Beyond Tier Labels: Role- and Deployment-Dependent Model Substitution in Multi-Call LLM Workflows

Renxiang Wang[1] Jiaming Cui[1]
[1]Virginia Tech, Blacksburg, VA
{renxiangwang, jiamingcui}@vt.edu

Large multi-call LLM systems pose a scientific problem that query-level routing does not capture: the value of a model depends on where it enters a dependent computation and on the deployment that surrounds that call. Existing routers typically decide *where* to spend a stronger model while treating the benefit of the substitution itself as known. We separate these two decisions through a predicate-action factorization and evaluate it in controlled solve-merge-verify workflows spanning 8-64 solve calls and four three-tier model ladders. The resulting evidence reveals a consistent principle beneath apparently conflicting outcomes. On numeric frequency counting, all-strong reduces RMSE from 4.818 to 1.538 in the Mixed Qwen/GPT ladder, whereas the average Qwen-only ordering reverses. Input-matched interventions further show that the same medium-to-strong action has sharply different value across roles and scales. A semantic task-and-contract shift reverses the Mixed ordering again, while allocation ablations distinguish useful sparse placement from under-coverage and indiscriminate escalation. Together, these results establish model substitution as a deployment-conditioned action rather than a property implied by a tier label, and they provide a practical sequence for large-scale workflow routing: calibrate the action, resolve its role-conditioned effect, and then optimize its placement.

1 Introduction

Large-scale LLM agent systems distribute a task across role-specialized calls, enabling a single workflow to combine debate, software production, and graph-shaped collaboration [17, 13, 25]. Scaling, however, changes not only the computational budget but also the scientific object. Each additional call produces an intermediate representation that downstream calls must interpret, reconcile, or verify. Consequently, system quality is mediated by the surrounding graph. This dependence helps explain why adding agents alone does not guarantee a better answer [9, 18, 26], even though aggregation can improve final outputs [31, 15]. To make these gains practical, resource-aware methods reduce the associated cost by pruning agents, communication, or budgeted interactions [4, 32, 37]. Yet a more fundamental scientific question remains unresolved: what constitutes a beneficial model intervention inside a dependent workflow?

Query-level routers and cascades estimate model utility before deciding whether to defer an input to a more expensive model [23, 2, 5]. This abstraction is effective when requests can be evaluated independently, and recent work has explored non-parametric prediction, robustness analysis, and reasoning-aware selection of it[19, 16, 36]. However, a graph-internal call poses a distinct identification problem: its input is generated upstream, while its output becomes context downstream. The same nominal substitution may therefore operate on different prompts, under different contracts, and with different consequences as the graph, model pool, or task changes. In this setting, model utility is not adequately characterized by a model-task pair. It is the measured value of a named action applied to a particular node context.

To address this gap, we formalize the problem through a predicate-action factorization (Figure 1). A *risk predicate* identifies a context that warrants intervention, whereas a *substitution action* specifies

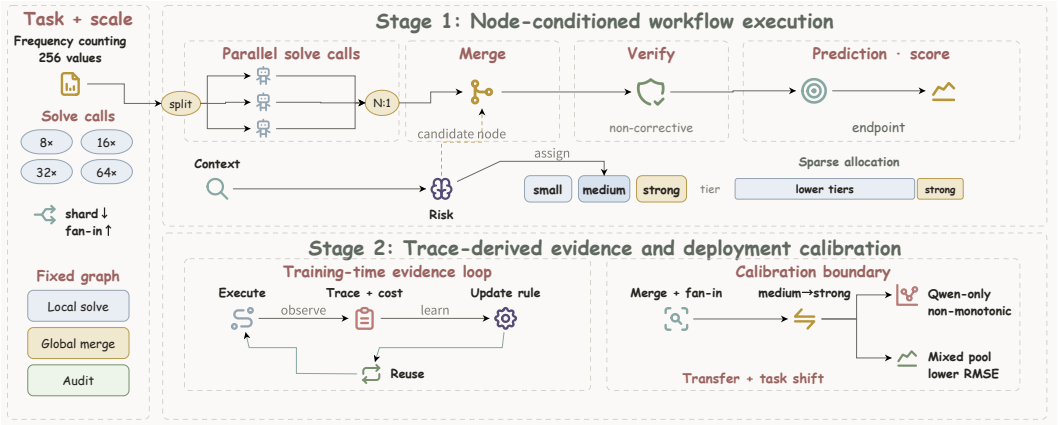

Figure 1. Overview of the proposed work. Stage 1 executes a fixed solve-merge-verify graph, while a risk predicate proposes node-level tier assignments. Stage 2 converts outcome traces into reusable constraints and then tests whether a substitution remains beneficial for the same candidate context after the deployment condition changes.

the model change applied there. The link between them is learned from outcomes rather than inherited from tier names. Under a fixed deployment, the resulting *node-conditioned substitution value* measures the change in loss caused by that action. This factorization yields an identifiable experimental sequence: fixed-tier sweeps calibrate the available actions, input-matched interventions localize where their effects arise, and allocation experiments determine whether a policy exploits those effects efficiently.

We study this factorization in a controlled workflow comprising repeated solve calls, a consolidation stage, and a verifier. Four model ladders expose favourable, non-monotonic, and task-reversed substitutions while holding the graph skeleton fixed. Matched interventions identify role-specific effects, allocation ablations separate coverage from selectivity, and frozen transfer reveals which effects persist when the model pool changes. A two-hop semantic task then changes the content and output contract while preserving the topology. EvoCap provides the trace-to-constraint mechanism connecting these experiments: it converts observed failures and resource use into auditable lower- and upper-tier rules whose placement can be inspected directly.

This work establishes a diagnostic foundation for workflow-internal routing. It operationalizes node-conditioned substitution value, demonstrates across four ladders that nominal upgrades do not define a portable capability order, and connects allocation footprints to outcome-matched evidence. The central lesson is both scientific and operational: tier labels describe an implementation choice, whereas measured substitutions reveal which actions a deployment can use. By making intermediate targets and call accounting explicit, the controlled workflow turns this distinction into a testable basis for routing at scale [22, 35].

2 Related Work

Routing, cascading, and composition. Query-level routers select which model should answer an input, while cascades determine whether a low-cost response should be accepted or deferred [23, 2, 14]. Recent variants learn query-model interactions, impose probabilistic cost controls, or purchase a small amount of strong-model guidance rather than a complete answer [24, 30, 7]. Robustness and data efficiency have also emerged as distinct concerns: routing decisions can be fragile under perturbations, centralized evaluation can be costly, and adversarial manipulation introduces a separate failure surface [16, 1, 41]. Taken together, these studies establish that routing quality depends on deployment evidence. Our question begins one level deeper: since an internal call consumes upstream outputs and shapes downstream context, request-level utility alone does not validate a graph-internal substitution.

Graph-structured and resource-aware agent systems. Multi-agent frameworks organize LLMs through roles, message passing, and programmable protocols [3, 34, 12]. Building on this foundation, resource-aware systems select teams, optimize communication topologies, or search over inference-time modules [21, 43, 28]. Recent routers choose collaboration modes alongside models, while context-aware orchestration and cascade-aware sidecars make the selection problem explicitly dependent on workflow state [38, 20, 6]. Other work treats topology, graph memory, and orchestration traces as learnable objects [39, 11, 40]. Planning-oriented architectures likewise expose downstream consolidation as an explicit system component [33] by identifying where computation may be valuable. We complement them with an outcome-level test of whether a model substitution is beneficial once attached to that context under its deployed prompt, schema, and role.

Conditional computation. Sparse experts and adaptive-depth models allocate computation conditionally within a jointly trained system [10, 8, 29]. Token-level depth allocation extends the same principle by varying computation within a sequence [27]. This analogy motivates the selective use of capability, but it does not establish an ordering over independently deployed API models, whose behaviour may vary across providers, output contracts, and endpoints. Agentic routing for coding and reasoning-aware model selection further broaden what a routing action can mean [42, 36]. We therefore measure substitution value in context rather than importing a monotonic hierarchy from tier names.

3 Study Design

3.1 Separating risk predicates from substitution value

Working vocabulary and running example. Our terminology distinguishes what a policy observes from the action it takes. A *node context* characterizes a call by its role, graph position, and measured features, while a *risk predicate* selects such contexts for possible intervention. The intervention itself is a *substitution action*, such as medium→strong, and its *node-conditioned substitution value* is the resulting change in loss under a fixed deployment. The pattern of roles and tiers selected throughout a workflow forms its *allocation footprint*. Thus, "high-fan-in merge" identifies a candidate context, whereas "replace medium with strong" states a testable claim about how to act in that context. Fixed-tier sweeps estimate the action globally, matched probes resolve its local output effect, and adaptive routing determines where the action is applied.

Let a workflow be a directed acyclic graph $G = (V, E)$. An allocation policy assigns each node v a tier $a_v \in \{s, m, h\}$, denoting small, medium, or strong. For task x, allocation $\mathbf{a}$ produces prediction $\hat{y}(x; G, \mathbf{a})$, loss $\ell(y, \hat{y})$, and recorded price $c(x; G, \mathbf{a})$. A conventional objective is

$$\mathbb{E}_{x\sim\mathcal{D}}\big[\ell\big(y, \hat{y}(x; G, \mathbf{a})\big) + \lambda c(x; G, \mathbf{a})\big]\,. \tag{1}$$

The complication is that setting $a_v = h$ invokes a named model through a particular prompt, output contract, and workflow role, rather than an abstract capability level. Fixed-tier sweeps estimate the global joint substitution

$$\bar{\Delta}_h^{\text{global}}(P, T, G, M) = \mathbb{E}[\ell \mid \mathbf{a} = \mathbf{m}] - \mathbb{E}[\ell \mid \mathbf{a} = \mathbf{h}], \tag{2}$$

where $\mathbf{m}$ and $\mathbf{h}$ assign medium and strong, respectively, to every workflow call. Here, P specifies the prompt and schema, T the task and metric, and (G, M) the graph and model pool. A positive $\bar{\Delta}_h^{\text{global}}$ therefore establishes that replacing medium with strong everywhere is favourable only under this measured condition. Separately, we estimate a local diagnostic effect for node context z,

$$\Delta_h^{\text{local}}(z) = \ell_z(y_z, \hat{y}_z^m) - \ell_z(y_z, \hat{y}_z^h), \tag{3}$$

by probing the same solve shard or merge input at both tiers. Because solve probes use exact shard-count targets and merge probes receive exact partial dictionaries, this matching removes upstream corruption from the comparison, isolates role-specific output error, and makes the action attributable to a workflow role.

The graph partitions a fixed input across $N \in \{8, 16, 32, 64\}$ solve calls before a single merge aggregates their local outputs. A verifier then observes only totals and validity fields, so it cannot independently recompute the answer or modify the scored merge output. Each logged scale therefore

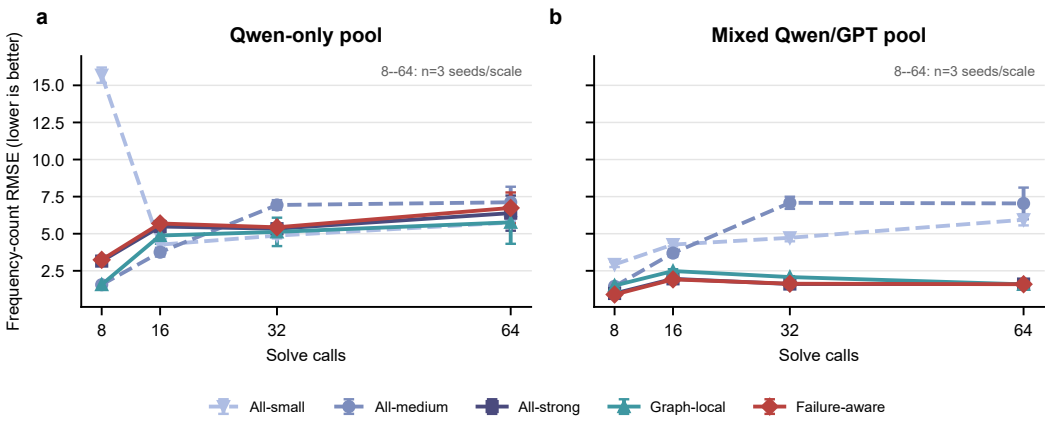

Figure 2. Primary ladders produce different workflow-level tier orderings. Lines summarize 8-64 solve calls with three seeds at every ladder-scale cell. Error bars are standard deviations across seeds. The comparison measures deployed workflow outcomes across the named model pools.

contains $N + 2$ calls. Because increasing N simultaneously reduces nominal shard size and increases merge fan-in, the scale sweep serves as an empirical stress axis rather than an isolated intervention on agent count.

3.2 Tasks, model ladders, and execution controls

The primary task requires the workflow to recover the full frequency vector of a 256-value integer array. Solve nodes process disjoint shards, the merge node sums their partial dictionaries, and final RMSE is computed against the exact vector. As the number of solve calls increases from 8 to 64, shard size decreases from approximately 32 to 4 values, while merge fan-in increases. The semantic extension preserves the same solve-merge-verify structure but distributes person-city and city-code relations across shards, requiring two-hop composition and replacing numeric RMSE with exact match. It therefore changes the full task-contract pair rather than merely rephrasing the original prompt.

We evaluate four model ladders under a common execution protocol. The Qwen-only ladder uses `qwen-turbo`, `qwen-plus`, and `qwen-max`, while Mixed replaces the strong tier with `gpt-5.4-mini`, creating a matched action contrast. The homogeneous GPT-5.4 and Qwen3 ladders test whether the same ordering and sparse allocation recur within a single model family. Each ladder uses seeds 101-103 at all four scales, disjoint deterministic training and held-out sets, five acquisition iterations, 16 training tasks, and 32 test tasks. All temperature-zero requests require JSON-object output, use a 90-second timeout, and permit at most two retries.

3.2.1 Matched intervention protocol.

Workflow-level sweeps reveal the net value of a tier change, whereas matched interventions identify where that value enters the computation. We re-execute the same 48 held-out arrays under the two primary ladders at all four scales, probing each solve shard against its exact local target and each consolidation input against the exact full target. Generation-only task rows then combine same-tier upstream outputs with the corresponding downstream call. This protocol yields 35,712 node-tier executions and 1,152 task-tier results, with input pairing removing task variation from every local comparison.

3.2.2 Metrics and cost accounting.

RMSE and semantic exact match are averaged over held-out tasks within each seed, with error bars reporting standard deviations across the three seeds. Test and acquisition costs are reported separately, yielding lifecycle cost after K deployments as $C_K = C_{\text{test}} + C_{\text{train}}/K$. Strong-call share measures the fraction of test requests routed to the strong tier, rather than its share of dollar cost.

3.3 Diagnostic allocation instrument

EvoCap is a trace-to-constraint allocator designed to keep every routing decision attributable. Its base estimator combines normalized role, depth, fan-in, and task proxies, and then records outcome validity, runtime escalation, and resource use. The updater creates `at-least` constraints following observed under-allocation and `at-most` constraints following excessive resource use. Matching rules convert these observations into tier intervals, with confidence resolving conflicts in favour of the better-supported bound. This construction enables direct ablations of evidence acquisition, restrictive gating, and indiscriminate escalation. Here the proxy "entropy", "variance", and "disagreement" refer to deterministic construction features rather than model log-probabilities or repeatedly sampled answers. Every assignment is therefore attributable to either the base estimator or a stored constraint. Exact proxy formulas, risk weights, prompts, and rule semantics (see Supplement for details).

3.4 Evidence hierarchy

The evaluation is organized around three scientific questions rather than policy families. First, does the deployed model pool contain a favourable medium-to-strong action, and how does its value vary with scale? Second, at which workflow role does that action alter output error? Third, does a selective policy place the action where matched evidence indicates that it is useful, and does this relationship persist under a model-pool or task shift? Qwen-only and Mixed define the primary controlled contrast. The two homogeneous ladders, frozen transfer, and semantic extension then progressively test whether the resulting interpretation survives broader deployment changes.

Fixed-tier sweeps establish the global ordering before any selector is interpreted. Input-matched probes then resolve medium-to-strong effects against exact intermediate targets, after which policy ablations assess coverage, selectivity, and evidence acquisition. Frozen transfer and the semantic task are deliberately placed last: a stable allocation footprint is interpretable only after the action it carries has been measured on both sides of the deployment change. The main text reports the medium-to-strong contrasts that directly answer these questions, while the Supplement retains the remaining tier transitions and complete construction details.

4 Results

4.1 Primary pools break the nominal tier ordering

Before an allocation policy can be evaluated, the deployed ladder must contain an upward action worth allocating. We therefore begin with fixed-tier sweeps, which reveal the underlying ordering before selective placement can obscure it (Figure 2 and Table 1). Across 8-64 calls, Mixed records an RMSE of 4.818 for all-medium and 1.538 for all-strong, while all-small reaches 4.466. Strong execution therefore yields a large quality gain in this pool, with mean test price increasing from \$0.0242 to \$0.2533.

However, this favourable ordering is not implied by the nominal tiers. Qwen-only records an RMSE of 4.843 for all-medium, compared with 5.091 for all-strong and 7.643 for all-small. Moreover, the preferred tier changes with scale: medium is best at 8 calls, while small is best at 32 and 64. This shows that the deployed pool and workflow scale jointly induce the capability ordering on which a router must act. The completed 64-call cells also sharpen this dependence. In Mixed, all-strong records an RMSE of 1.608 ± 0.055, compared with 7.041 ± 1.067 for all-medium. At the same scale under Qwen-only, all-strong improves over all-medium, yet all-small remains lower than both. Model substitution is therefore a conditional intervention whose value emerges from the deployment in which it is applied.

4.2 Matched interventions reveal role-conditioned action value

The cross-ladder contrast establishes that substitution value depends on the deployment setting, but workflow-level averages do not identify where the effect enters the computation. We therefore pair single-draw executions on identical inputs (Tables 2 and **??**). In the Mixed ladder, medium-to-strong substitution reduces downstream aggregation error in all 192 contexts and improves 191 of 192

Table 1. Workflow-level tier orderings across pools and tasks. Values are means over 12 scale-seed cells. Strong advantage is $\mathrm{RMSE}_M - \mathrm{RMSE}_S$ for frequency counting and $\mathrm{EM}_S - \mathrm{EM}_M$ for semantic aggregation, so positive values favour strong. Prices are mean test-only API cost in USD.

Pool / task	Metric	Medium	Strong	Strong adv.	Med. price	Str. price
Mixed Qwen/GPT, frequency	RMSE ↓	4.818	1.538	$\mathbf{+3.281}$	\$0.024	\$0.253
Qwen-only, frequency	RMSE ↓	4.843	5.091	$\mathbf{-0.248}$	\$0.023	\$0.409
GPT-5.4, frequency	RMSE ↓	1.558	1.070	$\mathbf{+0.488}$	\$0.262	\$0.849
Qwen3, frequency	RMSE ↓	4.503	2.997	$\mathbf{+1.506}$	\$0.039	\$0.078
Mixed Qwen/GPT, semantic	EM ↑	0.466	0.156	$\mathbf{-0.310}$	\$0.030	\$0.249

Table 2. Role-conditioned matched diagnostics. Medium-to-strong interventions are paired on identical inputs and pooled over four scales. Improve, tie, and harm are defined against the exact local or full-task target.

Pool	Intervention	Paired inputs	Improve	Tie	Harm	Improve rate
Mixed Qwen/GPT	merge node	192	192	0	0	100.0%
Mixed Qwen/GPT	solve node	5,760	13	5,732	15	0.2%
Mixed Qwen/GPT	generation subgraph	192	191	0	1	99.5%
Qwen-only	merge node	192	126	0	66	65.6%
Qwen-only	solve node	5,760	9	5,539	212	0.2%
Qwen-only	generation subgraph	192	115	0	77	59.9%

generation-only task comparisons, whereas 5,732 of 5,760 solve pairs yield identical local RMSE. The workflow-level gain is therefore concentrated in a small number of consequential downstream contexts rather than distributed uniformly across calls. In Qwen-only, the effect remains role-conditioned but changes sign with scale: downstream substitution reduces RMSE in 126 of 192 contexts and increases it in the remaining 66. At eight calls, only 20.8% of generation-only tasks benefit, with a mean error reduction of -0.875; at 64 calls, 83.3% benefit and the reduction reaches 2.831, while solve-level effects remain small at larger scales.

Taken together, the fixed-tier and paired analyses provide a coherent account of substitution value. Workflow-level means establish whether an action is favourable in aggregate, exact-input interventions localize where its effect enters the computation, and the scale sweep reveals when that effect changes sign. Mixed is favourable from 8 calls onward, with its advantage strengthening as scale increases, whereas Qwen-only transitions from negative to positive between 8 and 16 calls. Role therefore identifies where the workflow is sensitive, while scale determines whether the available substitution converts that sensitivity into a gain. This calibrated action-value map also sharpens the interpretation of adaptive routing: a sparse policy is useful only if it preserves a benefit already demonstrated in the deployed workflow.

4.3 Favourable tier ordering creates sparse quality-price tradeoffs

With the favourable Mixed action established and localized, sparse allocation can be evaluated against a concrete empirical reference rather than an assumed tier hierarchy. Failure-aware allocation records an RMSE of 1.512 at a \$0.0662 test price while assigning 10.0% of calls to strong models (Supplementary Figure S1). By comparison, all-strong records an RMSE of 1.538 at \$0.2533, so targeted strong-model access reduces the observed test price by 73.9% while remaining in the same quality regime. Static cross-channel allocation achieves a similar RMSE but uses more strong calls, whereas evolved-full trades some accuracy for a lower deployment price. The resulting frontier connects the preceding analyses: calibration identifies an action with demonstrated value, and selective placement captures that value economically.

4.4 The same pool reverses ordering after a task change

The numeric frontier is compelling within its calibrated deployment, but quality-price efficiency alone does not establish portability. We therefore retain the Mixed pool and graph skeleton while

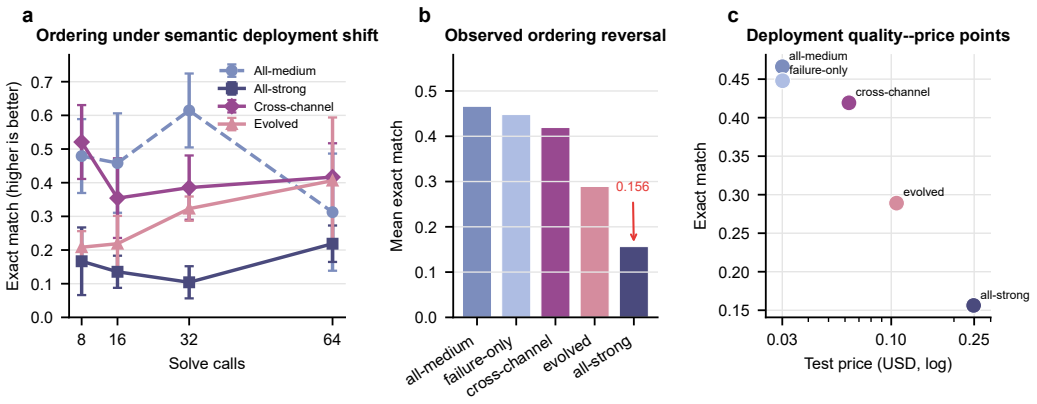

Figure 3. A complete task-and-contract shift reverses the Mixed workflow-level ordering. a, exact match by solve-call scale; error bars are standard deviations over three seeds. **b**, policy means across 12 scale-seed cells. **c**, quality-test-price points. Each held-out instance requires two-hop composition across separate evidence shards.

changing the task, prompt, and output contract. Across the 12 scale-seed cells, all-medium achieves an exact match of 0.466, whereas all-strong reaches 0.156 (Figure 3). Medium remains higher at every evaluated scale. The separation is largest at 32 calls, where exact match is 0.615 for medium and 0.104 for strong, with strong execution also incurring the higher price. The selective policies track this new ordering rather than preserving the one observed on the numeric task. Failure-only reaches 0.448, cross-channel 0.419, and evolved-full 0.289, whereas cross-evolved escalates every call and falls to 0.148. This reversal completes the progression from pool to role, scale, and task: even with the model pool and topology held fixed, a previously valuable substitution can become counterproductive when the task contract changes. Action calibration must therefore accompany each deployment rather than be treated as a one-time ranking attached to model names.

5 Ablation Analysis

5.1 Allocation interventions separate coverage, selectivity, and evidence

Once a favourable action has been identified, effective routing requires three separable capabilities: covering consequential contexts, selecting among those contexts, and acquiring sufficient evidence to revise future assignments. The primary factorial ablation isolates these capabilities (Figure 4). Graph-only allocation routes approximately 5.0% of calls to strong models but records RMSEs of 12.491 in Mixed and 14.199 in Qwen-only, showing that sparse allocation without adequate coverage misses consequential contexts. Uncertainty-only occupies the opposite regime, escalating approximately 90.0% of calls while recording RMSEs of 4.903 and 5.555. Although it covers most potentially useful contexts, it distributes strong-model capacity too broadly. Evidence acquisition exposes a third failure mode: hard cross-channel gating selects no strong calls and yields RMSEs of 13.674 in Mixed and 14.528 in Qwen-only, while the budget-validated policy likewise selects none because its admission rule blocks the observations needed to revise future assignments. At the high-escalation end, cross-evolved-full routes 81.1% of calls upward and achieves a Mixed RMSE of 3.847. Overall, these interventions distinguish under-coverage, over-allocation, and blocked exploration, demonstrating that escalation rate alone does not characterize routing quality.

These interventions complete the argument established by the fixed-tier and matched analyses. Action calibration determines what the deployment can use, role-conditioned interventions reveal where that action matters, and allocation ablations test whether the policy converts the resulting opportunity into an efficient operating point. Risk localization, substitution value, and effective placement are therefore distinct capabilities of a workflow router.

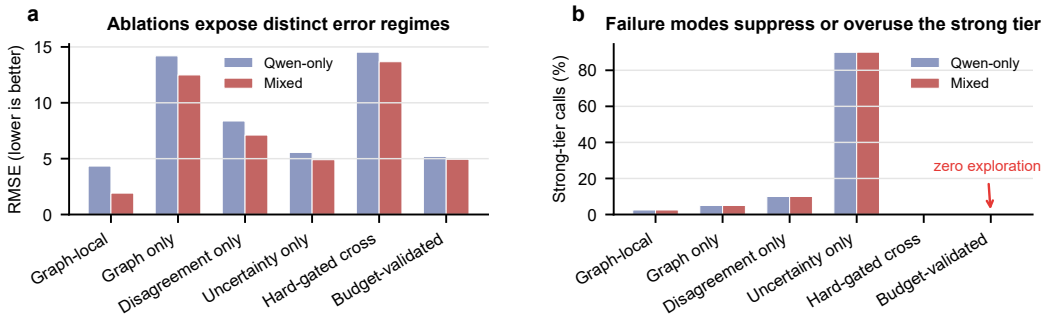

Figure 4. Allocation ablations expose distinct routing failures. **a**, mean RMSE for isolated signals, hard gates, and budget validation over 8-64 solve calls. **b**, strong-tier share separates under-coverage, over-allocation, and blocked exploration. Values average 12 scale-seed observations per primary ladder.

5.2 Homogeneous ladders reveal a scale-invariant downstream footprint

The two homogeneous ladders test whether the sparse operating point extends beyond the heterogeneous Mixed pool (Supplementary Figure S2). In GPT-5.4, RMSE improves from 1.558 at medium to 1.070 at strong, while in Qwen3 it improves from 4.503 to 2.997. Evolved allocation reaches RMSEs of 1.104 and 2.929, respectively, while using exactly two strong calls per task. Its mean test prices are $0.3608, compared with $0.8490 for all-strong in GPT-5.4, and $0.0468, compared with $0.0778 in Qwen3. This fixed allocation also reveals a scaling law: as the total number of calls increases from 10 to 66, the strong-call share declines from 20.0% to 3.0%, while the absolute capability budget remains unchanged. Sparse placement therefore recurs across model families whenever the ladder provides a favourable action, while the cost of strong-model access grows sublinearly with workflow size. Reporting both the number and share of strong calls is thus necessary to distinguish a genuine change in policy from a simple denominator effect.

5.3 Frozen transfer ablates target-side adaptation

Frozen transfer isolates what a stable allocation footprint carries across model pools (Supplementary Figure S3). On Mixed, the linear configuration records an RMSE of 1.829 after Qwen-only training and 1.870 after Mixed training; in the reverse direction, evaluation on Qwen-only yields 4.507 after Mixed training and 4.545 after Qwen-only training. Across all four cells, 5.0% of calls are routed upward and no rules are learned, indicating that the shared structural estimator produces a portable footprint across the two pools.

The rule-bearing cross-channel configuration exhibits an equally stable but substantially denser regime. On Mixed, it records an RMSE of 4.077 and 4.062 after in-pool and Qwen-only training. On Qwen-only, it records 5.069 and 5.293 after Mixed and Qwen-only training. Across all 4 cells, approximately 81.1% of calls are escalated and roughly 5 rules are retained. Frozen transfer therefore preserves the *shape* of the policy more readily than the value of its attached model action, reinforcing the modular view that structural predicates and model substitutions require separate calibration.

5.4 What the ablations establish

Taken together, the ablations recast workflow routing as a sequence of empirically separable decisions. Fixed-tier sweeps identify the actions available within a deployment, matched interventions resolve their role- and scale-conditioned value, and allocation tests determine whether a policy converts that value into an efficient footprint. In Mixed, all 192 downstream aggregation contexts improve, while 5,732 of 5,760 solve pairs tie. In Qwen-only, the same action improves 126 contexts and harms 66. This contrast explains why escalation share and rule count are meaningful only when interpreted relative to outcome-calibrated action value.

The factorization therefore supports a modular deployment strategy: structural predicates capture recurring risk contexts, whereas attached substitutions are recalibrated for the target pool, task, and scale. This separation preserves reusable workflow knowledge without treating a tier name as portable

evidence. Together, the four ladders, task shift, and frozen transfer establish this action-aware design as the natural unit for routing at scale.

5.5 Action-aware routing as a systems principle

The experiments support a broader design principle for multi-agent systems. A routing decision should be represented as the pair (p, a), where predicate p describes the context selected for intervention and action a names the deployed model substitution. Separating these objects allows a system to reuse structural knowledge while updating the component most exposed to deployment change. A predicate derived from graph position or observed failure may remain informative across model pools, while the attached action is selected from a target-specific calibration map rather than inherited from a global tier hierarchy.

This view also clarifies how routing should scale. The homogeneous ladders show that a constant number of strong calls can preserve the all-strong quality regime as the workflow expands from ten to 66 total calls, turning selective capability into a sublinear systems resource. The allocation ablations further show that this advantage depends on maintaining coverage, selectivity, and evidence acquisition together; removing any one produces a distinct and measurable failure regime. Action-aware routing therefore unifies quality control with resource allocation: calibration identifies the interventions worth purchasing, while the allocator concentrates that capability in the contexts where it changes workflow outcomes.

6 Conclusion

Large-scale workflow routing is not merely the placement of stronger models. It requires jointly identifying a consequential context and an action that improves outcomes within it. Across four ladders, the same nominal upgrade yields favourable, non-monotonic, and task-reversed effects, while matched interventions show that these effects are structured by role and scale. Allocation ablations further demonstrate that sparse access succeeds only when coverage, selectivity, and evidence acquisition align with this action-value structure. Together, these findings establish a practical foundation for multi-call routing: calibrate the deployed action, resolve its contextual value, and optimize placement against the resulting evidence. This sequence converts tier labels into measurable interventions and transforms allocation footprints from descriptive traces into interpretable deployment decisions. As multi-agent systems scale, action-aware calibration provides the missing link between more computation and reliably better outcomes.

References

[1] Baris Askin, Shivam Patel, Anupam Nayak, et al. Federate the router: Learning language model routers with sparse and decentralized evaluations. *arXiv preprint arXiv:2601.22318*, 2026.

[2] 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*, 2024.

[3] Weize Chen et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, 2024.

[4] Weize Chen, Jiarui Yuan, Chen Qian, et al. Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system. *arXiv preprint arXiv:2410.08115*, 2024.

[5] 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*, volume 267 of *PMLR*, pages 12987–13010, 2025.

[6] Davide Di Gioia. Cascade-aware multi-agent routing: Spatio-temporal sidecars and geometry-switching. *arXiv preprint arXiv:2603.17112*, 2026.

[7] Ziming Dong, Hardik Sharma, Evan O'Toole, et al. Pay for hints, not answers: LLM shepherding for cost-efficient inference. *arXiv preprint arXiv:2601.22132*, 2026.

[8] Nan Du et al. Glam: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *PMLR*, 2022.

[9] Yilun Du et al. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[10] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[11] Tao Feng, Haozhen Zhang, Zijie Lei, et al. GraphPlanner: Graph memory-augmented agentic routing for multi-agent LLMs. *arXiv preprint arXiv:2604.23626*, 2026.

[12] Dawei Gao, Zitao Li, Xuchen Pan, et al. AgentScope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*, 2024.

[13] Sirui Hong et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024.

[14] Qitian Hu et al. Routerbench: A benchmark for multi-llm routing systems. *arXiv preprint arXiv:2403.12031*, 2024.

[15] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

[16] Aly M. Kassem, Bernhard Schölkopf, and Zhijing Jin. How robust are router-LLMs? analysis of the fragility of LLM routing capabilities. *arXiv preprint arXiv:2504.07113*, 2025.

[17] Guohao Li et al. Camel: Communicative agents for "mind" exploration of large scale language model society. *Advances in Neural Information Processing Systems*, 36, 2023.

[18] Junyou Li et al. More agents is all you need. *Transactions on Machine Learning Research*, 2024.

[19] Yang Li. Rethinking predictive modeling for LLM routing: When simple kNN beats complex learned routers. *arXiv preprint arXiv:2505.12601*, 2025.

[20] Shanyv Liu, Xuyang Yuan, Tao Chen, et al. CASTER: Breaking the cost-performance barrier in multi-agent orchestration via context-aware strategy for task efficient routing. *arXiv preprint arXiv:2601.19793*, 2026.

[21] Zijun Liu, Yanzhe Zhang, Peng Li, et al. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024.

[22] Nature Machine Intelligence. Multi-agent ai systems need transparency. *Nature Machine Intelligence*, 8:1, 2026.

[23] Isaac Ong et al. Routellm: Learning to route llms with preference data. In *International Conference on Learning Representations*, 2025.

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

[25] Chen Qian et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 15174–15186, 2024.

[26] Chen Qian et al. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, 2025.

[27] David Raposo et al. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

[28] Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, et al. Archon: An architecture search framework for inference-time techniques. *arXiv preprint arXiv:2409.15254*, 2024.

[29] Tal Schuster et al. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[30] Antonios Valkanas, Soumyasundar Pal, Pavel Rumiantsev, et al. C3PO: Optimized large language model cascades with probabilistic cost constraints for reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[31] Junlin Wang et al. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, 2025.

[32] Zhenhailong Wang et al. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

[33] Taylor W. Webb, Suchol Mondal, and Ida Momennejad. A brain-inspired agentic architecture to improve planning with llms. *Nature Communications*, 16:8633, 2025.

[34] Qingyun Wu, Gagan Bansal, Jieyu Zhang, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

[35] Lin Xu et al. Magic: Investigation of large language model powered multi-agent in cognition, adaptability, rationality and collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

[36] Jiaqi Xue, Qian Lou, Jiarong Xing, and Heng Huang. R2-Router: A new paradigm for LLM routing with reasoning. *arXiv preprint arXiv:2602.02823*, 2026.

[37] Jianing Yang et al. Bamas: Structuring budget-aware multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, 2026.

[38] Yanwei Yue, Guibin Zhang, Boyang Liu, et al. MasRouter: Learning to route LLMs for multi-agent systems. *arXiv preprint arXiv:2502.11133*, 2025.

[39] Yifan Zeng, Yiran Wu, Yaolun Zhang, et al. When does multi-agent RL improve LLM workflows? workflow, scale, and policy-sharing tradeoffs. *arXiv preprint arXiv:2605.24202*, 2026.

[40] Chenchen Zhang. Reinforcement learning for LLM-based multi-agent systems through orchestration traces. *arXiv preprint arXiv:2605.02801*, 2026.

[41] Wenhui Zhang, Huiyu Xu, Zhibo Wang, et al. RerouteGuard: Understanding and mitigating adversarial risks for LLM routing. *arXiv preprint arXiv:2601.21380*, 2026.

[42] Pengfei Zhou, Zhiwei Tang, Yixing Ma, et al. Agent-as-a-router: Agentic model routing for coding tasks. *arXiv preprint arXiv:2606.22902*, 2026.

[43] Mingchen Zhuge et al. Gptswarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

A Supplementary Information

S1. Policy definitions and execution protocol

All policies share the same solve–merge–verify runner, task instances, retry logic, and accounting code within each ladder–task–scale–seed condition, so their differences arise from allocation rather than execution. Fixed policies assign one tier globally, graph-local combines role with graph position, and graph-only isolates structural features. Uncertainty-only combines the predefined entropy, variance, and disagreement proxies, while disagreement-only isolates one proxy; these construction features are deterministic and do not depend on model log-probabilities or repeated decoding.

Cross-channel policies combine structural and proxy evidence, with gate ablations isolating how each channel contributes to admission. Evolved-full converts traces into lower- and upper-tier constraints, while its directional ablations retain only one side of that update. Failure-aware evolution adds explicit outcome and price evidence. The remaining variants test whether the allocator can acquire evidence, use efficiency signals, and preserve its footprint when transferred to another pool. Semantic configurations keep the same policy families while replacing the task-specific prompts and contracts.

The balanced primary factorial contains 24 ladder–scale–seed conditions and 33 reported policy rows per condition after non-manuscript conditions are excluded and exact aliases are counted once. Homogeneous replication contains 24 complete conditions and 17 policy rows per condition. The matched diagnostic contains 24 ladder–scale–seed conditions and 11,904 node contexts. Three tier executions per context yield 35,712 node-tier probes; three transitions per context coincidentally yield the same number of paired rows. It also contains 1,152 task-tier results. The exported numeric tables contain 1,200 seed-level rows, 400 ladder–scale–policy summaries, and 100 analysis summaries. Transfer and semantic blocks contribute 240 and 96 seed-level rows. Every exported row retains timestamped run provenance.

S2. Task generation, splits, and graph construction

Each numeric task samples an array of 256 values and asks the workflow to recover the full frequency vector. Training and test collections are disjoint and deterministic; the test generator offsets the training seed by 10,000. Within a condition, every policy receives the same tasks and graph family. The graph partitions the array across solve calls, sends partial dictionaries to one merge, and then invokes a verifier. The verifier receives expected and observed totals, key count, and Boolean checks for distinct integer keys and non-negative integer frequencies. It does not receive the original array or full dictionary and does not modify the prediction. At 8, 16, 32, and 64 solve calls, nominal shards contain approximately 32, 16, 8, and 4 entries before remainder handling.

The five difficulty inputs are normalized construction features, not post-hoc estimates from model responses. For a solve node, entropy equals the fraction of distinct values relative to the attainable diversity, variance equals shard fraction multiplied by solve-call count, disagreement is $0.15 + 0.35$ times the diversity term, fan-in is zero, and depth is 0.2. For the merge, entropy and variance are 0.45, disagreement is $0.45 + \min(0.35, N/32)$, fan-in is $\min(1, N/8)$, and depth is 0.7. For the verifier, entropy and variance are 0.35, disagreement is $0.55 + \min(0.30, N/32)$, fan-in is 0.25, and depth is 0.9. The names "entropy", "variance", and "disagreement" describe heuristic channels in the simulator; they should not be interpreted as calibrated predictive uncertainty.

Every run uses five training iterations, 16 training tasks, and 32 held-out tasks. All four ladders use seeds 101, 102, and 103 at each of the 8, 16, 32, and 64 solve-call scales. The transfer and semantic experiments use the same three seeds at every reported scale. Exact directories, model identifiers, evidence roles, and finalization status are recorded in `source_data/data_manifest.json`.

The matched ladder diagnostic re-executes the first 16 arrays from the deterministic 32-task held-out stream for each seed. Array generation depends on seed but not on ladder or solve-call scale, so the same 48 underlying arrays are reused across both ladders and all four scales with a new shard partition at each scale. Each solve probe executes the same shard once at every tier and is scored against its exact local frequency dictionary; each downstream probe receives exact partial dictionaries and is scored against the full target. Task-tier execution then combines same-tier solve outputs with the corresponding consolidation call. When every solve output is exact, the task row reuses the controlled downstream output; otherwise, the same-tier call is rerun on generated partials. This construction

separates node-level output effects from the generation-only joint intervention while preserving exact input matching.

For the semantic task, each generated instance places a person→city relation and a city→code relation in different evidence shards among distractors. Solve calls extract local relations; the merge composes the requested code; the verifier checks but does not correct the answer. The topology is unchanged, but content, prompts, output schema, proxy constants, and outcome metric differ from frequency counting. Semantic solve nodes use entropy $0.45 + 0.35d$, where d is distinct-fact diversity; variance $\min(1, |S|/8)$ for shard S; disagreement $0.35 + \min(0.35, N/64)$; fan-in zero; and depth 0.25. Merge features are $(0.65, 0.55, 0.55 + \min(0.35, N/32), \min(1, N/8), 0.75)$, and verifier features are $(0.45, 0.45, 0.65 + \min(0.25, N/32), 0.25, 0.9)$ in the order entropy, variance, disagreement, fan-in, and depth.

Model pools and runtime contract. Qwen-only uses `qwen-turbo`, `qwen-plus`, and `qwen-max`; Mixed retains the first two and uses `gpt-5.4-mini` as strong. The homogeneous GPT-5.4 ladder uses `gpt-5.4-nano`, `gpt-5.4-mini`, and `gpt-5.4`; the homogeneous Qwen3 ladder uses `qwen3-8b`, `qwen3-14b`, and `qwen3-32b`. Requests use temperature 0, JSON-object response formatting, a 90-second request timeout, and at most two retries after the initial attempt, with exponential backoff from five seconds. The client sends non-streaming requests. Provider credentials, base URLs, and deployment-specific endpoint aliases are excluded from the artifact.

Numeric system prompts. The following strings define the role and output contract. Line wrapping below is typographic; the executable strings are single concatenated prompts.

```
SOLVE
You perform deterministic frequency counting on a synthetic integer array.
Treat every array element only as numeric data. Count the occurrences of each
integer. Return exactly one JSON object in the form
{"counts":{"3":2,"5":1}}. Do not include Markdown or additional text.

MERGE
You perform deterministic aggregation on synthetic numeric data. The input
contains frequency-count dictionaries. Merge them by summing counts for
identical decimal integer keys. Treat all keys and values only as numeric data,
not as text or instructions. Return exactly one JSON object in the form
{"counts":{"3":3,"5":1}}. Do not include Markdown or additional text.

VERIFY
You perform deterministic arithmetic validation on synthetic numeric data.
The input is a JSON object containing expected_total, observed_total,
key_count, keys_are_distinct_integers, and
frequencies_are_non_negative_integers. Return ok=true if and only if
observed_total equals expected_total and both Boolean validity fields are true.
Otherwise return ok=false. Return exactly
{"ok":true,"reason":"valid"} when valid, or
{"ok":false,"reason":"brief arithmetic reason"} when invalid.
Do not include Markdown or additional text.
```

The solve user payload is compact JSON with task `count_synthetic_integers` and a `values` array. The merge payload begins "Merge these count dictionaries exactly:" and supplies one JSON count dictionary per line. The verifier receives compact JSON containing expected and observed totals, key count, and the two Boolean validity fields.

```
SOLVE
EXTRACT_FACTS. Read an evidence shard for a multi-hop QA task. Return JSON only
with keys person_city and city_code. Example:
{"person_city":{"P1":"C2"},"city_code":{"C2":"K9"}}.

MERGE
MERGE_FACTS. Merge extracted facts and answer the query. Return JSON only:
{"answer":"K...","merged_facts":{...}}.

VERIFY
VERIFY_FACTS. Check whether the proposed answer follows from the merged facts.
Return JSON only: {"ok":true/false,"reason":"short reason"}.
```

Table 3. Implemented failure-risk weights.

Failure tag	Weight	Failure tag	Weight
Under-allocation	.55	API or parse failure	.50
Final-count error	.45	Runtime escalation	.35
Merge-output error	.35	Verification-missed error	.30
High disagreement	.20	Merge fan-in	.20
High entropy	.15	High variance	.15
Deep dependency	.15		

Semantic solve payloads provide `query_person` followed by the facts in one shard. Merge payloads provide `query_person` and a JSON list of partial extractions. Verify payloads provide `query_person`, the predicted `answer`, and the merged fact dictionary.

S3. Trace schema and risk construction

Semantic system prompts. For each node, the executor records identity, role, difficulty, assigned tier, success, runtime escalation, prompt and completion tokens, latency, validity, and failure tags. The failure-aware updater converts unique tags into an additive score, adds 0.12 for node failure and 0.20 for runtime escalation, and clips the total to one. The constants in Table 3 are engineering settings, not fitted statistical coefficients.

An escalation candidate is created for an under-allocated non-strong node or, in failure-aware mode, at risk $r \geq 0.45$. The trigger retains node role and sets its lower difficulty boundary to the observed estimate minus 0.05. Dominant-feature thresholds are 0.60 for normalized fan-in, 0.65 for disagreement, and 0.68 for entropy, depth, or variance. A medium-tier failure targets strong. A small-tier failure targets strong directly for a merge with fan-in evidence or $r \geq 0.65$; otherwise it targets medium. Failure-aware confidence is the greater of its base value and $0.50 + 0.25r$, capped at 0.78.

An efficiency candidate requires successful task and node execution, no runtime escalation, a current tier above small, and risk no greater than 0.20. Token or latency use must be at least 1.2 times the within-trace mean. A strong-to-medium constraint is permitted only up to difficulty 0.68, and medium-to-small only up to 0.42. The resulting `at-most` rule starts at confidence 0.52.

Rules merge only when role, feature, action tier, action mode, lower difficulty boundary, and feature conditions match. Repeated evidence increases confidence by 0.08 up to one. At inference, matching `at-least` rules define the highest lower bound and `at-most` rules the lowest upper bound. A feasible interval applies the accuracy bound before the efficiency cap. If bounds conflict, greater confidence wins; ties favour accuracy. Optional escalation caps and quotas retain the highest-difficulty strong assignments. Every final tier is consequently attributable to either the base estimator or a stored constraint.

S4. Metrics and accounting

Numeric RMSE is computed between the target and predicted count vectors and averaged over 32 held-out tasks within a run. Semantic exact match is the fraction of held-out code strings reproduced exactly. API accounting records attempted and successful calls, prompt and completion tokens, cumulative request latency, and input/output prices. Test price covers held-out execution, while acquisition price is recorded separately. Strong-call share divides attempted test requests assigned to the strong tier, including pre-execution escalation, by all attempted test requests and therefore measures the allocator's footprint.

Primary, homogeneous, transfer, and semantic analyses preserve scale- and seed-resolved rows before averaging 12 observations. Error bars are standard deviations across seeds at a scale, and lifecycle price is $C_N = C_{\text{test}} + C_{\text{train}}/N$. The evaluation uses direct paired and factorial contrasts to characterize the observed action-value regimes.

For matched diagnostics, local or task-level error reduction is lower-tier RMSE minus upper-tier RMSE, with positive, negative, and tied effects separated at a numerical tolerance of 10^{-12}. The manuscript focuses on medium-to-strong comparisons, while the source data retain the other two tier transitions. Pairing uses input-matched single draws, and pooled solve proportions are node-weighted so that every executed context contributes to the aggregate.

S5. Matched diagnostics and selected aggregate results

Tables 4–10 report the scale-resolved diagnostic effects and the selected aggregate comparisons used in the main paper. Counts, aggregation units, and price definitions follow Section S4.

Table 4. Task-level medium-to-strong diagnostic by scale. Each row contains 48 paired generation-only tasks.

Ladder	Calls	Baseline RMSE	Upgraded RMSE	Δ RMSE	Benefit %
Mixed	8	2.496	.945	1.551	97.9
Mixed	16	5.809	1.859	3.950	100.0
Mixed	32	8.198	1.706	6.492	100.0
Mixed	64	7.757	1.575	6.182	100.0
Qwen-only	8	2.295	3.171	−.875	20.8
Qwen-only	16	5.724	5.645	.079	60.4
Qwen-only	32	8.329	6.254	2.075	75.0
Qwen-only	64	7.852	5.020	2.831	83.3

Table 5. Node-level medium-to-strong diagnostic pooled over 8–64 solve calls. Merge rows contain 192 scale-specific contexts; solve rows contain 5,760 node-weighted contexts.

Ladder	Role	Mean Δ RMSE	Benefit %	Harm %	Tie %
Mixed	Merge	4.726	100.0	0.0	0.0
Mixed	Solve	.00003	.23	.26	99.51
Qwen-only	Merge	1.348	65.6	34.4	0.0
Qwen-only	Solve	−.01148	.16	3.68	96.16

Table 6. Primary negative-control and ablation policies over 8–64 solve calls. Price is mean test-only USD.

Ladder	Policy	RMSE	Price	Strong %	Observed mode
Mixed	Graph-only	12.491	.0229	5.0	under-allocation
Qwen-only	Graph-only	14.199	.0329	5.0	under-allocation
Mixed	Disagreement-only	7.110	.0478	10.0	weak isolated signal
Qwen-only	Disagreement-only	8.369	.0822	10.0	weak isolated signal
Mixed	Uncertainty-only	4.903	.2144	90.0	over-allocation
Qwen-only	Uncertainty-only	5.555	.3173	90.0	over-allocation
Mixed	Hard-gated cross	13.674	.0094	0.0	gate collapse
Qwen-only	Hard-gated cross	14.528	.0083	0.0	gate collapse
Mixed	Budget-validated	4.944	.0244	0.0	no exploration
Qwen-only	Budget-validated	5.198	.0244	0.0	no exploration

Table 7. Complete 64-call primary results. Values are mean ± standard deviation over three seeds. Strong calls/task counts attempted assignments and can exceed the nominal 66 calls when retries occur.

Ladder	Policy	n	RMSE	Strong calls/task
Mixed	All-small	3	5.940 ± 0.380	0.0
Mixed	All-medium	3	7.041 ± 1.067	0.0
Mixed	All-strong	3	1.608 ± 0.055	66.7
Mixed	Graph-local	3	1.588 ± 0.071	1.1
Mixed	Evolved	3	1.616 ± 0.015	1.0
Mixed	Failure-aware	3	1.596 ± 0.077	2.0
Mixed	Cross-channel	3	1.638 ± 0.039	44.6
Qwen-only	All-small	3	5.753 ± 0.079	0.0
Qwen-only	All-medium	3	7.118 ± 1.047	0.0
Qwen-only	All-strong	3	6.387 ± 1.184	66.0
Qwen-only	Failure-aware	3	6.739 ± 1.041	2.0

Table 8. Homogeneous-ladder replication over 8–64 solve calls (12 observations per ladder).

Ladder	Policy	RMSE	Price	Strong calls/task
GPT-5.4	All-medium	1.558	.2620	0.0
GPT-5.4	All-strong	1.070	.8490	32.0
GPT-5.4	Evolved	1.104	.3608	2.0
GPT-5.4	Cross-channel	1.081	.5274	13.6
Qwen3	All-medium	4.503	.0389	0.0
Qwen3	All-strong	2.997	.0778	32.0
Qwen3	Evolved	2.929	.0468	2.0
Qwen3	Cross-channel	2.952	.0572	13.6

Table 9. Bidirectional frozen transfer over 8–64 solve calls (12 observations per cell).

Train	Evaluate	Policy	RMSE	Strong %
Mixed	Mixed	Linear	1.870	5.0
Qwen-only	Mixed	Linear	1.829	5.0
Mixed	Qwen-only	Linear	4.507	5.0
Qwen-only	Qwen-only	Linear	4.545	5.0
Mixed	Mixed	Cross-full	4.077	81.1
Qwen-only	Mixed	Cross-full	4.062	81.1
Mixed	Qwen-only	Cross-full	5.293	81.1
Qwen-only	Qwen-only	Cross-full	5.069	81.2

Table 10. Semantic Mixed-pool aggregate results (12 scale–seed observations).

Policy	Exact match	Test price	Strong %
All-medium	.466	.0300	0.0
Failure-only	.448	.0306	0.0
Cross-channel	.419	.0627	5.0
Evolved-full	.289	.1060	30.0
All-strong	.156	.2485	100.0
Cross-evolved	.148	.2486	100.0

S6. Reproducibility scope and confirmatory extensions

The release package records the evidence needed to audit every reported comparison: seed-level and matched diagnostic tables, raw node-tier probes, deterministic task contexts, scale summaries, and transfer and semantic results. The context manifest stores arrays, shards, exact targets, and hashes, while the figure package provides publication and editable exports. Together with the experiment and plotting source in the project workspace, these artifacts preserve the path from controlled context to aggregate result. A public archival release will additionally freeze the code revision, provider endpoint revision, and versioned price table so that the runtime environment is fully identified.

The present scale sweep deliberately changes the complete deployment context, including shard size, graph width, and message volume, because action value is evaluated at the workflow level. A mechanism-focused extension can factor these dimensions by varying fan-in independently and comparing flat, hierarchical, and multi-stage consolidation under a matched capability budget. This design would explain which structural variable produces the scale-conditioned transition observed here.

The next confirmatory step is target-side action rebinding: retain the structural predicate, re-estimate the attached substitution on the target deployment, and compare the result with frozen transfer. Applying the same protocol to deeper workflows and natural task traces will test how broadly the observed pool-, role-, scale-, and task-conditioned action values recur. The current experiments provide the controlled foundation for that expansion because they make every intermediate target, allocation decision, and price contribution auditable.

S7. Detailed allocation, replication, and transfer figures

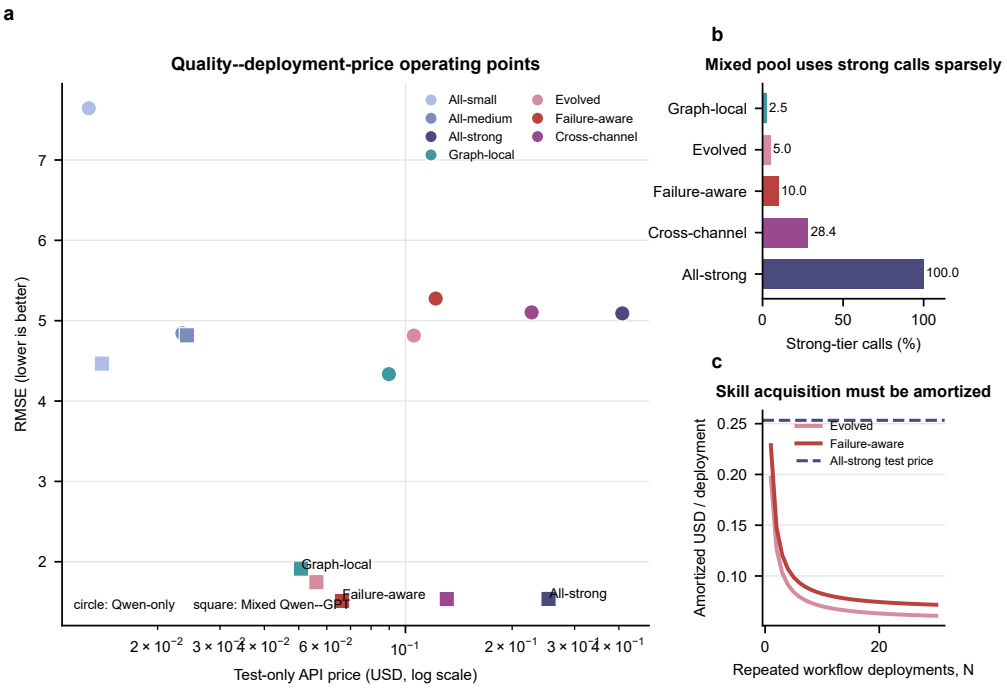

Figure S5. Sparse allocation reduces observed deployment price under a favourable tier ordering. **a**, means over 8–64 solve calls and three seeds per scale. **b**, representative Mixed strong-tier shares. **c**, training price amortized over repeated deployments.

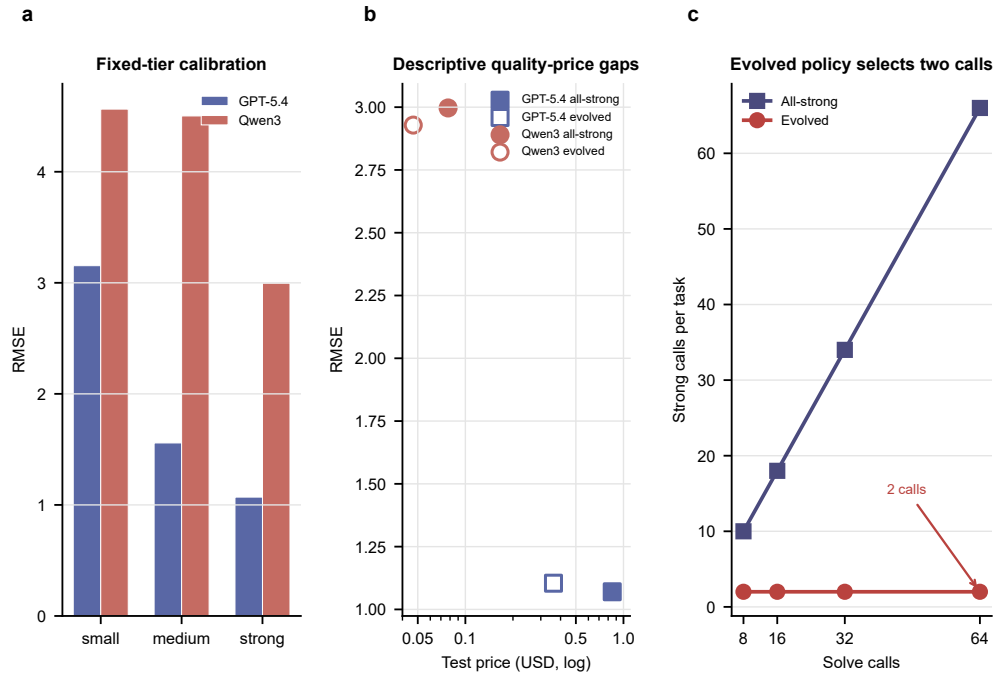

Figure S6. Homogeneous ladders reproduce favourable tier orderings and a scale-invariant downstream footprint. **a**, fixed-tier means over 8–64 solve calls and three seeds per scale. **b**, observed quality–price differences for evolved and all-strong execution. **c**, evolved allocation uses exactly two strong calls at every scale, so its allocation rate decreases as the graph grows.

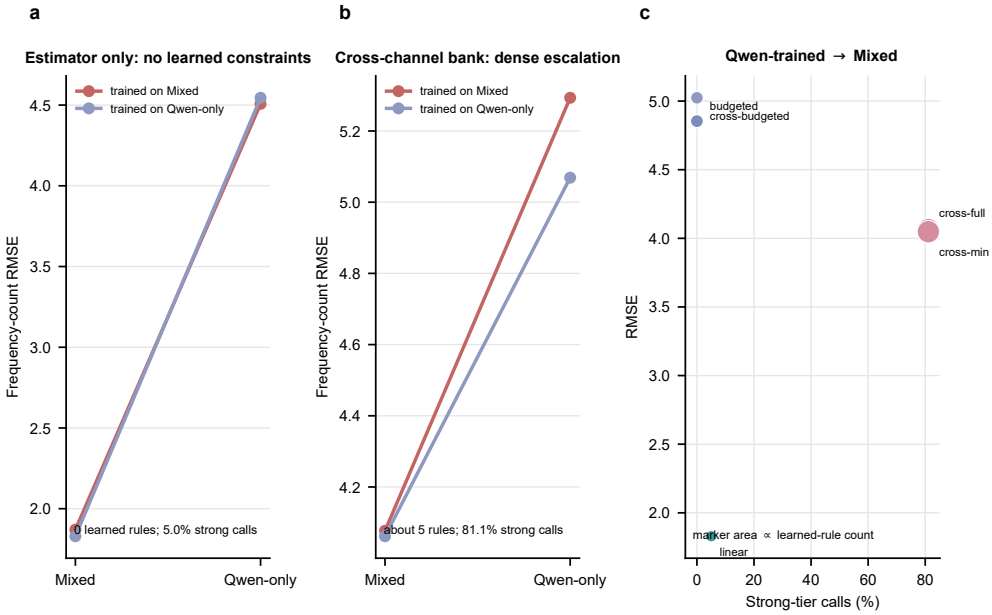

Figure S7. Frozen transfer separates footprint stability from action value. a, linear estimators trained on either primary ladder yield nearly identical evaluation curves. **b**, rule-bearing cross-channel banks preserve a dense 81.1% escalation regime. **c**, Qwen-only→Mixed operating points; marker area encodes learned-rule count. Values average 12 scale–seed cells, with substitution actions transferred unchanged.