Routing Before Reasoning: Scaling Tool Use in LLM Agents

Aditya Kuniyil Kattil
Independent Researcher
West Lafayette, Indiana, USA
akuniyil@purdue.edu

Abstract

Tool-using language-model agents are often given every available tool schema at inference time. As a catalog grows, this design increases the context processed by the expensive reasoning model even when only one tool is relevant. We study a pre-routing architecture in which a lightweight router retrieves a top-k candidate set from the catalog and exposes only those tools to the same downstream model. Our controlled benchmark contains 100 deterministic mock tools and 78 fixed, primarily single-step tasks arranged in nested toolspaces from 5 to 100 tools. Full-toolspace reasoning remained reasonably capable as the catalog grew: its execution success rate was 0.769 at 100 tools. Its priced inference cost under pricing version v1, however, rose from \$0.00145 to \$0.01224 per attempt between 5 and 100 tools. With Jev top-5 pre-routing, downstream context remained nearly flat and cost rose only from \$0.00146 to \$0.00157; execution success was 0.756 at 100 tools. The paired difference relative to the baseline was -0.0128 (task-sample bootstrap 95% CI $[-0.1026, 0.0641]$), which does not establish equivalence. A fixed-N ablation shows the mechanism: increasing k generally improves routing coverage, while downstream selection accuracy declines and cost rises. In particular, $k = 3$ and $k = 5$ have identical strict recall but different task-level outcomes, showing that binary coverage alone does not characterize downstream exposure or execution. Finally, provider-reported router confidence did not sufficiently separate correct from incorrect top-1 decisions for the tested adaptive policy. These results frame pre-routing as a way to relocate, rather than remove, toolspace complexity.

1 Introduction

An agent that can call external tools must first decide which capabilities are relevant and then reason about how to use them. A common architecture combines these operations: the reasoning model receives the full catalog of tool descriptions and parameter schemas, selects a tool, and supplies its arguments. This is simple and preserves all capabilities, but its inference context grows with the catalog. A larger prompt may also make selection among near-miss tools harder.

Pre-routing separates the two operations. A lightweight model first retrieves a small candidate set from compact tool summaries; the reasoning model then receives full schemas only for those candidates. This arrangement compresses the capability space before expensive reasoning, but it adds a new failure boundary. Too narrow a candidate set can omit the required tool. A wider set can recover coverage while restoring some of the downstream selection burden and context cost that routing was meant to avoid.

We measure this tradeoff using Jev as a concrete lightweight router and `gpt-5.6-sol` as the shared downstream reasoning model. The purpose is not to establish that one named model beats another. Instead, we ask where capability retrieval can be separated from reasoning, how the separation changes resource use and failure modes, and what router signal might choose the amount of context dynamically. Retrieving a short tool list before LLM selection is established in systems such as ToolLLM, ToolRerank, Toolshed, PORTS, and Toollery [8, 11, 5, 6, 10]. Our contribution is therefore an empirical decomposition of this architecture, not the pre-routing pattern itself.

We organize the study around three questions:

RQ1: Toolspace scaling. As the number N of available tools grows, how do full-toolspace reasoning and lightweight pre-routing differ in execution success, inference context, priced cost, latency, and failures?

RQ2: Candidate-set size. At a fixed large toolspace, how does k trade routing coverage against downstream selection difficulty?

RQ3: Adaptive routing. Can router confidence reliably determine whether the reasoning model should see a smaller or larger candidate set?

The answer to RQ1 is not that full-toolspace reasoning catastrophically fails. In this benchmark, baseline execution success remains between 0.744 and 0.808 over $N \in \{5, 10, 25, 50, 100\}$. The clearer separation is economic: baseline agent context grows from 282 to 2,982 mean tokens, whereas top-5 agent context remains near 290 tokens and the lightweight router absorbs catalog growth. At $N = 100$, baseline and top-5 differ by one

net success among 78 paired tasks, but the associated uncertainty is too wide to support an equivalence claim.

For RQ2, increasing k improves strict Recall@k from 0.872 at $k = 1$ to 0.962 at $k = 10$, while conditional selection accuracy falls from 1.000 to 0.853. Execution success increases more slowly than routing coverage, and cost rises as additional full schemas reach the reasoning model. No decision rule was specified, so the ablation defines a tradeoff curve rather than a winning k. For RQ3, the tested confidence policy routes nearly every scored holdout example to $k = 1$: some wrong top-1 routes carry high reported confidence, and the low-confidence escalation branch is unused.

Our contributions are deliberately limited:

1. a controlled benchmark for toolspace scaling with fixed tasks and nested, difficulty-aware capability sets;
2. a comparison of full-toolspace reasoning and lightweight top-k pre-routing from 5 to 100 tools;
3. an empirical decomposition showing that routing coverage and downstream execution are distinct objectives: increasing k recovers routing misses while admitting downstream selection errors, and identical strict Recall@k can accompany different execution outcomes;
4. a negative evaluation of one confidence-based adaptive candidate-set policy; and
5. a manifest-pinned path from immutable executions to publication figures and paired analyses.

2 Pre-Routing Architecture

Full-toolspace reasoning. For a request x and a presented toolspace $\mathcal{T}_N$, the baseline sends full definitions for all N tools to the reasoning model. The model emits a tool name and arguments; the selected deterministic mock implementation executes; a deterministic evaluator scores the attempt:

$$x \to \text{reasoner}(\mathcal{T}_N) \to \text{tool call} \\ \to \text{execution} \to \text{evaluation}.$$

The model therefore performs relevance filtering and final selection in one step.

Fixed-k pre-routing. The routed path first sends the request and each tool's frozen *routing summary* to JEV. The returned Choice distribution is sorted locally, and the top k names are retained. The same reasoner used by the baseline receives the full schemas of only those tools:

$$x \to \text{JEV}(\mathcal{T}_N) \to \mathcal{C}_k \\ \to \text{reasoner}(\mathcal{C}_k) \to \text{execution} \to \text{evaluation}.$$

The router never receives full JSON schemas, while the reasoner always does for its candidate set. This intentionally shifts catalog-scale input to a lower-priced model. Router input still grows with N; the claim is about controlling what reaches expensive reasoning, not making total context independent of N.

Adaptive candidate sets. The adaptive path uses the same JEV ranking plus its provider-reported confidence. A frozen threshold policy chooses $k = 1$, $k = 5$, or an escalation to an LLM router at $k = 5$ before invoking the downstream reasoner. Confidence is stored separately from the highest Choice probability and is the only branch signal.

Coverage is not selection. These paths expose two distinct events. *Routing coverage* asks whether the required tool is included in $\mathcal{C}_k$. *Downstream selection* asks whether the reasoner chooses the required tool when it is available. Increasing k can improve the former and hurt the latter. Moreover, two sets can have the same binary coverage but different distractors, ranks, or acceptable alternatives. This distinction is central to interpreting the ablation.

3 Experimental Methodology

3.1 Registry, tasks, and nested toolspaces

The registry contains 100 deterministic mock tools across files, email, calendar, code, chat, documents, CRM, and task-management domains. Each definition includes a name, description, JSON parameter schema, compact routing summary, and a hand-authored list of near misses. Executors use fixed fixtures rather than production external APIs.

Dataset v0.2 contains 78 fixed single-step tasks. Each has one required tool; some additionally label acceptable alternatives. Concrete expected arguments are included only when stated in the prompt and are checked by subset match. Prompts do not expose tool identifiers or copy routing summaries verbatim.

For task i, the presented set at size N is

$$\mathcal{T}_{i,N} = \mathcal{R}_i \cup \text{prefix}(\mathcal{D}_i, N - |\mathcal{R}_i|), \tag{1}$$

where $\mathcal{R}_i$ is the required-tool set and $\mathcal{D}_i$ is a frozen distractor sequence. Near misses are merged first; a global domain-interleaved tail follows. Thus the same task is evaluated at every supported size, the required tool remains available, and $\mathcal{T}_{i,5} \subset \cdots \subset \mathcal{T}_{i,100}$. The sequence is authored from tool descriptions rather than model scores and stored with each run.

3.2 Frozen experiments

M3 is the scaling matrix: baseline, JEV top-1, and JEV top-5 at $N \in \{5, 10, 25, 50, 100\}$. M4 fixes $N = 100$ and independently executes JEV with $k \in \{1, 3, 5, 10\}$. Both

use JEV `jev-1.13.0`, downstream model `gpt-5.6-sol`, temperature 0, seed 0, and one repetition per primary cell. M4 re-executes $k = 1$ and $k = 5$ rather than reusing M3 outputs. The adaptive experiment uses a 24-task held-out split from dataset v0.1 at $N = 20$, with policy thresholds selected earlier on its disjoint development split.

All result directories are immutable and pinned by epoch manifests dated 2026-09-24: M3 at 05:58:54Z, M4 at 14:06:21Z, and adaptive at 04:57:43Z. The publication synthesis is checkpoint `ed3dd18`; the narrative checkpoint is `ad38327`. The same executor and evaluator code is used across architectures. We report only the frozen synthesis and do not combine cells across epochs into a new aggregate.

3.3 Metrics

Execution Success Rate. ESR is a narrow deterministic execution metric, not full semantic task success. An attempt succeeds exactly when (i) the selected tool is required or acceptable, (ii) any labeled expected-argument subset matches, and (iii) deterministic execution returns success:

$$\begin{aligned}\text{success} = [t_{\text{sel}} \in \mathcal{R}_i \cup \mathcal{A}_i] \\ \wedge\, [\text{arguments match}] \\ \wedge\, [\text{executor succeeds}].\end{aligned} \tag{2}$$

There is no LLM judge and no evaluation of a natural-language final answer.

Routing and selection. Strict Recall@k is the fraction of required tools present among the first k candidates. Because these tasks have one required tool, it is a binary per-task hit. Lenient recall also counts a required or acceptable candidate, but does not replace the strict primary metric. Selection accuracy is defined only for attempts with a valid agent decision and a required tool in the candidate set; routing misses are excluded from its denominator.

Resources. Input and output tokens are recorded separately for router and reasoner. Agent-context plots use mean downstream-agent tokens per attempt. Cost applies the frozen model-specific token rates in pricing version v1 and is therefore *priced inference cost under v1*, not a universal deployment cost. Latency is measured as monotonic wall-clock time through routing, reasoning, synchronous execution, and evaluation. Because latency varied materially between M3 and M4 epochs, we treat it as secondary.

Failures. The R0–R6 taxonomy assigns the earliest attributable stage. R0 is an infrastructure or malformed-provider-output failure and is excluded from ESR and routing denominators. R1 is a valid routing decision that omits the required tool; R2 is wrong selection when the required tool is available; R3 is an argument mismatch; R4 is tool execution failure; R5 is reserved for reasoning after successful execution and does not fire in this benchmark; R6 denotes benchmark ambiguity. R6 is not silently counted as model failure.

3.4 Paired analysis and scope of uncertainty

At $N = 100$, baseline and top-5 outcomes are paired by the 78 shared task IDs. The predeclared synthesis reports the observed ESR difference, an exact two-sided McNemar test on discordant binary outcomes, and a percentile interval from paired bootstrap resampling of task IDs. These intervals characterize uncertainty over this benchmark task sample. With one primary repetition, they do not estimate provider or run-to-run variability and are not equivalence tests. We do not add post-hoc tests across the complete grid.

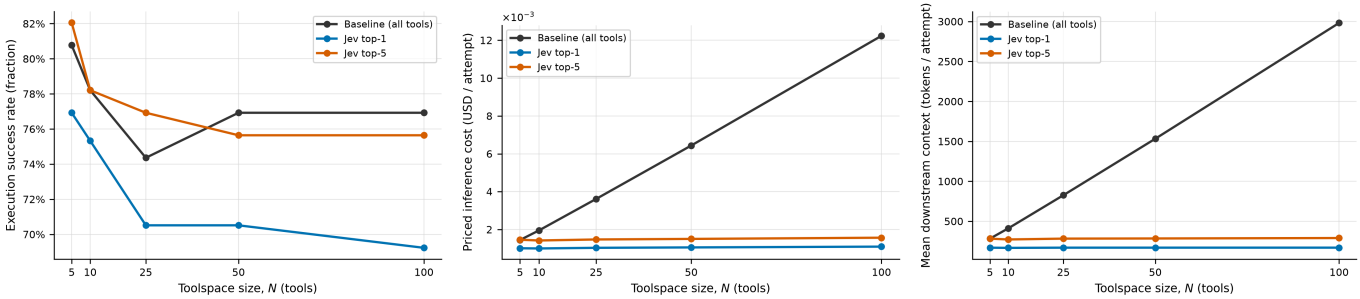
4 Scaling Toolspaces

Figure 1 answers RQ1. Baseline ESR is 0.808 at $N = 5$ and 0.769 at $N = 100$, with a minimum of 0.744 at $N = 25$. The series does not show a catastrophic monotonic collapse. Top-5 moves from 0.821 to 0.756, while top-1 moves from 0.769 to 0.692 as Recall@1 becomes the tighter bottleneck.

The resource curves separate more clearly. Mean baseline agent tokens grow from 282 at $N = 5$ to 2,982 at $N = 100$. Top-5 grows from 282 to 290, and top-1 remains near 169. Under pricing version v1, baseline cost per attempt rises by about 8.5×, from \$0.00145 to \$0.01224, while top-5 rises from \$0.00146 to \$0.00157. Thus the first clear scaling pressure in this benchmark appears economically, before any catastrophic degradation in execution success. These totals include router inference for routed paths; the apparent flatness in downstream context is therefore not obtained by ignoring router work. Rather, catalog growth is processed by the cheaper router instead of being repeatedly included as full schemas in the reasoner's input.

Table 1 prevents over-reading the endpoint. Top-5 records one fewer success than baseline: six tasks succeed only under baseline and five only under top-5. The observed execution-success difference is small, but uncertainty is too wide to support an equivalence claim. The McNemar result likewise does not imply identical performance; it indicates that this small paired sample does not show a directional imbalance among the 11 discordances.

Where failures move. Seven baseline tasks that succeed at $N = 5$ fail at $N = 100$; six become R2 selection failures and one becomes R3. Among all 18 baseline failures at $N = 100$, 11 are R2 and seven are R3. Top-5 rescues five baseline failures, four of which are baseline



(a) Execution success. **(b)** Priced cost under v1. **(c)** Downstream agent tokens.

Figure 1: M3 scaling over 78 tasks. Full-toolspace execution remains reasonably capable through $N = 100$, while agent context and priced cost grow strongly. Fixed top-k confines the schemas sent to the expensive reasoner; router context, included in cost but not panel (c), absorbs catalog growth. Plot inputs are the frozen synthesis payloads.

Table 1: Paired M3 comparison at $N = 100$. "Top-5 only" and "baseline only" are successful attempts under one condition but not the other.

Quantity	Observed value
Baseline ESR	0.7692
Top-5 ESR	0.7564
ΔESR (top-5 − baseline)	−0.0128
Paired bootstrap 95% CI	[−0.1026, 0.0641]
Both succeed / both fail	54 / 13
Baseline only / top-5 only	6 / 5
McNemar exact two-sided p	1.000
Cost/attempt: baseline	$0.01224
Cost/attempt: top-5	$0.00157

R2 errors. It also loses six baseline successes: two routing misses (R1), two downstream selection errors (R2), one argument error (R3), and one execution error (R4). Pre-routing therefore changes which tasks fail rather than uniformly improving or degrading them.

Argument failures are unusually persistent. Six tasks are R3 in all 15 M3 cells, and a seventh is R3 in 14 cells; per-cell R3 counts stay near seven or eight. This concentration is consistent with a largely fixed task subset that is mostly orthogonal to routing treatment. It also cautions against attributing every remaining failure to insufficient retrieval.

Latency does not support an equally clean headline. Mean routed latency is higher than baseline at several small and middle N values but lower for top-5 at $N = 100$ in M3. Independent M4 executions show substantial epoch variation. We therefore report latency as a secondary wall-clock observation rather than evidence of a stable crossover.

5 How Many Tools Should the Agent See?

M4 holds the 100-tool space fixed and changes only k. Table 2 and Figure 2 show the expected first-order trade-

Table 2: M4 at $N = 100$ (78 attempts per row). Cost is USD per attempt under pricing version v1.

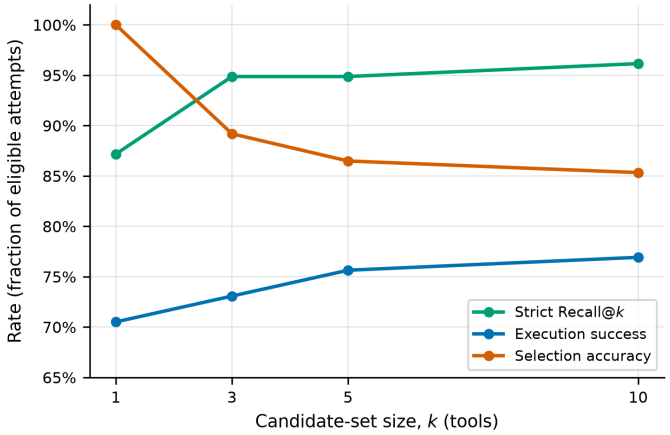
k	Strict R@k	ESR	Sel. acc.	Cost
1	0.872	0.705	1.000	0.001091
3	0.949	0.731	0.892	0.001334
5	0.949	0.756	0.865	0.001569
10	0.962	0.769	0.853	0.002186

off. Strict recall generally rises as the reasoner sees more candidates, from 0.872 at $k = 1$ to 0.962 at $k = 10$. Conditional selection accuracy moves in the other direction, from 1.000 to 0.853. The perfect value at $k = 1$ is conditional: when the required tool is the sole candidate, the reasoner selects it, but routing misses are outside the denominator. ESR rises from 0.705 to 0.769 rather than tracking recall one-for-one. Mean agent tokens rise from 169 to 444, and priced cost per attempt roughly doubles within the tested range.

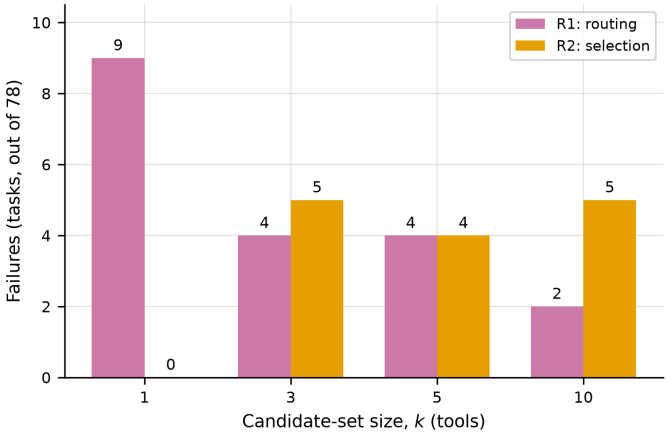
The failure counts make the stages concrete. R1 falls from nine cases at $k = 1$ to two at $k = 10$. R2 is absent at $k = 1$ but ranges from four to five once the reasoner must choose among multiple candidates. R3 remains exactly eight in each cell. Thus a wider candidate set removes some routing failures while admitting selection failures and leaving the persistent argument subset essentially untouched.

Coverage versus composition. The $k = 3 \to 5$ transition is especially informative. Strict Recall@k is unchanged at 0.949, yet ESR rises from 0.731 to 0.756. Binary required-tool coverage alone is therefore insufficient to characterize the candidate set presented downstream or to predict its execution outcome. Adding two tools also changes the exposed set, but this experiment does not independently manipulate composition while holding k fixed. Task-level transitions include both gains and regressions, so the aggregate ESR increase should not be read as monotonic benefit for every task.

Adjacent-k analysis reinforces this point. From $k = 1$



(a) Strict routing recall, ESR, and conditional selection accuracy.



(b) Routing (R1) and downstream selection (R2) failures.

Figure 2: M4 candidate-set ablation at $N = 100$. Larger sets usually recover routing coverage, but selection among the exposed schemas becomes harder. The ablation has no decision rule and does not designate an optimal k.

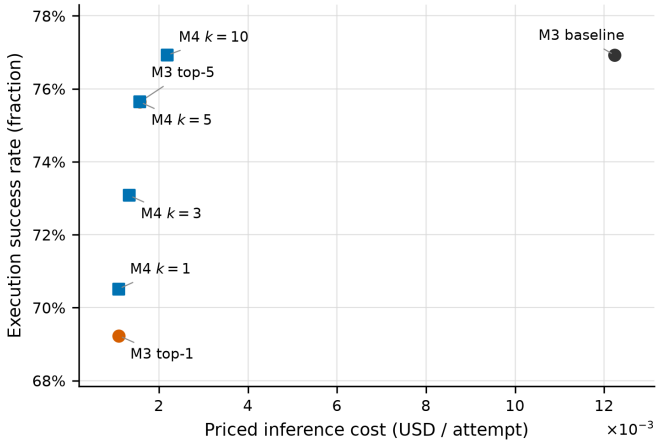


Figure 3: Measured cost–ESR operating points at $N = 100$. M3 and M4 cells are shown as separate executions. The plot is not a deployment extrapolation or a rule for selecting k.

to $k = 3$, six additional tasks gain required-tool coverage, but only two of those become execution successes; two previously successful tasks regress. From $k = 3$ to $k = 5$, no additional task gains strict coverage, while task outcomes still change. From $k = 5$ to $k = 10$, one task gains coverage but does not become successful, and one prior success regresses. These observations are consistent with candidate identity, rank, and set size affecting downstream selection, but the present ablation establishes only that strict coverage is not a sufficient explanation.

Figure 3 summarizes the measured economic operating points. The M4 $k = 10$ cell reaches ESR 0.769 at \$0.00219 per attempt, while the independently executed M3 baseline has the same rounded ESR at \$0.01224. This descriptive equality of two aggregate point estimates neither makes the task outcomes identical nor establishes general cost-quality equivalence. Without a predeclared utility function over failures, context, cost, and latency, the frozen analysis intentionally leaves `decision_rule` null.

Independent re-execution. M4 independently re-executed $k = 1$ and $k = 5$. Binary task outcomes agree with M3 on 77 of 78 tasks for $k = 1$ and all 78 for $k = 5$. The $k = 1$ ESR changes from 0.692 to 0.705; $k = 5$ remains 0.756. This is useful descriptive evidence about these two epochs, not a formal replication study and not an estimate of run-to-run provider variance.

6 Adaptive Candidate-Set Selection

We tested whether provider-reported confidence could allocate context dynamically: a frozen policy used $k = 1$ at high confidence, $k = 5$ at intermediate confidence, and an LLM-router escalation at low confidence. It did not reliably distinguish incorrect high-confidence top-1 routes. Three incorrect routes had confidence 0.66, 0.80, and 0.98, all above the high threshold. Consequently, 18 of 19 routing-scored adaptive attempts used the $k = 1$ branch, one used $k = 5$, and none escalated; mean selected k was 1.21.

We therefore treat confidence-based adaptation as a negative result for this policy rather than retuning on the 24-task holdout. JEV's reported confidence did not provide sufficient separation between correct and incorrect top-1 decisions here, but this does not generalize to other confidence signals or adaptive policies. Threshold mechanics and the descriptive adaptive/fixed-k cell results are reported in the appendix.

7 Discussion

Retrieval changes the location of complexity. At 100 tools, the full-toolspace reasoner remains capable, so

pre-routing is not required to make the benchmark function. Its main measured benefit is architectural: compact summaries and catalog-scale ranking are assigned to a lightweight model, while only a bounded set of full schemas reaches expensive reasoning. The cost curve reflects this division. The router's context still grows, and its errors become part of the system.

A candidate budget creates two pressures. Small k makes retrieval decisive: R1 dominates and conditional selection is easy. Larger k improves coverage but increases agent context and admits R2 errors. Neither Recall@k nor selection accuracy alone measures the whole path because their denominators refer to different stages. ESR connects routing, argument construction, and deterministic execution, but it still does not evaluate semantic final-answer quality.

Recall is not a complete candidate-set description. The unchanged strict recall but changed ESR between $k = 3$ and $k = 5$ shows that a binary required-tool hit does not determine downstream execution. Candidate identity, rank, and set size are plausible mechanisms because candidate sets are structured prompts, but this study does not isolate them causally. A controlled distractor-composition experiment at fixed k and fixed required-tool coverage is a direct next step.

Economic pressure precedes catastrophic quality loss. The baseline's priced cost grows by about 8.5× from $N = 5$ to $N = 100$, while top-5 cost stays comparatively flat and baseline ESR remains reasonably capable. In this benchmark, the first strong scaling signal is therefore economic rather than catastrophic execution degradation. This is a systems observation about where full schema context is processed, not evidence that larger catalogs never reduce quality.

Cost-quality decisions are application-specific. The measured points can support a utility calculation, but they do not supply one. Applications may value an R1 differently from an R2, require a maximum context window, price latency tails, or impose heterogeneous tool risks. The lack of a frozen decision rule is why we report observed operating points instead of calling top-5 or any other k optimal. Priced costs also depend on the pinned v1 rates and should not be extrapolated to deployment traffic without new assumptions.

Confidence needs calibration for the target decision. The adaptive policy asks a demanding question: not merely whether confidence correlates with correctness on average, but whether the high-confidence tail excludes enough wrong top-1 routes to justify withholding additional candidates. The observed overlap violates that requirement. Prior calibration and cost-aware routing work likewise treats the routing signal, target constraint, and held-out decision rule as central [2, 1, 7, 4].

8 Limitations

The study has 78 primary evaluation tasks and a 24-task adaptive holdout. Tools are deterministic mocks rather than production APIs, and tasks are primarily single-step. This design isolates selection and arguments but omits authentication, changing external state, multi-call plans, recovery, and natural-language answer quality.

We evaluate one main lightweight-router family, one downstream reasoning model, one frozen prompt/summary design, and catalog sizes only through 100. Tool descriptions, routing summaries, near-miss lists, and task mix are benchmark-specific. The observed tradeoffs may change with other models, schemas, domains, or larger catalogs; we make no universal generalization.

Primary M3 and M4 cells have one repetition. Paired task-sample bootstrap intervals resample the 78 benchmark tasks and do not estimate provider or run-to-run variance. We performed no equivalence test and make no equivalence claim. The independent M3/M4 overlap is only a descriptive re-execution check. Latency is noisy across epochs and is secondary.

ESR is intentionally narrower than end-to-end success. It credits required or acceptable tool selection, labeled argument checks, and deterministic executor success, but not whether a user-facing answer is complete. Selection accuracy is conditional on required-tool coverage, which makes it unsuitable as a stand-alone system metric. The failure taxonomy contains reserved categories that do not fire here.

Finally, the candidate-size ablation has no predeclared utility function and tests four fixed values. The adaptive experiment evaluates one threshold policy on a small holdout; absence of escalation and high-confidence misses justify a negative conclusion only for that policy and signal. More policies cannot be responsibly selected on the same holdout.

9 Related Work

Retrieving tools before reasoning. Pre-selecting tools is established prior work, not a contribution of this paper. ToolLLM adds a neural API retriever to a tool-use model trained over more than 16,000 APIs [8]. Subsequent systems improve retrieval through LLM-generated queries [3], hierarchy-aware reranking and adaptive truncation [11], or retriever alignment to a frozen tool-calling model [6]. These works primarily improve retrieval quality or end-to-end tool use across large, often heterogeneous libraries.

Catalog and shortlist scaling. Toolshed varies both total tool count and top-k while reporting retrieval, agent, and token-cost tradeoffs [5]. Toollery compresses hundreds to roughly 79,000 capabilities into bounded top-k sets before final LLM selection [10]. Most directly, Repantis et al. treat shortlist depth itself as the object of evaluation, show that shorter candidate lists can improve downstream selection, and learn an adaptive depth policy across registries of 20 to 3,251 tools [9]. These results substantially overlap the architectural motivation and rule out claims that the pre-routing architecture or the coverage–selection tension is new here.

Our narrower contribution is a controlled systems decomposition. Nested toolspaces hold tasks and required capabilities fixed while N changes; baseline and routed paths share the downstream model and deterministic evaluator; and task-level outcomes are attributed to routing, selection, arguments, or execution. The k ablation then relates strict coverage, conditional selection, execution, context, and priced inference cost in the same frozen sample. The unchanged Recall@k but changed ESR at $k = 3 \rightarrow 5$ is evidence that the binary retrieval metric alone does not characterize downstream behavior. We do not claim this is the first study of shortlist size, nor do we causally identify candidate composition.

Confidence-based routing and cascades. FrugalGPT and RouteLLM study cost-aware routing among language models rather than among candidate-set sizes [1, 7]. Calibration work formalizes when reported probabilities track empirical correctness [2]; recent cascade work explicitly calibrates uncertainty before threshold-based escalation [4]. Our adaptive experiment is smaller and tool-specific: it asks whether one router's reported confidence can choose k. Its negative result supports no general claim about calibrated cascades.

10 Conclusion

Growing the full catalog did not catastrophically reduce ESR in this benchmark, but it strongly increased the context and priced cost borne by the reasoner. Pre-routing bounded downstream context while moving failures to the boundary between coverage and candidate selection. At $N = 100$, top-5's observed ESR was close to the baseline point estimate, but uncertainty was too broad for equivalence. More coverage also did not translate one-for-one to successful execution, and the tested confidence signal was insufficient for reliable adaptive allocation. Pre-routing can compress the capability space before reasoning; it does not eliminate complexity, but changes where cost and risk are paid.

Code and artifacts. Code, benchmark definitions, methodology, and canonical synthesis artifacts are available at `github.com/Akk525/jev_eval`. Figures and tables trace to the repository's frozen synthesis artifacts; raw provider result directories are not published.

References

[1] 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.

[2] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.

[3] Mohammad Kachuee, Sarthak Ahuja, Vaibhav Kumar, Puyang Xu, and Xiaohu Liu. Improving tool retrieval by leveraging large language models for query generation. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 29–38, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics.

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

[5] Elias Lumer, Vamse Kumar Subbiah, James A. Burke, Pradeep Honaganahalli Basavaraju, and Austin Huber. Toolshed: Scale tool-equipped agents with advanced RAG-tool fusion and tool knowledge bases. *arXiv preprint arXiv:2410.14594*, 2024.

[6] Lorenzo Molfetta, Giacomo Frisoni, Nicolò Monaldini, and Gianluca Moro. PORTS: Preference-optimized retrievers for tool selection with large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10007–10030, Suzhou, China, November 2025. Association for Computational Linguistics.

[7] 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*, 2025.

[8] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] Vyzantinos Repantis, Ameya Gawde, Harshvardhan Singh, and Joey Blackwell. How many tools should an LLM agent see? a chance-corrected answer. *arXiv preprint arXiv:2605.24660*, 2026.

[10] Xiangxi Tian and Ran Guan. Toollery: Scaling LLM agents to thousands of skills and tools. *arXiv preprint arXiv:2609.22218*, 2026.

[11] Yuanhang Zheng, Peng Li, Wei Liu, Yang Liu, Jian Luan, and Bin Wang. ToolRerank: Adaptive and hierarchy-aware reranking for tool retrieval. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pages 16263–16273, Torino, Italia, May 2024. ELRA and ICCL.

A Failure Taxonomy

Table 3: Frozen failure taxonomy. The earliest attributable stage wins.

Code	Name	Definition
R0	Infrastructure failure	A provider, network, API, or malformed-output event prevents a valid decision. Excluded from applicable quality denominators and reported separately.
R1	Routing failure	The router returns a valid decision, but the required tool is absent from the candidate set.
R2	Selection failure	A required tool is available to the reasoner, but it selects an incorrect tool.
R3	Argument failure	The selected tool is correct, but labeled expected arguments do not match.
R4	Execution failure	The invocation is appropriate, but the deterministic tool returns failure.
R5	Reasoning failure	Execution succeeds but task completion is wrong; reserved and not assigned in the single-step benchmark.
R6	Benchmark ambiguity	The expected execution is too restrictive or multiple reasonable paths exist; not silently counted as model failure.

B Detailed Construction and Controls

Each of the eight tool domains contributes deterministic definitions and fixture-backed implementations. Registry hashing covers names, descriptions, domains, parameter schemas, routing summaries, near misses, and the global tail. For each task, required-tool near-miss lists are merged round-robin with tool-name tie-breaking, followed by the global tail with duplicates removed. The exact ordered toolspace is recorded for every attempt. A task would be ineligible if its required set could not fit in N; all v0.2 tasks have one required tool and are eligible at the supported sizes.

Routers rank only routing summaries. The baseline and downstream reasoner receive full name, description, and JSON schema records. The agent prompt, model parameters, task labels, registry, executor fixtures, and evaluation logic are held fixed across baseline and routed conditions. Result lines include raw routing candidates, selected tool and arguments, token accounting, latency, cost components, failure label, and exact configuration identifiers.

C Full Scaling and Ablation Tables

Table 4: M3 execution success, priced cost, and downstream agent tokens. Values are copied from `analysis/synthesis/synthesis.json`.

	ESR			Cost/attempt			Agent tokens		
N	Base	$k=1$	$k=5$	Base	$k=1$	$k=5$	Base	$k=1$	$k=5$
5	.808	.769	.821	.001446	.001012	.001463	282	169	282
10	.782	.753	.782	.001957	.001002	.001423	411	167	272
25	.744	.705	.769	.003617	.001036	.001477	826	169	282
50	.769	.705	.756	.006440	.001057	.001506	1533	169	284
100	.769	.692	.756	.012236	.001098	.001569	2982	169	290

D Adaptive Policy Details

The frozen policy uses provider-reported `confidence`, never `top1Probability`. Confidence ≥ 0.6 chooses JEV top-1; confidence in $[0.5, 0.6)$ chooses JEV top-5; confidence below 0.5 escalates to an LLM router at top-5. Thresholds were selected from a preregistered grid using routing hits on the even-hash development split. The adaptive manifest

Table 5: M4 extended ablation.

k	Strict R@k	Lenient R@k	ESR	Sel. acc.	Agent tok.	Cost	R1	R2	R3
1	.872	.897	.705	1.000	169	.001091	9	0	8
3	.949	.962	.731	.892	230	.001334	4	5	8
5	.949	.974	.756	.865	290	.001569	4	4	8
10	.962	.987	.769	.853	444	.002186	2	5	8

evaluates the complementary odd-hash holdout. Of six incorrect fixed top-1 routes on the holdout, confidence ranges from 0.34 to 0.98, and three exceed the high threshold. Adaptive ESR is 0.684 on its scored denominator, compared with 0.667 for fixed $k = 1$ and 0.783 for fixed $k = 5$ in the same manifest; these small cells do not establish a general ordering. No thresholds were fit after observing these outcomes.

E Reproducibility and Statistical Notes

The immutable M3 and M4 manifests point to 15 and four completed result directories, respectively; the adaptive manifest points to four completed cells. Publication plots are generated from JSON payloads beneath `analysis/synthesis/figures/`, themselves derived from the canonical synthesis. The paper includes rendered versions of those payloads rather than recomputing series from result directories.

For the paired ESR interval, a bootstrap draw resamples the 78 shared task identifiers with replacement, retaining each task's two binary outcomes, and recomputes the mean within-task difference. The interval therefore describes sensitivity to the benchmark task sample. It contains neither repeated provider calls nor a hierarchical run component. Exact McNemar uses only the six baseline-only and five top-5-only successes and is reported as supporting analysis, not as an equivalence test.