33, 12.67]$) and Kimi-only by 4.00 points ($[-2.00, 10.00]$); these are secondary comparisons.

As an exploratory sensitivity, selecting $q \approx 0.2787$ from the confirmation set's costs alone matches its mean nominal cost at \$0.002119. The associated random-trigger success expectation is 0.7006, versus 0.7800 for StopRoute, a 7.94-point difference. This post-hoc calculation does not replace the frozen-q analysis or satisfy the prespecified equal-cost gate. The full-tree collection made 11,700 calls and recorded 1,709,892 prompt tokens and 3,517,531 completion tokens. It encountered 16 HTTP retries and five empty-response retries, with no incomplete tasks. The served IDs were `kimi-k3`, `deepseek-flash`, and `qwen-flash`, each at 3,900 nodes. These are raw-tree expenditure totals; executing one policy would visit only its selected path.

F Fresh-cohort nominal-cost-calibrated DataFlow follow-up

After inspecting the first 300-task cohort, we chose to run a further test within the same synthetic DataFlow-v2 family. This decision was adaptive, but the new cohort's manifest, 39-branch collection protocol, four replay policies, analysis code, and random-trigger probability were frozen before its model calls. The probability used only the earlier cohort's nominal costs:

$$q = \frac{\bar{C}_{\mathrm{StopRoute},v3} - \bar{C}_{\mathrm{random},v3}(0)}{\bar{C}_{\mathrm{random},v3}(1) - \bar{C}_{\mathrm{random},v3}(0)} \approx 0.2787.$$

The three source costs were \$0.002119, \$0.001648, and \$0.003337 per task. No new-cohort outcome or cost was used to retune q. The 300 new 1,200-row tables use disjoint seeds and cover all 168 semantic configurations, with 132 independently seeded repeats. As before, the random control invokes the step-1 Kimi replacement with probability q independent of the detector; its reported success is the exact expectation over that decision on stored branch responses.

Table 11: Frozen-policy replay on the fresh 300-task cohort. Success for the random control is an expectation. Cost uses fixed nominal model weights; latency sums recorded call durations along the replayed path, not online policy wall-clock time.

Policy	Success	Cost/task	Calls/task	Latency/task (s)
Kimi-only + stop	$210/300$	\$0.005623	2.727	23.60
Fixed route + stop	$217/300$	\$0.001549	2.690	11.60
Random trigger ($q \approx 0.2787$)	$213.7/300$	\$0.002058	2.979	14.06
StopRoute	$\mathbf{239/300}$	\$0.001962	3.000	13.85

The primary paired StopRoute-minus-random success difference is 8.45 percentage points. Its 20,000-draw task-bootstrap 95% interval is $[5.77, 11.22]$ points; configuration-cluster resampling gives $[5.63, 11.27]$. The mean nominal-cost difference is $-\$0.000096$ per task (task-bootstrap interval $[-\$0.000213, \$0.000029]$). The absolute mean-cost gap is 4.89% of StopRoute cost, within the prespecified 5% point-estimate tolerance, though close to its boundary. Relative to the fixed route, StopRoute gains 7.33 points ($[4.67, 10.33]$) at \$0.000413 additional nominal cost per task. Relative to Kimi-only it gains 9.67 points ($[3.33, 16.33]$) and costs \$0.003661 less per task. These are descriptive secondary comparisons.

Post-hoc detector-placement control. StopRoute invokes the step-1 fallback on 39 of 300 tasks. Conditional on exactly 39 invocations, uniformly permuting those fallback labels across tasks gives an expected success of $215.4/300$ after rounding. StopRoute's advantage is 7.85 percentage points (task-bootstrap 95% interval $[5.29, 10.64]$; configuration-cluster interval $[5.29, 10.54]$). Summing the fixed-count combinatorial tail gives an exact one-sided $p = 7.82 \times 10^{-11}$; none of 100,000 Monte Carlo permutations attained the observed success. This analysis was designed after seeing the v4 result. It preserves fallback frequency, not resource use: the shuffled expectation has 2.825 calls and \$0.001787 nominal cost per task, versus 3.000 calls and \$0.001962 for StopRoute, because downstream early termination depends on where the labels are placed.

Progressive trigger-information degradation. We summarize the fixed-count placement contrast along an exact-expectation path on the same frozen tree, without new API calls. At each level we replace exactly s of the 39 observed detector-triggered locations by s detector-negative locations, preserving the fallback count. The final level, $s = 34$, is the nearest integer to the 33.93 expected replacements under a uniform fixed-count permutation. Table 12 reports exact expectations over placements; mutual information is between the observed and degraded binary trigger labels. Success decreases monotonically from 79.67% to 71.80% as this information approaches zero. The paired task-bootstrap interval for the endpoint difference is $[5.30, 10.66]$ percentage points.

Table 12: Post-hoc trigger-placement degradation on the frozen v4 tree. Each row uses exactly 39 step-1 fallbacks. Cost and calls vary because placement changes downstream early termination.

Replaced locations	MI (bits)	Success	Cost/task	Calls/task
0	0.5574	79.67%	\$0.001962	3.000
8	0.2903	77.82%	\$0.001921	2.959
17	0.1267	75.73%	\$0.001874	2.912
25	0.0387	73.88%	\$0.001833	2.871
34	0.0000	71.80%	\$0.001786	2.824

The straight line in replacement count is imposed by this construction. Let A be the 39 originally triggered tasks and B the other 261 tasks. Write $Y_i(1)$ and $Y_i(0)$ for the stored-tree success outcomes with and without the step-1 fallback, and $\Delta_i = Y_i(1) - Y_i(0)$. Uniformly removing s labels from A and adding s labels to B gives inclusion probabilities $1 - s/39$ and $s/261$, respectively. By linearity of expectation,

$$\mathbb{E}[\bar{Y}(s)] = \bar{Y}(0) + \frac{s}{300}\left(\bar{\Delta}_B - \bar{\Delta}_A\right).$$

Thus the measured slope and endpoint contrast quantify the placement advantage; the intermediate points are not independent replications or evidence for an empirical linear response law. The contrast is conditional on one sampled completion per stored branch and on the operational detector labels, rather than gold latent states. Exact fallback-count matching still allows total calls and costs to change. The endpoint differs slightly from the uniform-permutation expectation because it fixes the replacement count at 34 rather than averaging over all overlap counts. Intervals from random label placements describe placement variation, whereas paired task-bootstrap intervals describe task resampling; neither measures fresh-call variability.

Relative-price sensitivity. Offline price reweighting leaves every success outcome fixed. Table 13 multiplies both Kimi token weights by 0.5, 1, or 2 while retaining the frozen DeepSeek and Qwen weights and the original q. The 5% cost-window decision passes in the first two scenarios and misses in the Kimi-premium scenario. These multipliers are stress tests, not contemporaneous provider prices.

The full tree contains 11,700 calls, 1,734,971 prompt tokens, and 3,569,808 completion tokens, with 25 HTTP and three empty-response retries. The served IDs were `kimi-k3`, `deepseek-flash`, and `qwen-flash`, each at 3,900 nodes. A fingerprint field was present for DeepSeek only; neither a served ID nor that fingerprint establishes an immutable model snapshot. These totals describe collection of all branches, not the cost of running one policy. Nominal prices are not provider invoices.

The original collector completed 299 task trees and stopped on a `ValueError` while parsing one final numeric field for task `v4tes192`. After the collection process ended, we collected that task's

Table 13: V4 frozen-q relative-price sensitivity. The gap is the absolute mean-cost difference divided by StopRoute mean cost.

Kimi multiplier	StopRoute	Frozen-q random	Gap
$0.5\times$	\$0.001092	\$0.001133	3.76%
$1.0\times$	\$0.001962	\$0.002058	4.89%
$2.0\times$	\$0.003703	\$0.003909	5.55%

full 39-branch tree anew with a guard that treats malformed final numerals as incorrect rather than raising an exception. The 299 completed trees were preserved; the recovery and resulting file hashes were logged. On all recovered final-step records, the original parser agrees with the stored scores, so the guard did not change their evaluation. Excluding the recovered task yields an 8.38-point primary difference and a 4.85% mean-cost gap. Two Kimi nodes returned empty text even after one retry; both are Qwen-first branches that none of the four policies visit and were scored as failures. A separate audit checks the frozen source hashes, complete branch tree, response metadata, cost-only provenance of q, and independent policy replay. The study remains a within-family, one-completion-per-node test; it does not establish transfer to long-horizon code agents or fresh-call variability.

G PROSPECTIVE FRESH-CALL ONLINE REPLICATION

The stored-tree studies share one completion at each branch node across replayed policies. We therefore froze an online replication before making any new API calls. It contains 120 new 1,200-row tables with distinct semantic configurations and disjoint table seeds, four policies, and three fresh repetitions per task and policy, for 1,440 episodes. The policies are StopRoute, the v4 state-independent random trigger, the fixed route, and Kimi-only. A separate four-episode connectivity check is excluded. All 1,440 frozen episode keys completed; no episode was replaced or rerun.

The random policy inherits the v4 cost-only target $q \approx 0.2787$. Before calls, we assigned exactly 33, 33, and 34 of the 120 tasks to step-1 replacement in repetitions 1–3, respectively. Thus it triggers 100 times across 360 runs, independently of task state. Each random-trigger run first calls DeepSeek and, when assigned, also calls Kimi and uses Kimi's state. The task is the primary unit: the three binary outcomes are averaged within task before paired inference. The primary interval uses 20,000 task-bootstrap draws; the two-sided randomization test uses 100,000 task-level sign flips. Operational failures count as failures.

Table 14: Fresh-call online replication. Descriptive cells are mean $\pm$ SE across 120 task averages, each from three fresh episodes. SE is computed across task averages as the sample SD divided by $\sqrt{120}$. Costs use frozen token weights and wall time includes the complete episode. Agreement is the fraction of tasks with three identical binary outcomes.

Policy	Success (%)	Cost (10^{-3} USD)	Calls	Wall (s)	Agreement
StopRoute	78.61 ± 2.46	1.969 ± 0.0590	3.019 ± 0.0234	27.93 ± 0.392	56.7%
Random trigger	71.94 ± 2.73	2.183 ± 0.0713	3.019 ± 0.0397	29.02 ± 0.485	49.2%
Fixed route	70.56 ± 2.59	1.555 ± 0.0496	2.664 ± 0.0388	23.70 ± 0.422	42.5%
Kimi-only	70.00 ± 2.51	5.600 ± 0.1228	2.761 ± 0.0344	41.78 ± 0.624	39.2%

Table 15: Paired task-level comparisons in the online replication. Differences are StopRoute minus the comparator. Intervals are paired task-bootstrap 95% intervals. The p-value is a two-sided task-level sign-flip test.

Comparator	Success diff.	95% interval	Cost diff.	p
Random trigger	$+6.67$ pp	$[1.11, 12.50]$	$-\$0.000214$	0.0298
Fixed route	$+8.06$ pp	$[1.94, 14.17]$	$+\$0.000414$	0.0157
Kimi-only	$+8.61$ pp	$[2.22, 15.00]$	$-\$0.003631$	0.0120

Table 15 reports the paired gains. StopRoute and the random policy have the same mean call count, but StopRoute costs less (paired cost-difference 95% CI $[-\$0.000362, -\$0.000071]$). The random arm's cost is 10.89% higher relative to StopRoute, outside the prespecified 5% matching tolerance. This study supports a fresh-call success advantage, not a matched-cost claim. The subsequent two-stage study in Appendix H provides the independently calibrated fresh-call comparison within that nominal-cost tolerance.

The run records 4,127 calls, 616,683 prompt tokens, 801,387 completion tokens, five HTTP retries, no empty-response retries, and zero operational failures. The nominal total is \$4.071 under the frozen accounting weights, not a provider invoice. Served aliases are recorded for every call. The replication removes shared stored completions from the within-family policy comparison, while retaining the same synthetic task generator and provider endpoints.

H TWO-STAGE FRESH-CALL MATCHED-COST CONFIRMATION

The first online replication inherited the v4 trigger probability and exceeded the prespecified mean-cost tolerance. We therefore froze a two-stage confirmation before any calls. Stage 1 contains 40 new tables, three repetitions, and three policies: StopRoute, a state-independent trigger that never fires, and one that always fires. Only their nominal costs enter

$$q = \frac{\bar{C}_{\mathrm{StopRoute}} - \bar{C}_{q=0}}{\bar{C}_{q=1} - \bar{C}_{q=0}}.$$

The respective costs are \$0.001879, \$0.001406, and \$0.003458, giving $q \approx 0.2307$. Outcomes were not read for calibration. We then froze 200 disjoint test tables, all 1,200 task–repetition–policy keys, and exactly 46 random triggers in each repetition before any test call. The test covers every semantic configuration once and 32 independently seeded repeats.

Table 2 reports the held-out means and task-level standard deviations. Each policy has three new completions on each of 200 tasks (600 runs). The three binary outcomes agree on 57.5% of tasks for StopRoute and 48.5% for the random trigger.

The paired success comparison and its inference appear with Table 2. StopRoute's additional nominal cost is \$0.0000546 per run (paired task-bootstrap 95% interval $[-\$0.0000800, \$0.0001913]$), an absolute gap of 2.67% of StopRoute cost and therefore within the prespecified 5% mean-cost tolerance. This comparison does not establish cost equivalence. The mean wall-time difference is -0.37 seconds (paired task-bootstrap 95% interval $[-3.30, 2.60]$), and the call-count difference is 0.063 (paired task-bootstrap 95% interval $[-0.012, 0.140]$).

All 360 calibration and 1,200 test episodes completed without operational failure. The test records 3,612 calls, 505,668 prompt tokens, 715,512 completion tokens, 148 HTTP retries, no empty-response retries, and \$2.420 in frozen nominal accounting weights. Retries are retained in the records; no test episode was replaced or rerun. The study establishes a within-family fresh-call matched-cost result, not cross-domain transfer.

I CONTROLLED OBSERVATION EXPERIMENT AND POLICY-CLASS CHECKS

Table 1 compares the policy classes at $s^* = 0.85$. The unmixed perfect-state final-step witness achieves success 0.8875 at expected cost 4.123, and the deterministic state-history policy achieves success 0.8568 at expected cost 3.662. State history includes the current state and past actions.

Design. The simulation uses the parameters in Section 4.5, fixed seed 20270907, and 200,000 independently generated trajectories for each of six (r, f) settings. At each step a Bernoulli call outcome is drawn using the selected model's conditional probability; after a failure an independent Bernoulli draw with probability γ determines corruption. The final observation is sampled conditional on the incoming state before drawing the final call outcome. Costs count weak calls as 1 and strong calls as 10. The code evaluates every fixed schedule exactly and compares the noisy witness formula with simulated success and cost, using Monte Carlo standard errors for the latter comparison. The random algebra check independently sums over the four (X_K, O_K) cases.

Randomization changes the baseline. At target 0.85 the deterministic open-loop optimum costs 12. A mixture choosing the one-strong suffix schedule with probability $(0.85 - g_0)/(g_1 - g_0) \approx 0.2578$

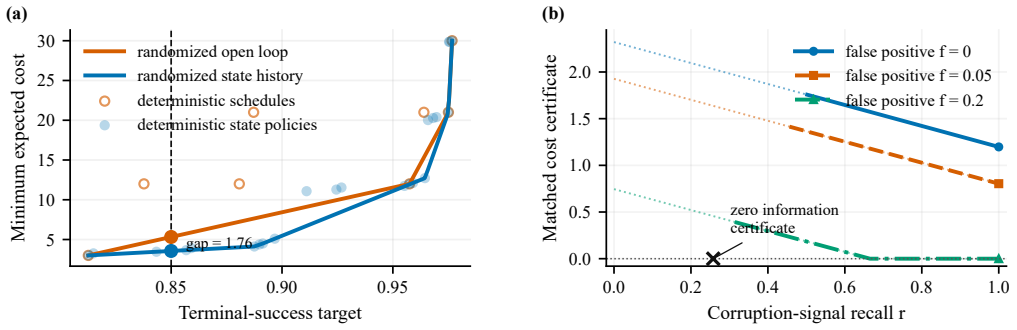

Figure 4: Full-range companion to Figure 1. **Left:** randomized cost frontiers and deterministic policy points over the full feasible target range. **Right:** solid portions are feasible noisy-signal certificates; dotted portions show the cost expression where the success constraint fails and are not valid certificates. The main figure omits those infeasible portions.

r	f	Exact success	Simulated success	Exact cost	Feasible at 0.85
1.0	0.00	0.8875	0.8878	4.123	yes
0.8	0.05	0.8760	0.8760	4.292	yes
0.5	0.05	0.8536	0.8532	3.955	yes
0.25	0.25	0.8500	0.8495	5.321	yes
0.0	0.00	0.8126	0.8113	3.000	no
1.0	1.00	0.9575	0.9574	12.00	yes

Table 16: Controlled noisy-observation witnesses, rounded to four significant digits. Feasibility uses unrounded exact model probabilities. These are synthetic model checks, not API performance estimates.

and all-weak otherwise costs 5.321. A linear program over all eight schedules verifies that this is the randomized open-loop optimum from Proposition 1. The perfect-state witness costs 4.123 and succeeds with probability 0.8875. Mixing it with the all-weak policy with probability approximately 0.4989 reaches the target exactly before rounding, at cost 3.560. Convexifying the full deterministic state-history frontier verifies that 3.560 is optimal after rounding. Using unrounded values, the matched information gap is approximately $5.321 - 3.560 = 1.760$.

Markov versus history-dependent control. Exhaustive enumeration of the 2^{2K} deterministic Markov action maps gives optimal cost 4.123 at success target 0.85 in this instance. The existing Pareto recursion gives 3.662 at success 0.8568. The latter permits distinct continuation plans after different histories reach the same current state: its feasible class is deterministic history-dependent policies. We therefore specify histories in Definition 2. The witness bounds hold also for the smaller Markov class because their witnesses are Markov. In contrast, the exact constrained optimum returned by the recursion should be labeled by its history-dependent class. The scalar soft-budget Bellman objective still admits a deterministic Markov optimum.

Limits of the observation result. Proposition 2 provides a feasible policy calculation under a calibrated pre-action signal, not a general POMDP solver. For a general observation channel, a belief $q_t = \Pr(X_t = 1 \mid H_t)$ summarizes the state uncertainty only when the assumed transition and emission model is valid. For example, after a binary signal with sensitivity r_t and false-positive rate f_t, a prior q is updated to $r_t q/[r_t q + f_t(1-q)]$ on a positive signal, when the denominator is nonzero. The API detector evaluates the current candidate after paying for its call and does not supply this channel for free. Estimating an action-dependent emission model and optimizing belief-state cascades are separate tasks from the present finite-family replay study.

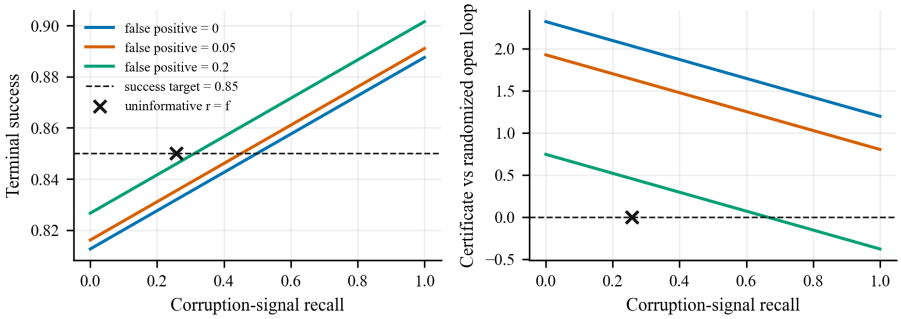

Figure 5: Exact success and matched cost certificate as signal recall changes. The cross marks the feasible uninformative control $r = f \approx 0.2578$, whose certificate against randomized open loop is zero. Certificate values apply only where success reaches the horizontal target.

Table 17: Four-domain diagnostics. ϕ: error association (step pairs). Δ_{cost}: oracle saving divided by marginal-heuristic cost at matched success, with three calls per assignment for ToolChain/DataFlow-v2 and one for the other domains; 95% task-bootstrap CI. These observational diagnostics do not test the quantitative window of Theorem 1.

Domain	Execution	ϕ	Δ_{cost}	95% CI	Match
HotpotQA (ctx)	real API	weak–mid	+13.2%	[0, 28]	✓
ToolChain (syn.)	real API	0.04/0.92	+33.9%	[16.7, 51.6]	✓
DataFlow-v1	real exec	0.94/0.94	≈ 0	–	✓
DataFlow-v2	real exec	0.08/0.59	+36.3%	[16.6, 58.0]	✓
+ 4th tier	real exec	n/a	+37.4%	[21.0, 55.9]	✓

J Definitions and audit of legacy diagnostics

Let e_t denote the binary error label of a recorded node. Adjacent pairs are joined by exact task ID and assignment-prefix identity within each replicate file. We compute $\phi = \mathrm{Corr}(e_t, e_{t+1})$ on these pairs, pooling the specified replicate files. Branches share tasks and prefix nodes, so this is a descriptive pooled correlation. Shared task difficulty, model allocation, and prompt features can produce association without causal transmission; ϕ is not an estimator of γ.

Matched-quality diagnostic. For task i, let $\widehat{p}_i(a)$ be final success averaged over recorded realizations of full assignment a. Let $\bar{c}_{it}(m)$ average recorded cost over all nodes with step t and model m, and define the surrogate path cost $\widetilde{C}_i(a) = \sum_t \bar{c}_{it}(a_t)$. The per-step score $q_{it}(m)$ averages $\widehat{p}_i(a)$ over full assignments having $a_t = m$. At threshold τ, the heuristic selects the cheapest model with $q_{it}(m) \geq \tau$; if none qualifies, it selects the highest score, breaking ties by cost. Write the resulting schedule as $a_i^{\mathrm{h}}(\tau)$. For $s_i^* = \max_a \widehat{p}_i(a)$, the hindsight oracle cost is $C_i^{\mathrm{or}} = \min_{a:\widehat{p}_i(a) \geq s_i^*} \widetilde{C}_i(a)$. Only when $\widehat{p}_i(a_i^{\mathrm{h}}) \geq s_i^*$ do we define

$$\Delta_i(\tau) = \frac{\widetilde{C}_i(a_i^{\mathrm{h}}(\tau)) - C_i^{\mathrm{or}}}{\widetilde{C}_i(a_i^{\mathrm{h}}(\tau))}.$$

Otherwise the task has a quality shortfall, not zero saving. The reported mean and task-bootstrap interval condition on matched tasks. At $\tau = 1$, all 16 tasks match in the three-repeat DataFlow-v2 diagnostic, yielding 36.3%. The statistic's denominator is heuristic cost; describing it as "36.3% excess cost relative to the oracle" would use the wrong denominator. The cost surrogate averages over different prefixes, whereas primary replay sums the costs of actually traversed nodes and configured retries.

Window reconstruction. The final-step witness chooses a strong model when either earlier all-weak prefix node failed its gold check. We reconstruct that decision separately within each replicate, then average within tasks and across tasks. This avoids deciding state by majority vote after merging repeats. All four proxy windows remain empty (Table 18). Because this irreversible history proxy

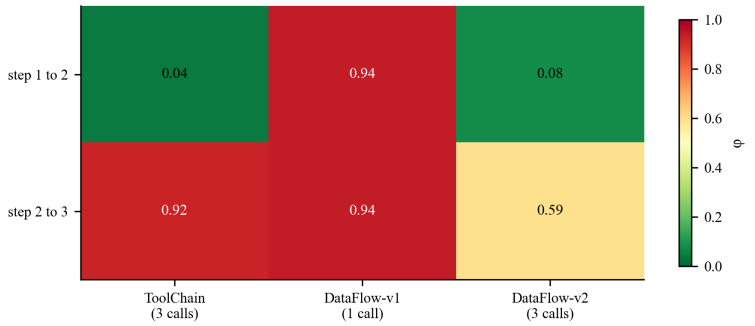

Figure 6: Adjacent-error association ϕ by domain and step pair. DataFlow-v1 has $\phi = 0.94$ and negligible retrospective routing saving. Association alone is insufficient to establish a saving or a causal mechanism.

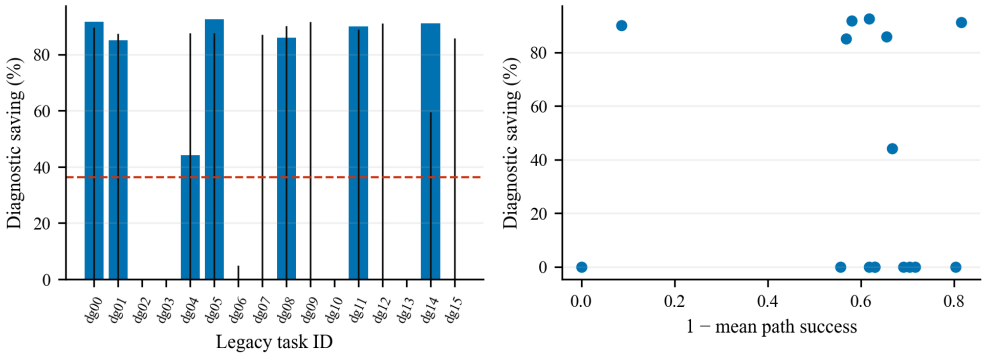

Figure 7: Legacy DataFlow-v2 diagnostic. Left: pooled three-repeat saving $\Delta_i(1)$ per task; black vertical segments show the range obtained by recomputing the entire diagnostic separately in each repeat. These ranges are not confidence intervals. Right: saving versus empirical difficulty $1 - \mathrm{mean}_a\, \widehat{p}_i(a)$, computed from the same traces. Task IDs index released records. The association is descriptive and shares data with the outcome used to construct the saving.

differs from the chain's recoverable state, it is an empirical analogue rather than a fitted theorem instance.

Domain	$\widehat{g}_0$	$\widehat{g}_1$	$\widehat{P}_Z$	Signed width
HotpotQA	0.7500	0.6875	0.7500	-0.0625
ToolChain	0.2917	0.2917	0.2917	0.0000
DataFlow-v1	1.000	1.000	1.000	0.000
DataFlow-v2	0.6250	0.8750	0.6250	0.0000

Table 18: Legacy history-proxy witness checks: 16 task clusters per domain. Each repeated trajectory is evaluated before averaging within tasks. A positive signed width is required for a non-empty empirical target window. These rates are not fitted homogeneous-chain parameters.

K Observation calibration on the recorded workflow

The proxy Z_2 is one if table A is wrong or absent. The proxy Z_3 is one if either A or B is wrong or absent. These are audit labels computed from prefix gold correctness; deployed routing never reads them. We estimate $\Pr(D_t = \text{reject} \mid Z_t = z, m_t = m)$ and $\Pr(Y_t = 1 \mid Z_t = z, m_t = m)$ from all appropriate prefix nodes in the 48 test tasks. D_t is a post-call detector output, while Z_t labels

incoming memory, so these are not the r, f of a pre-call observation channel. Each bootstrap draw resamples whole tasks, retaining all their branches.

Step	Model	Success $Z = 0$	Success $Z = 1$	Reject $Z = 0$	95% CI	Reject $Z = 1$	95% CI
2	Kimi	80/93	47/51	0.043	[0.010,0.088]	0.039	[0.000,0.098]
2	DeepSeek	82/93	45/51	0.086	[0.032,0.154]	0.078	[0.018,0.160]
2	Qwen	8/93	3/51	0.753	[0.634,0.863]	0.765	[0.635,0.880]
3	Kimi	170/170	0/262	0.000	[0.000,0.000]	0.863	[0.799,0.917]
3	DeepSeek	129/170	1/262	0.000	[0.000,0.000]	0.679	[0.612,0.748]
3	Qwen	169/170	0/262	0.000	[0.000,0.000]	0.615	[0.548,0.677]

Table 19: Conditional calibration by memory-corruption proxy, over 48 test-task clusters. Intervals resample whole tasks 10,000 times. Success counts use the same denominators as rejection rates. Boundary intervals such as [0,0] describe the empirical resampling distribution and do not certify zero population risk.

For final calls, Kimi succeeds on 170/170 clean-proxy inputs and 0/262 corrupted-proxy inputs; DeepSeek succeeds on 129/170 and 1/262; Qwen succeeds on 169/170 and 0/262. Conditional success intervals, including step 2, are released in `review_analysis.json`. Boundary bootstrap intervals are degenerate because no counterexamples were observed; they should not be read as population certainty. Step-2 success is roughly unchanged by earlier failure for DeepSeek (82/93 versus 45/51), consistent with its independent quarter-expression prompt. Moreover, Qwen's strong final-step performance and weak code-step performance show why a single homogeneous capability matrix cannot describe this workflow.

Identification boundary. A plug-in chain estimate would require a measured state matching Definition 1, identifiable transitions, and evidence for reset and time-homogeneity. Under Z_t, a successful step 2 can retain corrupted A, directly violating reset. Consequently we do not fit a causal γ from ϕ or from these proxy rates. A future causal test would intervene on the relevant intermediate table while holding task, downstream model, and remaining prompt fixed, repeat both clean and corrupted continuations, and separately test the reset assumption. The current data support conditional descriptions and policy replay, not that intervention.

L Exploratory replay comparisons and selection stability

All analyses in this appendix were added after examining the original test result and reviewer feedback. They use no new API calls. The original 24/48 split and selected policy are preserved, and the added results are exploratory. Table 20 includes: (i) a true fixed route executing all three steps, selected over 27 such routes on development data; (ii) the same fixed assignments with detector stopping; (iii) the selected StopRoute with stopping disabled, continuing on the collector's missing-table context after exhausted rejection; (iv) unconditional use of the selected cascade's configured fallback, retaining the stop rule; (v) a development-rate-matched random fallback trigger; (vi) each homogeneous ordered pair across all steps; and (vii) a small task-feature router. Configured retries use sibling nodes on the same prompt, so these are cross-model fallbacks, not fresh same-model retries.

Task-feature baseline. Candidate decision stumps split on revenue versus units or all-products versus filtered-products. A split is eligible only if each leaf has at least six development tasks. Within each leaf, choose the fixed route with stopping that maximizes development success, breaking ties by cost; choose the split by the same objective. The selected split is metric. This is a low-capacity task-feature baseline, not a reimplementation of an entropy router or a learned sequential policy. The stored records have no token log-probabilities or learned progress scores, and evaluating such policies would require their models and compatible traces.

Incremental effects. StopRoute versus fixed route with stopping has 2 wins, 0 losses and 46 ties (exact McNemar $p = 0.5$). The recorded cost difference is $(0.2766 \pm 1.110) \times 10^{-3}$ USD, with sample SD across 48 paired task differences. The selected cascade's only fallback is at step 1. Replacing its detector decision by an independent trigger with the development rejection probability $1/24$ gives expected success 0.831 and cost \$0.001972. Relative to this randomized control, StopRoute's paired expected-success difference is 0.0443 with task-bootstrap CI $[0.0017, 0.1059]$. Its

Policy	Success	Cost (10^{-3} USD)
Random escalation (dev-rate matched)	39.88/48 expected	1.972 ± 0.9009
Fixed route, always run	40/48	1.991 ± 0.8250
StopRoute, no early stop	42/48	2.186 ± 1.039
Always use configured fallback	37/48	3.453 ± 1.410
Homogeneous pair (dev-selected)	40/48	6.495 ± 2.160
$D \to K$ at each step	38/48	0.6225 ± 0.9184
$Q \to K$ at each step	29/48	3.494 ± 2.954
$Q \to D$ at each step	29/48	0.5637 ± 0.5735
Task-feature stump	40/48	1.927 ± 0.9178
Kimi, always run	37/48	6.515 ± 2.217

Table 20: Exploratory replay ablations. Costs are mean $\pm$ sample SD across 48 tasks. The random row summarizes each task's exact expected cost under an independent development-calibrated trigger, not realized random runs. The original StopRoute, fixed-route-with-stopping and Kimi-with-stopping baselines appear in Table 6. K, D, Q denote Kimi-K3, DeepSeek-Chat and Qwen-Flash. Paired inference remains in the released JSON.

cost difference is $(0.2122 \pm 1.079) \times 10^{-3}$ USD (mean $\pm$ sample SD across paired tasks). This expectation integrates over the random trigger exactly, not another set of model calls. StopRoute versus the task-feature stump has 3 wins, 1 loss, and 44 ties ($p = 0.625$). Stopping alone saves 4.2% of cost for the selected fixed route while preserving its observed 40 successes. For the selected cascade, disabling stopping changes recorded cost by only \$0.00000195 per task and does not change its outcomes. Thus model heterogeneity and conditional fallback, rather than early termination of that particular cascade, explain its measured operating point. These post-review comparisons have no multiplicity-adjusted superiority claim.

L.1 Selection resampling and common development constraints

Using seed 20270906, draw 2,000 bootstrap samples of 24 development tasks, repeat the full 729-policy selection in each, and evaluate the resulting policies on the unchanged 48 test records. Ties use original enumeration order after success and cost. The original policy is selected 691/2000 times; the next most frequent policy (451/2000) uses DeepSeek→Kimi at both code steps and Qwen at the final step. It achieves 45/48 on these test records at \$0.000608. This exploratory observation exposes selection instability; it does not replace the original frozen result.

The central 95% ranges across selected policies are $[0.6875, 0.9375]$ for success and $[0.000442, 0.002184]$ for recorded nominal cost. Resampling the test tasks independently after each development selection yields joint empirical percentile ranges $[0.7083, 0.9792]$ and $[0.000379, 0.002405]$. These are sensitivities conditional on the observed task pools and responses, not validated confidence bounds for a new training pipeline or simultaneous confidence bands over a frontier.

For each target in Table 21, select the cheapest policy whose development success meets the target. The fixed-route family is nested in the cascade family. The thresholds are shared between families, but achieved test success need not match, and the targets were introduced after review. For example, the 0.90 target yields only 0.875 test success for the cascade and 0.833 for the fixed route. The primary lexicographic objective differs from this constrained objective, which explains why the primary policy need not be the cheapest observed high-success test policy.

Development mean-cost budgets \$0.0005, \$0.001, \$0.002, and \$0.004 give test success 0.875, 0.9375, 0.875, and 0.875, at respective test mean costs \$0.000472, \$0.000608, \$0.002184, and \$0.002184. These are mean-cost constraints fitted on development data, not pathwise hard budgets. The non-monotonic test results show why a larger independent validation set is needed before selecting a production operating point. A quality-matched population cost comparison would require a specified success target or non-inferiority margin and adequate independent evaluation; this artifact does not establish one.

Dev target	SR dev	SR test	SR cost	Fixed dev	Fixed test	Fixed cost
0.70	0.792	0.812	0.305	0.792	0.812	0.305
0.80	0.833	0.875	0.472	0.833	0.812	1.826
0.90	0.917	0.875	0.442	0.958	0.833	1.907
0.95	0.958	0.938	0.608	0.958	0.833	1.907

Table 21: Minimum-development-cost policies at a common development success target. SR is the cascade family; fixed denotes fixed routes with stopping. Costs are test means in 10^{-3} nominal USD. Targets were specified during this post-review analysis; they are not preregistered or guaranteed on the test distribution.

M RECORDED COSTS, UNCERTAINTY, AND PRICE SENSITIVITY

For task i and policy π, sum prompt and completion counts for every visited model call, including rejected candidates that trigger a configured fallback. For provider-specific prices (u_m, v_m) the reweighted cost is

$$C_i(\pi; u, v) = \sum_m [u_m N_{im}^{\text{in}}(\pi) + v_m N_{im}^{\text{out}}(\pi)].$$

These sums include response-length differences and downstream input-length differences along the selected recorded branch. Mean tokens appear in Table 22. Multiplying each provider's two nominal weights by 0.5, 1, or 2 produces 27 scenarios, all stored in `review_analysis.json`. The policies remain fixed during reweighting.

Policy	Kimi input/output	DeepSeek input/output	Qwen input/output
StopRoute	178.3/197.3	87.9/49.5	161.1/407.0
Fixed route + stop	157.4/169.0	87.9/49.5	153.1/387.4
Kimi + stop	537.7/629.2	0.0/0.0	0.0/0.0

Table 22: Mean recorded prompt/completion tokens per test task, including configured fallback calls. These counts permit price reweighting but exclude any unlogged empty-response retry.

Uncertainty. Mean-cost and paired cost-difference intervals use 20,000 task bootstrap samples with seed 20270906. Each sampled task retains both policies' outcomes and all shared calls. The binary success-difference bootstrap distribution is computed exactly by convolving the empirical three-point distribution on $\{-1, 0, 1\}$ 48 times, avoiding Monte Carlo quantile jitter. Its percentile interval against Kimi-K3-only with stopping is $[-2/48, 12/48] = [-0.0417, 0.2500]$; McNemar's test uses the 15 discordant pairs. Wilson intervals describe individual task-success proportions and their overlap is not used as a paired significance test.