Institute of Trustworthy Embodied AI
Fudan University

Disagree to Explore, Agree to Commit: Routing-Guided Test-Time Scaling for Software Agents

Kang Chen[1*], **Junjie Nian**[1*], **Yixin Cao**[1,2†], **Yu-Gang Jiang**[1]

[1]Fudan University, [2]Shanghai Innovation Institute

Abstract

Software-engineering agents solve repository-level tasks through long, stochastic tool-use trajectories, and repeated attempts often find fixes missed by one run. Test-time scaling is difficult because patches lack canonical answer forms, while sibling actions from a shared prefix are correlated. We study whether native MoE router traces can guide steering and selection without an external judge or selection-time test execution. Our analysis shows that routing provides a robust behavioral-role signal; token-granular readouts and decision-matched comparison sets turn it into effective control. We therefore introduce RISA (*Routing-Informed Steering and Arbitration*): within trajectories, routing encourages diverse exploration and controlled convergence during patch commitment; across separately sampled trajectories, agreement at informative patch positions selects a final candidate. We evaluate on SWE-bench Verified using open-weight sparse MoE agents across scales and reasoning-effort settings. RISA's routing arbitration raises the macro-average resolved rate from 44.9% under uniform sampling to 48.2% on the `gpt-oss` family, matching text consensus without answer-string matching, and it transfers to `Qwen3.6`, where it improves on uniform choice and matches text consensus on the full 500-task benchmark.

Correspondence: kchen24@m.fudan.edu.cn, yxcao@fudan.edu.cn

Web: https://CckFdu.com/RISA

1 Introduction

Repository-level repair unfolds as a sequence of decisions. During one attempt, an agent may inspect a failing test, search for a symbol, execute a diagnostic command, revise a file, and repeat this cycle before producing a diff. Giving the agent more inference compute therefore creates two nested coordination problems. At each tool step, many actions sampled from the same prefix compete for execution; after several complete runs, multiple non-canonical patches compete for submission. Although both resemble best-of-N selection, their candidates have very different dependence: siblings inherit the same trajectory state, whereas complete attempts may reach a patch through different repository paths.

Existing systems coordinate these choices with execution feedback, surface comparison, trained verifiers, or judge models [1, 2, 25]. Sparse mixture-of-experts (MoE) models expose a complementary signal already produced during inference: for every token and layer, the router records the selected experts and their weights. Because expert identities are shared across tokens and actions, this sparse trace can place textually different

* Contributed equally.
† Corresponding author.

Figure 1 Routing supports decision-matched control. **(a)** Patch-writing fingerprints overlap across outcomes, motivating relational use. **(b)** Behavioral roles separate cleanly. **(c)** Across 658 final-step generations, the least-probable quarter carries 72% of total token surprisal (median and interquartile range; dotted line: equal contribution).

candidates in a common coordinate system. It records where the model allocated internal computation, not only what text it produced. Prior work studies lexical and semantic specialization in these traces [13, 15, 22], but not how their meaning changes across the nested choices of a long tool-use trajectory.

Routing similarity becomes informative relative to the candidates being compared. Same-prefix siblings share nearly all preceding evidence, making their agreement partly shaped by a common continuation. Separately sampled attempts can inspect different files and run different diagnostics, making convergence at the patch stage a stronger signal. We therefore follow the trace outward through four questions. Action fingerprints first reveal *what* computation an agent is performing. Similarity to its executed history then marks computational revisitation. Within long patches, fine-grained variation concentrates at low-probability *decision tokens*. Finally, we test how agreement changes as the reference set moves from correlated siblings to independently developed fixes. Figure 1 previews the role geometry and decision-token concentration behind this progression.

These findings yield a simple principle: match the routing comparison set to the decision. We instantiate it as RISA (*Routing-Informed Steering and Arbitration*), a two-level controller. Within an attempt, a routing-derived role gate separates exploration from patch writing; exploration favors actions unlike recent executed history, while a cohort of write candidates uses guarded peer convergence. Across K attempts, each accumulated patch is re-encoded once, and the final patch with the highest decision-token agreement across attempts is selected. Step-level fingerprints come from the original generation pass, and terminal selection requires no external judge or executing candidate patches solely to select among them.

Across the six `gpt-oss` model–effort conditions, RISA improves over Uniform by 2.3–5.7 points, lifting the macro-average from 44.9% to 48.2%; Text reaches 48.0% and RISA-H 48.3%. On the cross-family full 500-task `Qwen3.6-35B-A3B` condition, RISA improves over Uniform by 3.5 points ($p < 0.001$) and matches Text (exact McNemar p=1.000), after refitting only the architecture-dependent role centroids.

Our contributions are as follows:

- A multiscale map of MoE routing as a behavioral coordinate system, spanning action roles, trajectory repetition, decision-token detail, and cross-attempt convergence.
- RISA, a two-level routing-guided controller that turns these findings into role-conditioned exploration, commitment, and final-patch selection.
- An evaluation over two model scales, three reasoning-effort settings, and a second MoE family, in which RISA raises resolved rate over Uniform in every condition while operating directly on routing traces.

2 Related Work

Interpreting MoE routing. Sparse MoEs expose expert identities and router weights at every token [8, 17]. Analyses find a strong lexical component [22], expert specialization [13], and same-token semantic sensitivity [15]. These studies establish token- and corpus-level routing structure. We extend the analysis to what aggregation preserves over heterogeneous actions, long spans, and different comparison sets. Routing-agreement decoding (RAD) demonstrates endpoint selection from fixed, token-aligned routing anchors [4]. Accumulated software patches have no canonical anchor, and long-horizon agents expose a second, same-prefix choice scale. We therefore localize final patches by teacher-forced token probability and reuse the coordinate against recent history, role-matched peers, and separately sampled attempts.

Internal signals for test-time control. Hidden states can reveal whether a reasoning model's answer or intermediate step is likely to be correct [24], and ReProbe turns them into a trained step verifier [14]. Chen et al. instead use sparse-neuron agreement for label-free best-of-N selection and early pruning [3]. These works use internal states to rank candidate quality. We complement them by mapping what sparse routing retains at action, trajectory, and attempt scales, then translating its role- and granularity-specific structure into coordination rules.

Test-time scaling and verification. Self-consistency aggregates repeated answer strings [20]; soft and universal variants compare likelihoods or use a separate language model to select free-form outputs [5, 19]. Compute-optimal and confidence-guided scaling further stress the selection signal [9, 18]. Our contribution is a native-routing coordination mechanism for repeated samples when long-horizon agents produce heterogeneous trajectories and non-canonical patches.

Agentic test-time scaling and software repair. SWE-bench Verified evaluates repository-level repair against hidden tests [6, 10]. Existing agents expose repository tools or decompose localization and repair [21, 23], while test-time scaling changes search, feedback, and trajectory reuse [1, 2, 7, 25]. Other systems summarize rollouts, train trajectory verifiers, or combine tool entropy with tests [11, 12, 16]. We complement these mechanisms with MoE-routing comparisons over recent history, same-prefix candidates, and separately sampled attempts. Terminal selection uses one same-model prefill per patch, but no trajectory summary, judge model, or additional candidate-patch execution for selection.

3 What Routing Reveals Across Agent Decisions

To turn this native MoE trace into control, we first need to understand what survives at each unit of an agent run. An action may be a short search command or a long patch-writing invocation; an attempt contains many such actions; and a task yields several independently sampled attempts. We ask four questions in sequence: what behavior survives aggregation over an action, whether an action repeats the trajectory's recent computation, where a patch carries its fine-grained routing variation, and which candidates may meaningfully agree. The answers become the role gate, history-based exploration rule, decision-token readout, and granularity-aware commitment rules.

3.1 From Router Rows to Comparable Fingerprints

To compare routing at these scales, we first need one common object. Our agent reads, searches, edits, and tests before submitting a patch. At each step the model proposes n sibling actions from the same prefix and commits to one. We call the resulting sequence of executed actions and its accumulated final diff an *attempt*; separately, K attempts produce the final patch candidates. In our configuration, n=16 candidate generations are sampled per step and K=4 attempts per task. The first choice set shares an immediate prefix, whereas the second can follow different repository paths. This distinction will determine what agreement means.

A raw router trace contains one sparse expert-weight vector for every token and layer, and candidate spans have different lengths. To make them comparable, we integrate the gate mass over a chosen span s and

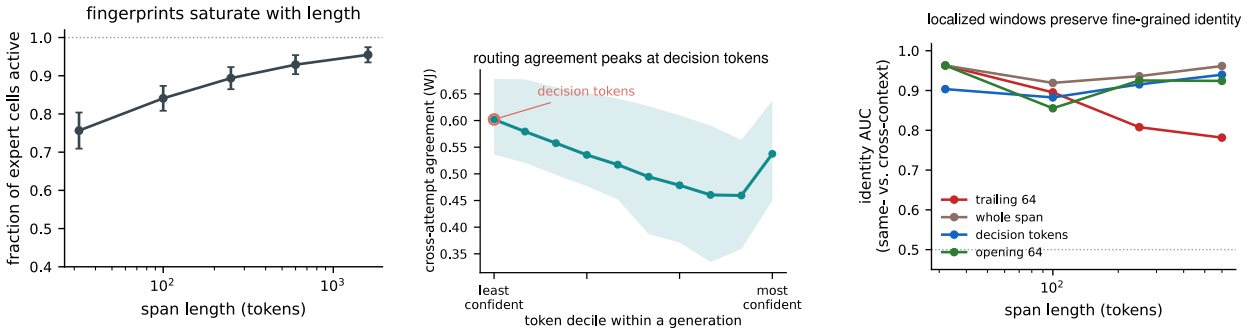

Figure 2 Saturation, decision-token localization, and window choice. Left: layer–expert occupancy (the fraction of cells receiving nonzero mass) grows with span length. Middle: splitting each generation into ten deciles by next-token probability, routing agreement between separately sampled attempts is highest in the lowest-probability decile and falls as probability rises (1,004 attempt pairs; mean and interquartile range); the final decile rebounds, consistent with highly predictable tokens being routed alike. Right: the context-identity AUC (same generation context versus different contexts) decays with length; whole-span routing retains coarse context, while opening and decision-token windows preserve finer separation.

normalize it into a layer-by-expert histogram. Let $w_{t,\ell,e}$ be the weight assigned by token $t \in s$ to expert e in layer ℓ, with zero for an expert that is not selected. The resulting *routing fingerprint* is

$$h(s)_{\ell,e} = \frac{\sum_{t \in s} w_{t,\ell,e}}{\sum_{t \in s} \sum_{\ell',e'} w_{t,\ell',e'}}. \tag{1}$$

Thus $h(s)$ records where the MoE allocated computation while removing span length. Since these fingerprints are nonnegative sparse profiles, we compare them with weighted Jaccard,

$$\mathrm{WJ}(u,v) = \frac{\sum_i \min(u_i, v_i)}{\sum_i \max(u_i, v_i)}, \tag{2}$$

where i indexes layer–expert cells. The score is one for identical routing mass and zero for disjoint mass. Action fingerprints are read from the original generation pass; accumulated final patches are re-encoded once in Section 4.2 so that the fingerprint represents the submitted artifact itself.

3.2 Insight 1: Routing Strongly Encodes Behavioral Role

We first ask what survives action-level aggregation. Figure 1a–b shows a clear geometry: writes from resolving and non-resolving branches occupy the same broad representation, while inspection, search, execution and patch writing separate cleanly. Using parsed tool calls to label the three roles that the scaffold's commands actually realize—inspect or execute, run tests, and write—we fit one routing centroid per role within each model configuration and assign the centroid with highest cosine similarity. On task-disjoint splits of 78,535 actions, three-way holdout accuracy is 0.940 against a 0.746 majority-class baseline; write-versus-rest recall is 0.93 at precision 1.00. For control, the two non-write roles are collapsed into an exploratory class, while write forms the commitment class. This distinction is operationally important because one sibling set can mix actions with different purposes: an ordinary read should not become preferable merely because reads are common when another candidate is ready to write a patch. Routing therefore supplies a high-precision behavioral gate that first makes candidates commensurable, then determines which comparison rule to invoke. Appendix A.3.2 gives the centroid data, holdout protocol, and write-cohort activation rule; Appendix D separately audits routing against exact parsed-tool gating.

3.3 Insight 2: Routing Complements Surface Similarity

We next ask what progress information routing contributes beyond surface form. Once an attempt has history, each executed action receives a repetition score: its mean WJ similarity to the three closest fingerprints among up to the previous 64 actions. Averaging these step scores within an attempt strongly tracks stagnation: resolved rate falls from 33.8% in the most-different quintile to 5.4% in the most-similar (n=1,021). Because the comparison is made in routing space, differently phrased commands can still count as a revisit when they allocate computation similarly. This trajectory-level signal motivates disagreement with recent routing history during exploration. Appendix J.5, with Figure 5, reports the quintile construction and contrasts the within-step and trajectory-level relationships.

At the patch scale, changed-line overlap and routing cover different cases. We score each diff by mean Jaccard overlap over its added and deleted file–line pairs, after removing trailing whitespace. This surface score ties at the top on 16.5%–47.4% of tasks across conditions, whereas routing still orders the candidates. Surface overlap is strongest when edits match explicitly; routing provides a complementary axis, motivating the hybrid selector in Table 2. Appendix A.4.2 defines the changed-line and file-set scores and reports their pairwise AUC and top-tie rates; Appendix C, especially Table 3, compares additional surface-centrality controls.

3.4 Insight 3: Decision Tokens Preserve Fine-Grained Routing Information

We then ask where in a long candidate the router should be read. As a span grows, layer–expert occupancy (the fraction of cells with nonzero mass) rises from 0.75 below 64 tokens to 0.96 beyond a thousand, so a whole-span fingerprint increasingly mixes many expert pathways. We quantify retained context with a *context-identity AUC*: the AUC for using WJ to distinguish fingerprint pairs produced under the same candidate-generation context from pairs produced under different contexts, where a context is the task-and-prefix state from which a candidate is sampled. On long outputs, whole-span fingerprints reach 0.88, decision-token fingerprints 0.93, and opening-window fingerprints 0.98; a trailing window falls to 0.58. Opening tokens preserve the originating context most strongly, but they precede many patch-specific choices. We therefore need a localized window that also coincides with the model's decisions. Appendix J.2 reports the full span-length saturation and context-identity controls.

The reason granularity matters is simple: a long patch contains many predictable tokens from diff syntax, formatting, and locally routine code. If all positions contribute equally, these common pathways can dominate the fingerprint even when the repair-defining choices differ. We therefore seek a compact set of positions where the model had to discriminate among plausible continuations, without assuming a task-specific delimiter.

Token probability provides that localization. Across 658 final-step generations, the least-probable quarter carries a median 72% (interquartile range 68–78) of total token surprisal $\sum_t -\log p_t$, rather than the 25% expected under equal contribution (Figure 1c). We call these positions *decision tokens*: an operational window that localizes routing variation without requiring a task-specific semantic label for each token. This concentration defines a compact, generation-agnostic window; matched readouts below test whether it preserves useful routing variation. For an accumulated final patch, analogous probabilities come from one teacher-forced re-encoding pass (Section 4.2). Mean cross-attempt routing agreement is 0.65 at decision tokens and 0.56 elsewhere; it generally falls with confidence before a final-decile rebound (Figure 2, middle).

Outcome variation also becomes visible at this scale. On write steps containing both outcomes, we score each candidate by mean WJ agreement with its siblings. Decision-token routing reaches AUC 0.69 on shorter candidates, above all other readouts (Table 1). Its advantage on compact branch decisions motivates the same localized readout across independent completed attempts.

Using each attempt's mean WJ agreement with the other attempts as its score, decision tokens provide the strongest attempt-ranking readout. In groups containing both outcomes, the pairwise AUC for ranking a resolving attempt above a non-resolving one is .657, compared with .639 for the whole span and .616–.634 for random, uniformly spaced, router-entropy, and highest-probability controls. Appendix J.1 enumerates this

	candidate length	
fingerprint read at	shorter half	longer half
whole span	.59	.52
opening 64 tokens	.49	.51
trailing 64 tokens	.51	.52
decision tokens	**.69**	.52

Table 1 Pairwise AUC for separating resolving from non-resolving sibling branches across 3,660 pairs from 423 mixed-outcome steps, split at the median candidate length. Decision tokens give the strongest readout, with their clearest advantage on shorter candidates.

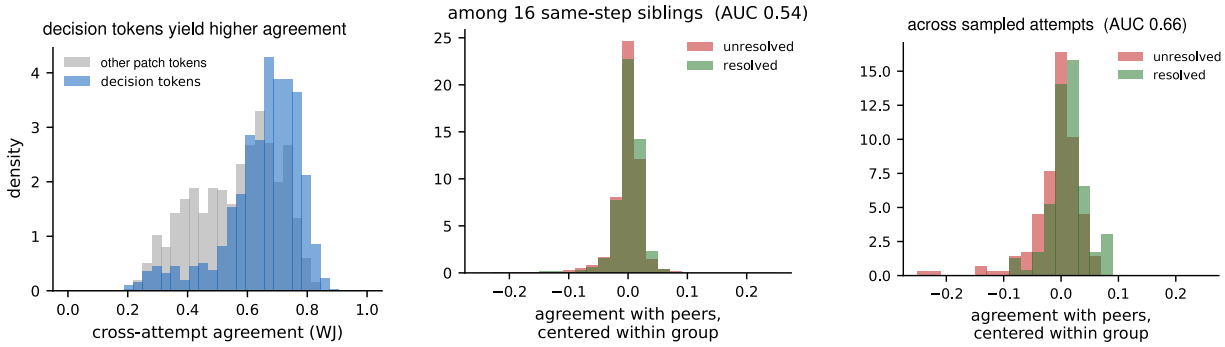

Figure 3 Why the reference set matters. Left: decision tokens yield higher cross-attempt agreement than other positions. Middle: same-prefix outcome distributions overlap, motivating history-relative novelty. Right: across separately sampled attempts, resolving patches receive higher mean agreement with their attempt group.

full control battery, while Appendix J.4 and Table 8 report window-fraction and neighborhood sensitivity. Fixed delimiters localize routing for canonical answers [4]; low token probability serves as the corresponding anchor for free-form patches.

3.5 Insight 4: Granularity Determines How Agreement Should Be Used

The final question is whose agreement is meaningful. Same-prefix siblings share most of their evidence, so their outcome-conditioned agreement distributions overlap (Figure 3, middle). Exploration therefore compares candidates with recent executed history (Section 3.3). Peer support enters only after the role gate forms a write cohort, where a pilot-fixed composite combines local support with routing-dispersion guards (Section 4.1). Appendix A.3.5 specifies the standardized score, fixed coefficients, and guard terms, while Appendix E, especially Table 4, ablates the individual components.

Across separately sampled attempts, agreement becomes a direct ranking signal. The trajectories follow different repository paths and share no immediate prefix; mean agreement between their decision-token fingerprints ranks resolving attempts above non-resolving ones with AUC 0.66 in mixed-outcome groups. Together, these regimes yield the operational rule in the title: disagree with recent computation to explore; use agreement only after role gating, and across trajectories to commit.

3.6 From Observations to Design

MoE routing is most useful here as a shared computational coordinate: the same layer–expert axes describe textually different actions and patches. Routing similarity derives its decision meaning from the chosen reference set. A high value can indicate stagnation against recent history, useful support among role-matched writes, or convergence across independent attempts. The controller must consequently change both the comparison set and the direction in which similarity is optimized. Against recent executed history, similarity

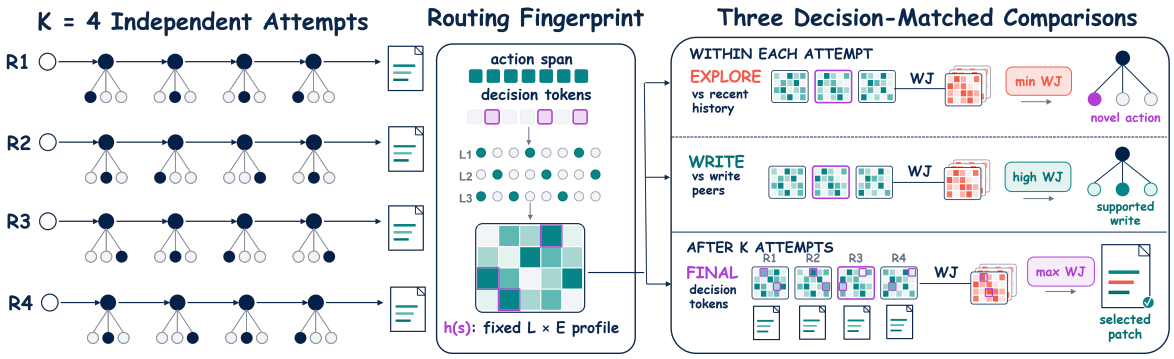

Figure 4 RISA: one MoE routing representation, three decision-matched comparisons. Exploratory actions are compared with recent executed history; patch-writing actions with their role-gated peers; and accumulated final patches with separately sampled attempts at decision tokens.

measures computational revisitation; within a role-matched write cohort, guarded centrality measures convergent commitment; across separately sampled attempts, decision-token centrality measures cross-path convergence. The fingerprint supplies the coordinate, while the comparison set supplies the semantics. Figure 4 turns these three relations into RISA.

4 Method: RISA

Inside each attempt, RISA repeats four steps: sample sibling generations, isolate and fingerprint their action spans, use the routing role gate to choose a comparison rule, and execute the selected action while updating history. After K complete attempts, RISA re-encodes the accumulated diffs and applies decision-token agreement once to choose the submission. The model weights remain fixed online, and the same MoE telemetry is reused at each scale by changing its reference set rather than training a separate scorer for every decision.

Step-level fingerprints come from the original generation. Three role centroids (inspect/execute, test, write) are fit per model configuration from parsed tool-call labels; at control time, each candidate is assigned to its highest-cosine centroid and the two non-write roles are grouped as exploratory. Architecture-specific centroids are fit from task-disjoint action labels, while the write-score coefficients are fixed on a separate pilot. Both are frozen before evaluation; role definitions, decision logic, and coefficients are shared across tasks and effort settings. The behavioral gate assigns candidates to the comparison rule suited to their role; the relation-specific score then ranks them. Parsed calls provide convenient supervision, while routing provides the portable runtime signal: 97.3% of writes here are shell commands inside a generic execution tool, whose runtime parsing requires a scaffold-specific command taxonomy. Reading routing keeps both levels on one signal and agrees with the exact parsed role on 96.8% of actions. Appendix D.2 details the command taxonomy required by the parsed gate, and Appendix D.3 reports the controlled gate-substitution test.

4.1 Steering a Single Attempt

At step t, the model samples n=16 candidate generations C_t in one batched call. A generation may contain reasoning followed by a proposed tool invocation; we fingerprint only the *action span* that serializes the invocation, located from the model's action-channel markers, so the comparison describes the proposed action rather than its preceding reasoning. We write $h(c)$ for this fingerprint. The history $\mathcal{H}_t$ stores the last W=64 selected and executed action fingerprints. Appendix A.3.2 specifies the span fallbacks: the trailing 64 generated tokens are used when action markers are missing, and accumulation becomes unweighted when gate weights are unavailable.

The role prediction is collapsed into write versus exploratory. A cohort of at least two writes activates the

commitment rule, because peer agreement is only defined when there is another role-matched proposal to support it; otherwise all candidates use the history-disagreement rule. The selected action is then executed and its fingerprint appended to $\mathcal{H}_{t+1}$.

Exploration: disagree with recent history. When the write rule is inactive and history is nonempty, set $m_t = \min(3, |\mathcal{H}_t|)$ and score

$$S_{\text{explore}}(c) = -\frac{1}{m_t} \sum_{u \in \mathcal{N}_{m_t}(c)} \text{WJ}\big(h(c), u\big), \tag{3}$$

where $\mathcal{N}_{m_t}(c)$ contains the m_t recent-history entries most similar to c; the maximum is selected. Closest matches expose repeated computation even when the rest of the history is unrelated. With no history, the controller uses a deterministic cold-start fallback. Appendix E.1 specifies that it uses the sibling routing medoid at the first eligible step and falls back to the first exchangeable sample only when no fingerprint is available. Thereafter, Equation 3 measures novelty relative to the attempt's own executed computation, not to same-prefix siblings.

Patch writing: commit toward peer agreement. Once several candidates propose writes, continuing to maximize novelty can delay a patch that multiple branches have independently supported. At the same time, blindly taking the global medoid may favor an uninformative routing concentration. We therefore combine local support and cohort-wide similarity with two dispersion guards. Let C_t^w be the role-gated write cohort. For $|C_t^w| \geq 2$, local peer support is the fraction of other write candidates within the fixed routing-similarity threshold τ=0.65:

$$M(c) = \frac{1}{|C_t^w| - 1} \sum_{\substack{c' \in C_t^w \\ c' \neq c}} \mathbf{1}\big[\text{WJ}\big(h(c), h(c')\big) \geq \tau\big]. \tag{4}$$

The score also uses mean peer similarity $\bar{S}(c) = (|C_t^w| - 1)^{-1} \sum_{c' \in C_t^w,\, c' \neq c} \text{WJ}(h(c), h(c'))$, fingerprint entropy $H(c) = -\sum_i h_i(c) \log h_i(c)$, and peak mass $P(c) = \max_i h_i(c)$, where i indexes layer–expert cells. Each statistic is standardized within the current write cohort as $z(x) = (x - \mu_x)/\sigma_x$ (a zero-variance statistic contributes zero), and the controller selects the maximizer of

$$S_{\text{write}}(c) = z\big(H(c)\big) + \tfrac{3}{2}\, z\big(M(c)\big) + z\big(\bar{S}(c)\big) - z\big(P(c)\big). \tag{5}$$

Here M captures local support, $\bar{S}$ cohort centrality, and H and P guard against a concentrated routing spike. The coefficients were fixed once on a disjoint pilot and are shared across tasks, models, and effort settings; commitment therefore rewards broad, role-consistent convergence rather than unconditional sibling majority.

4.2 Arbitrating Among Multiple Attempts

Each task runs K=4 attempts; let $\mathcal{A}$ index those ending with a nonempty accumulated final diff. Because later edits can overwrite earlier ones, no single generated action span necessarily represents the submitted artifact. We therefore fingerprint the artifact directly. For $k \in \mathcal{A}$, the task statement concatenated with final diff p_k is re-encoded in one teacher-forced pass, yielding a routing row and next-token probability for every patch token. This pass consumes the fixed patch without sampling new text, so all candidates are measured under the same task-statement context. Let D_k be its least-probable quarter; aggregating Equation 1 over D_k gives the decision-token fingerprint $h_{\text{dec}}(p_k)$. All attempts use the same task-statement prefix.

When $|\mathcal{A}| \geq 2$, the submitted patch maximizes mean agreement with the other separately sampled attempts:

$$\hat{k} \;=\; \arg\max_{k \in \mathcal{A}} \; \frac{1}{|\mathcal{A}| - 1} \sum_{\substack{j \in \mathcal{A} \\ j \neq k}} \text{WJ}\big(h_{\text{dec}}(p_k), h_{\text{dec}}(p_j)\big). \tag{6}$$

This surface-overlap-free rule selects the decision-token routing medoid of the available patches. A sole patch

			Attempt pool		Terminal arbitration				Upper bound
Model	Effort	Eligible	Yield	Steps	Uniform	Text	RISA	RISA-H	Oracle
gpt-oss-20b	low	496	79.1	9.5	31.8	34.5 (+2.7)	34.1 (+2.3)	**34.7** (+2.9)	48.4
	medium	498	94.8	26.5	45.7	47.8 (+2.1)	**48.2** (+2.5)	47.6 (+1.9)	63.3
	high	498	80.0	43.1	45.3	49.2 (+3.9)	**51.0** (+5.7)	50.0 (+4.7)	62.2
gpt-oss-120b	low	497	93.0	14.8	38.7	40.0 (+1.3)	**41.2** (+2.5)	40.4 (+1.7)	53.3
	medium	497	99.5	28.8	50.6	54.5 (+3.9)	53.5 (+2.9)	**54.9** (+4.3)	66.6
	high	498	94.6	41.9	57.2	62.0 (+4.8)	61.2 (+4.0)	**62.4** (+5.2)	71.5
MACRO AVG. (gpt-oss)		—	90.2	27.4	44.9	48.0 (+3.1)	48.2 (+3.3)	**48.3** (+3.5)	60.9
Qwen3.6-35B-A3B	default	498	73.5	47.5	41.7	45.0 (+3.3)	45.2 (+3.5)	**45.6** (+3.9)	50.6

Table 2 Final-patch arbitration on SWE-bench Verified (% resolved) over common RISA-steered K=4 pools. Rates condition on at least two graded patches. The full 500-task `gpt-oss` grid yields 496–498 eligible tasks per condition; the full 500-task `Qwen3.6-35B-A3B` condition has 498. RISA-H uses Text-first routing tie-breaking; Uniform is expected random choice and Oracle the success union. Parentheses in deployable columns show gains over Uniform; bold shading and underlining mark the best and second-best deployable estimates.

is submitted directly and an empty pool is unresolved. Selection adds one teacher-forced prefill per available patch but no candidate generation or execution; Appendix I.1 quantifies this terminal overhead alongside the batched step-generation cost.

5 Experiments

Setup. We evaluate on SWE-bench Verified with `gpt-oss-20b` and `gpt-oss-120b` at their native low, medium, and high reasoning-effort settings. Each step considers n=16 siblings and each task runs K=4 attempts. Official grading yields task-level resolved rate. In Table 2, *Yield* is the fraction of completed attempts ending with a nonempty final diff and *Steps* the mean executed actions per attempt. Selector rates use 496–498 tasks per condition with at least two graded patches; every terminal rule receives the same RISA-steered pool. Appendix A.5 details the shared environmental exclusions and selector denominator, while Appendix H.1 records how the final budgets and routing-readout settings were chosen.

Terminal arbitration. All rules act on the same RISA-steered pools. *Uniform* is expected random choice; *Text* maximizes mean Jaccard over added/deleted file–line pairs after trimming trailing whitespace. RISA uses Equation 6; RISA-H is Text-first with routing tie-breaking. *Oracle* is the attempt-pool success union. Appendix A.4.3 specifies the score tolerances, missing-fingerprint behavior, and fixed-index tie rules used by all selectors.

5.1 Main Results

Across the six `gpt-oss` conditions, RISA gains 2.3–5.7 points over Uniform, lifting the macro-average from 44.9% to 48.2%; the largest gains occur at high reasoning effort. RISA is best in three conditions and RISA-H in the other three, with macro-averages of 48.2% and 48.3%, respectively, versus 48.0% for Text. The 60.9% Oracle confirms substantial complementary coverage across the four-attempt pools. Appendix B reports the task-paired bootstrap intervals and McNemar tests; Appendix C, especially Table 3, compares alternative surface and routing centrality rules.

The same rules transfer to `Qwen3.6-35B-A3B`; only its architecture-dependent role centroids are refit from task-disjoint action labels. On the 498 eligible tasks, RISA resolves 45.2%, versus 41.7% for Uniform and 45.0% for Text; RISA-H reaches 45.6%. Both RISA and Text significantly outperform Uniform ($p < 0.001$). RISA matches Text, with 10 routing-only wins and 9 text-only wins (exact McNemar p=1.000). This cross-family condition is reported separately from the `gpt-oss` macro-average in Appendix G, Table 6.

5.2 Analysis

The fixed-pool comparison isolates arbitration. Every selector in Table 2 receives the same RISA-steered attempts, so the differences isolate terminal arbitration from candidate generation and step budget. Routing improves over Uniform in every reported condition, and the gains persist as Yield ranges from 79.1% to 99.5%, showing robustness across candidate availability.

Routing matches and complements surface consensus. RISA is best in three `gpt-oss` conditions and RISA-H in the other three; their 48.2% and 48.3% macro-averages edge Text's 48.0%, supporting complementarity on the `gpt-oss` grid. On the full Qwen condition, RISA matches Text (45.2% versus 45.0%; exact McNemar p=1.000). The cross-family result shows that routing-only arbitration maintains text-consensus performance while operating directly on internal routing traces.

Decision tokens provide the strongest terminal readout. As Section 3.4 shows, the least-probable quarter ranks mixed-outcome attempts better than the whole span and matched controls, motivating the terminal readout. Applied to identical attempt pools, this localized representation yields gains in every reported condition, and the same decision-token rule transfers to Qwen as a significant 3.5-point gain over Uniform. The token-level analysis thus connects directly to final-patch arbitration.

Supporting diagnostics. Table 2 deliberately holds generation fixed. Appendix J.6 isolates steering through patch-yield coverage on the empirically hard 80-instance set, while Appendix F, Table 5, decomposes steering and terminal arbitration on a fixed 200-task pool. On the hard set, steering raises submittable-patch yield from 79% to 94%; on a fixed 200-task `20b` subset, the full pipeline reaches 50.5% versus 45.4% for unguided generation with Uniform selection. Routing also improves selection on pools generated without steering, showing that steering expands candidate availability while arbitration contributes independently.

6 Conclusion

Sparse-MoE routing gives software agents a reference-dependent computational coordinate for coordinating nested decisions. This view yields RISA: novelty against recent history guides exploration, guarded peer support guides writing, and decision-token agreement across independently developed patches guides final selection. Across six `gpt-oss` conditions, routing arbitration gains 2.3–5.7 points over Uniform, reaching a 48.2% macro-average versus 48.0% for Text. On the full Qwen benchmark, it reaches 45.2% versus 41.7% for Uniform and 45.0% for Text, transferring as a significant gain over uniform selection while matching text consensus. Together, these results establish routing as a practical coordination signal that transfers across MoE families.

7 Limitations

RISA's current instantiation assumes accessible sparse-MoE routing and repeated trajectories, making it naturally suited to white-box MoE agents. The broader principle is to construct a shared internal coordinate, align it with behavioral roles, and choose reference sets that match the decision. Extending this principle to dense or closed models will require alternative readouts, while other domains will need role definitions suited to their action spaces. Combining routing coordination with semantic or execution-based evidence is a promising direction for handling rare but valuable outlier repairs.

References

[1] Vaibhav Aggarwal, Ojasv Kamal, Abhinav Japesh, Zhijing Jin, and Bernhard Schölkopf. DARS: Dynamic action resampling to enhance coding agent performance by adaptive tree traversal. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19808–19855. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.973. URL https://aclanthology.org/2025.acl-long.973/.

[2] Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. SWE-Search: Enhancing software agents with monte carlo tree search and iterative refinement. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/a1e6783e4d739196cad3336f12d402bf-Abstract-Conference.html.

[3] Kang Chen, Yaoning Wang, Kai Xiong, Zhuoka Feng, Minshen Yu, Wenhe Sun, Haotian Chen, and Yixin Cao. Do LLMs signal when they're right? evidence from neuron agreement. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=ZVRgcQq4D2. Spotlight.

[4] Kang Chen, Minshen Yu, Junjie Nian, Yaoning Wang, Yixin Cao, and Yugang Jiang. Does the same token mean the same state? MoE routing as signal for reasoning control, 2026. URL https://arxiv.org/abs/2606.22798.

[5] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal Self-Consistency for large language model generation, 2023. URL https://arxiv.org/abs/2311.17311.

[6] Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench verified. https://openai.com/index/introducing-swe-bench-verified/, 2024.

[7] Yifeng Ding and Lingming Zhang. SWE-Replay: Efficient test-time scaling for software engineering agents, 2026. URL https://arxiv.org/abs/2601.22129.

[8] 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. URL https://jmlr.org/papers/v23/21-0998.html.

[9] Yichao Fu, Xuewei Wang, Hao Zhang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8LqHs0KIM7.

[10] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

[11] Joongwon Kim, Wannan Yang, Kelvin Niu, Hongming Zhang, Yun Zhu, Eryk Helenowski, Ruan Silva, Zhengxing Chen, Srinivasan Iyer, Manzil Zaheer, Daniel Fried, Hannaneh Hajishirzi, Sanjeev Arora, Gabriel Synnaeve, Ruslan Salakhutdinov, and Anirudh Goyal. Scaling test-time compute for agentic coding, 2026. URL https://arxiv.org/abs/2604.16529.

[12] Chenhui Mao, Yuanting Lei, Zhixiang Wei, Ming Liang, Zhixiang Wang, Jingxuan Xu, Dajun Chen, Wei Jiang, and Yong Li. EGSS: Entropy-guided stepwise scaling for reliable software engineering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29481–29499. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1359. URL https://aclanthology.org/2026.acl-long.1359/.

[13] Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoE: Open mixture-of-experts language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xXTkbTBmqq.

[14] Jingwei Ni, Ekaterina Fadeeva, Tianyi Wu, Mubashara Akhtar, Jiaheng Zhang, Elliott Ash, Markus Leippold, Timothy Baldwin, See-Kiong Ng, Artem Shelmanov, and Mrinmaya Sachan. ReProbe: Efficient test-time scaling of multi-step reasoning by probing internal states of large language models. In *Proceedings of the 64th Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), pages 11667–11689. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.536. URL https://aclanthology.org/2026.acl-long.536/.

[15] Matthew Lyle Olson, Neale Ratzlaff, Musashi Hinck, Man Luo, Sungduk Yu, Chendi Xue, and Vasudev Lal. Probing semantic routing in large mixture-of-expert models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 18263–18278. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.991. URL https://aclanthology.org/2025.findings-emnlp.991/.

[16] Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 47717–47737. PMLR, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

[17] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

[18] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

[19] Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. Soft Self-Consistency improves language model agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 287–301. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-short.28. URL https://aclanthology.org/2024.acl-short.28/.

[20] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

[21] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying LLM-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE):801–824, 2025. doi: 10.1145/3715754. URL https://doi.org/10.1145/3715754.

[22] Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. OpenMoE: An early effort on open mixture-of-experts language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 55625–55655. PMLR, 2024. URL https://proceedings.mlr.press/v235/xue24c.html.

[23] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R. Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

[24] Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=O6I0Av7683.

[25] King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling test-time compute for LLM agents, 2025. URL https://arxiv.org/abs/2506.12928.

A Implementation Details

A.1 Models and Serving

The completed main grid uses the open-weight sparse mixture-of-experts models `gpt-oss-20b` (24 transformer blocks, 32 experts per MoE layer, top-4 routing, ~3.6B active parameters) and `gpt-oss-120b` (36 blocks, 128 experts, top-4 routing, ~5.1B active parameters), served with vLLM 0.15.1 through the raw `/v1/completions` endpoint. The cross-family condition uses `Qwen3.6-35B-A3B` (40 blocks, 256 experts,

top-8 routing, ~3B active parameters) on the same stack, and is reported separately from the `gpt-oss` grid. For the `gpt-oss` grid, conversations are rendered in the native Harmony token format and parsed back into analysis / tool / final channels; tools are declared in the system message. The serving stack returns, for every generated request, the per-token expert assignments (`routed_experts`, an integer tensor of shape $[T, L, R]$ for T tokens, L MoE layers, and R=4 routing slots) and the corresponding post-softmax gate weights (`routed_experts_weights`, same shape, rows summing to one). Both are recorded at generation time for every candidate action, together with next-token probabilities. Step-level steering needs no additional model pass; terminal selection adds one teacher-forced prefill per available final patch (Section A.3.6).

A.2 Agent Scaffold

The agent is a single-model tool-use loop over an isolated Docker container per task (SWE-bench instance image; the model edits the repository at `/testbed` and the final `git diff` is the submitted patch). Available tools are the model's in-distribution set: `container.exec`, `repo_browser.*` (search, tree, open file), and `apply_patch`. At every step the model proposes n=16 candidate actions sampled in parallel from the identical prefix; one candidate is selected by the exploration or write rule (Sections A.3.4 and A.3.5) and executed. At an explicit submission or budget exhaustion, the current `git diff` is retained as the final patch. Each task runs K=4 separately sampled trajectories.

A.3 Routing Fingerprints and RISA Rules

Throughout, RISA (*Routing-Informed Steering and Arbitration*) denotes routing-guided steering plus routing-only terminal arbitration; RISA-H keeps the same steering but uses the Text-first routing tie-breaker.

A.3.1 Fingerprint

For a token span S of a candidate, the fingerprint $h \in \mathbb{R}^{L \times E}$ accumulates gate mass,

$$h[l, e] \;=\; \sum_{t \in S} \sum_{q=1}^{R} w_{t,l,q}\, \mathbf{1}[\, r_{t,l,q} = e \,],$$

where $r_{t,l,q}$ is the expert index and $w_{t,l,q}$ its gate weight; h is L_1-normalized. Similarity between fingerprints is the weighted Jaccard (Ruzicka) score $\mathrm{WJ}(a, b) = \sum_i \min(a_i, b_i) / \sum_i \max(a_i, b_i)$, where i indexes layer–expert cells.

A.3.2 Candidate Span and Role Gate

A candidate generation may contain reasoning followed by a proposed tool invocation. Its *action span* is the token segment that serializes the final invocation, delimited by the model's action-channel marker tokens rather than decoded-string matching. If the markers cannot be located, the span falls back to the trailing 64 generated tokens; if gate weights are unavailable, accumulation is unweighted. We fit three centroids from actions labeled by parsed tool calls: inspect/execute (repository browsing, search and other shell commands, 53,923 actions), test (test-runner commands, 8,183) and write (patch application or an editing shell command, 16,429). Reading and searching are issued through the same generic execution tool as other commands and are not separated further, since only the write/non-write boundary gates a rule. At control time, the role whose centroid has highest cosine similarity to the candidate's routing fingerprint is predicted; the two non-write roles are collapsed to the exploratory class. Parsed labels supervise the centroids, while the deployed gate uses the routing prediction. Since $L \times E$ differs by architecture, centroids are fit separately within each model configuration using the same role definition. On task-disjoint 70/30 splits of 78,535 actions, three-way holdout accuracy is 0.940 against a 0.746 majority floor (the released centroid file carries exactly these three labels); write-versus-rest recall is 0.93 at precision 1.00. At least two predicted writes are required to activate commitment; otherwise all candidates remain under the exploration rule.

A.3.3 Diagnostic Branch Labels

For branch-level analyses, a write candidate carries the downstream official resolved/unresolved label of its evaluated branch. A *mixed-outcome step* contains at least one branch of each label. Pairwise area under the ROC curve (AUC) treats a resolved–unresolved pair as correctly ordered when the score ranks the resolving branch higher; the "resolved" and "unresolved" labels in main-paper figures refer to these downstream outcomes.

A.3.4 Exploration Steps: Disagree

Each candidate's action fingerprint is compared with the agent's own recently executed actions. With $m = \min(3, |\mathcal{H}_t|)$, we average the m largest similarities and select the candidate whose average is lowest; using the closest matches prevents one repeated computation from being masked by unrelated older entries. The history pool contains up to the last W=64 executed-action fingerprints, with no role matching. If history is empty, the sibling routing medoid—the candidate with greatest mean WJ to the other siblings—is used as a deterministic cold-start fallback, not as an outcome-ranking rule; if candidate fingerprints are unavailable, the rule samples uniformly.

A.3.5 Write Steps: Guarded Peer Support

Let C_t^w be the predicted write cohort. If $|C_t^w| \geq 2$, local peer support is

$$M(c) = \frac{1}{|C_t^w| - 1} \sum_{c' \neq c} \mathbf{1}[\mathrm{WJ}(h(c), h(c')) \geq \tau],$$

with τ=0.65. The other terms are fingerprint entropy $H(c) = -\sum_i h_i(c) \log h_i(c)$, mean peer similarity $\bar{S}(c) = (|C_t^w| - 1)^{-1} \sum_{c' \neq c} \mathrm{WJ}(h(c), h(c'))$, and largest layer–expert-cell mass $P(c) = \max_i h_i(c)$. Each statistic is standardized over the current cohort as $z(x) = (x - \mu_x)/\sigma_x$; a zero-variance term contributes zero. The fixed score is

$$S_{\text{write}}(c) = z(H(c)) + 1.5\, z(M(c)) + z(\bar{S}(c)) - z(P(c)),$$

and the maximizer is selected. Here M captures a local agreement neighborhood and $\bar{S}$ global centrality, while H and P guard against agreement caused by a highly concentrated routing spike. The coefficients $(1, 1.5, 1, -1)$ are constants in the released configuration: they are not refit per task, model, or effort setting, and no per-run optimization takes place. Their signs and relative magnitude were fixed once, before the main campaign, from a pilot study of 98 mixed-outcome write steps (650 resolved–unresolved candidate pairs) in which this combination reached a step-clustered bootstrap AUC of 0.592 (95% confidence interval $[0.512, 0.672]$) for ranking the resolving branch higher; that pilot pool is disjoint from the tasks used in the ablations reported here. The pilot-fitted scorer remains fixed throughout evaluation. Fewer than two predicted writes returns the full candidate set to the exploration rule.

A.3.6 RISA Arbitration Across Multiple Attempts

An available final patch is a nonempty accumulated `git diff`, which may combine hunks written at several steps, some overwriting earlier edits. We therefore re-encode each complete final patch once: the task statement concatenated with the patch is sent as a single prompt with `max_tokens`=1 and `temperature`=0, and the server returns a routing row and a teacher-forced token probability for every prompt position; the required one-token completion is discarded. Only positions inside the patch segment are used; the offset of that segment is the token length of the task statement, measured by the same request. Positions in the lowest probability quartile of the patch segment (the *decision tokens*) contribute their gate-weighted routing rows to the attempt fingerprint. The probabilities are the model's teacher-forced next-token probabilities before any sampling transform, so they do not depend on the temperature or nucleus setting used during the rollout, and all K attempts of a task are encoded under an identical prefix. The submitted attempt has the highest mean WJ agreement with the other available attempts; a single available patch is submitted directly, and a pool with no patch is unresolved. Terminal selection costs one prefill plus one discarded token per available

patch; it neither re-executes a patch nor generates an additional patch. Fractions from 10% to 50% form a stable operating region, changing cross-attempt ranking AUC by at most 0.01.

A.4 Diagnostic Pool, Surface Scores, and Tie-Breaking

A.4.1 Hard Diagnostic Pool

The 80-task pool used for the within-trajectory policy comparisons is defined by *empirical* difficulty measured from agent behavior. In a pilot phase each Verified task was attempted six times, giving a solve count in $\{0, \ldots, 6\}$; from that ranking we took every task solved once (55 tasks), a seeded random 10 with zero solves, and a seeded random 31 of those solved twice, for 96 tasks, then made a difficulty-stratified 80/16 split whose 80-task side is the pool used here (`seed`=42, both lists released with the code). Its composition is 36 `django`, 13 `sympy`, 11 `sphinx-doc`, 5 `matplotlib`, and nine or fewer each from five other repositories. Its benchmark-label distribution is 23 `<15 min`, 48 `15 min-1 hour`, 8 `1-4 hours`, 1 `>4 hours`. Its behavioral criterion selects tasks solved at most twice in six attempts, producing a focused diagnostic where uniform sibling choice resolves 15.9%.

A.4.2 Surface Scores

All text comparisons use one of two sets extracted from a unified diff. The changed-line set $H(p)$ contains one element per added or deleted line, namely the pair (target file, line content with trailing whitespace removed); target files are read from `+++ b/` headers and the `---`/`+++` header lines themselves are skipped. The touched-file set $F(p)$ contains the targets of `+++ b/` and `diff -git` headers. Both are compared with the Jaccard index $J(A, B) = |A \cap B|/|A \cup B|$, defined as 0 when both sets are empty, so two empty patches are not treated as agreeing. The Text selector scores each attempt by the mean $J(H(p_k), H(p_j))$ over the other available attempts; the file-level variant substitutes F. "Surface-overlap ties" in Insight 2 means that this mean is identical for the top two attempts to within 10^{-6}. On resolving–non-resolving cross-attempt pairs with unequal surface scores, changed-line centrality reaches pairwise AUC .64; over the complete pair set, ties reduce it to .52, while routing retains .58. Depending on the model–effort condition, changed-line centrality ties at the top on 16.5%–47.4% of eligible tasks.

A.4.3 Tie-Breaking

Attempts are ordered by a fixed arm index, and every selector takes the first attempt whose score exceeds the running maximum by more than 10^{-9}; ties therefore resolve to the lowest arm index, which is independent of the patches and of the routing. An attempt whose fingerprint is missing is scored $-\infty$ and is chosen only if no attempt has one. Risa-H first computes the Text score; if the top two attempts are within 10^{-6} it re-ranks the tied set by routing agreement under the same first-index rule, and otherwise returns the Text choice. Within a trajectory, the step-level rules use `argmax` over the candidate list in generation order, so ties there resolve to the lowest candidate index.

A.5 Grading Protocol

Patches are graded with the official SWE-bench harness (fail-to-pass and pass-to-pass tests inside a fresh instance container). We grade every stored final patch regardless of the agent's exit status. All terminal selectors within a model–effort condition use the same evaluable task and attempt pool.

The shared main-table eligibility rule yields 496–498 tasks per model–effort condition. Two Verified instances (`astropy-8707`, `astropy-8872`) encounter the same reproducible image-build incompatibility in their pinned 2019 build stack, including with unrestricted network access. Patchless attempts remain represented in the Yield column and count as unresolved in attempt-level analyses, while terminal selectors operate on available candidate patches. Selector comparisons include tasks with at least two graded patches, yielding up to two additional eligibility differences per condition. Shared eligibility preserves the paired comparisons; using all 500 tasks changes absolute rates uniformly by at most 0.5 points.

Terminal selector	Macro (%)	vs. Risa
Uniform	44.9	—
patch-length prior	46.7	−1.47 [−2.54, −0.37]
file-set Jaccard	46.9	−1.27 [−2.38, −0.13]
changed-line multiset	46.9	−1.34 [−2.48, −0.17]
TF–IDF cosine centrality	47.5	−0.74 [−1.71, +0.23]
character 3-gram Jaccard	47.5	−0.70 [−1.71, +0.33]
changed-line Jaccard (Text)	48.0	−0.20 [−1.17, +0.84]
routing concentration (single candidate)	45.6	−2.64 [−3.92, −1.31]
Risa routing agreement	**48.2**	—
Oracle	60.9	—

Table 3 Terminal selection-rule ablations on the six `gpt-oss` conditions (496–498 eligible tasks each; macro-average of the six). The right column is the paired difference against Risa, with bootstrap resampling clustered on SWE-bench task. Surface variants change how patch centrality is measured; routing concentration uses the identical fingerprint but no cross-attempt comparison. The 2.64-point separation identifies cross-attempt agreement, rather than intrinsic concentration, as the useful routing signal.

B Statistical Tests

All selector comparisons are paired: every selector acts on the same K attempts of the same task, so the unit of analysis is the task. Against Uniform we compute, per task, the difference between the selector's outcome and the expected outcome of choosing one available candidate patch uniformly, and bootstrap that difference with 10^4 resamples; selectors are compared with each other by McNemar's exact test on the discordant tasks.

The 2,984 pooled rows reuse the same 498 task IDs across up to six model–effort conditions rather than providing 2,984 independent observations. Pooled resampling is therefore clustered on the task: a bootstrap replicate draws tasks with replacement and keeps all available conditions for each drawn task. The clustered and unclustered intervals align closely (Risa−Uniform [+2.46, +4.19] clustered versus [+2.47, +4.16] unclustered), because the quantity being resampled is a *difference* against the same task's uniform expectation, which is far less correlated across conditions than the accuracies themselves. Each condition contains every task at most once, so per-condition intervals need no clustering.

Per condition, Risa−Uniform is +2.3, +2.5, +5.7 points for `gpt-oss-20b` at low, medium and high effort and +2.5, +2.9, +4.0 points for `gpt-oss-120b`, with unadjusted p from .042 to $< .001$. These condition-level p-values provide descriptive arm-wise summaries, while the task-clustered bootstrap gives the aggregate cross-condition estimate. Text and Risa-H reach +3.12 and +3.46 points over Uniform pooled, against +3.33 for Risa. Risa versus Text is +0.20 points with task-clustered bootstrap CI [−0.80, +1.21]; pooled cell-level McNemar values summarize the recurring task IDs across conditions.

C Terminal Selection-Rule Ablations

Table 3 tests whether the terminal result depends on the particular changed-line score used for Text, and separates cross-attempt routing agreement from a single-candidate routing-concentration score. The surface variants add sensitivity to order, repetition, token identity, file structure, or patch size. Every rule makes a realized 1-of-K choice on the same pools under the main-paper eligibility and grading convention.

D Role Gate: Routing versus Parsed Tool Calls

A natural question is why the write gate reads routing at all, when a candidate ends in a tool invocation that can be parsed. We replay both gates offline on the recorded candidate pools, holding everything downstream identical.

D.1 Agreement

Over 130,464 candidate actions, the deployed routing gate and an exact parsed-role gate assign the same write/non-write label 96.8% of the time, and 99.8% of candidates contain a parseable invocation. The two gates are therefore near-interchangeable in labeling accuracy.

D.2 What the Parse Actually Requires

Write actions span multiple invocation forms: of 54,855 write actions, 97.3% are shell commands issued through the generic execution tool (`sed -i`, a heredoc, `git apply`, an `apply_patch` invocation), and 2.7% arrive as a dedicated patch tool. The routing gate provides a portable alternative to a scaffold-specific command taxonomy: it is fit once per model configuration from the same labels and reads telemetry already produced during inference.

D.3 Substitution Test

On the 423 mixed-outcome write steps of the strong-label bank, we replace the gate and keep the write score, the cohort statistics, and the evaluation identical. Selecting within the routing cohort resolves 50.5% of the step-level branches against a same-step uniform baseline of 49.2% (+1.3 points, step-bootstrap CI [−2.5, +5.5]); the parsed cohort gives +0.7 points ([−3.6, +5.0]) and the ungated cohort +1.2 points ([−3.0, +5.4]). These aligned outcomes establish stability across gate implementations; the deployed system uses routing to keep both control levels on one telemetry source without a scaffold-specific parser.

E Write-Score Components

The write score $S_{\text{write}} = z(H) + 1.5\, z(M) + z(\bar{S}) - z(P)$ has fixed coefficients, so its terms can be audited directly. Table 4 reports step-internal pairwise AUC on 423 mixed-outcome steps (2,636 outcome-labeled candidates), with step-clustered bootstrap intervals; all statistics are computed exactly as at control time.

Score	AUC (95% CI)
peer similarity $\bar{S}$ alone	**.558** [.522, .592]
mode fraction M alone	.523 [.506, .541]
entropy H alone	.512 [.478, .547]
concentration $-P$ alone	.486 [.450, .521]
deployed $z(H)+1.5z(M)+z(\bar{S})-z(P)$	.520 [.486, .555]
equal weights	.522 [.486, .556]
without H	.513 [.477, .547]
without M	.518 [.483, .552]
without $\bar{S}$	.493 [.462, .528]
without P	.541 [.507, .574]

Table 4 Components of the write score on the strong-label bank. Mean peer similarity $\bar{S}$ provides the clearest standalone signal, and removing it produces the largest degradation among the combination ablations. The H and P terms act as stability guards; all coefficients were frozen before the campaign and remain fixed across tasks, models, and effort settings.

E.1 Cold-Start Fallback

When a trajectory has no executed history, the exploration rule falls back to the sibling routing medoid. This deterministic rule fires once per attempt, at its first step, and only when that step is not already a write step: it never ranks a write cohort or a final patch. If no fingerprint is available the rule returns the first sampled candidate, which is exchangeable with a uniform draw because the siblings are sampled independently from the same prefix.

F Pipeline Composition and Sampling-Budget Transfer

Table 5 evaluates the deployed system end to end and separately tests whether terminal selection transfers to a much smaller candidate-generation budget. Both pools use K=4 separately sampled attempts on the same fixed 200-task subset and the same grading convention; they differ only in how each step is generated. The *steered* pool samples n=16 candidate actions per step and runs the two-level controller; the *uncontrolled* pool samples one action per step and executes it, so its candidate-token cost is roughly 1/16 of the steered pool at equal K (prefix caching makes the wall-clock ratio smaller). "Uncontrolled + Uniform" is therefore the no-controller pipeline; steered + Routing is Risa, while steered + Hybrid is Risa-H.

Model	Pool	Unif.	Text	Rout.	Hyb.	Oracle
20b med	steered	45.5	49.5	**50.5**	50.0	63.5
	uncontrolled	45.4	50.0	48.0	**51.0**	63.0
120b med	steered	49.9	54.3	53.3	**55.3**	65.3
	uncontrolled	53.5	56.3	54.8	**55.3**	71.9

Table 5 Pipeline composition and sampling-budget comparison on the fixed 200-task subset (% resolved, paper grading convention; 200 tasks for 20b and 199 with four completed attempts for 120b). End to end, the Risa (steered + Routing) reaches 50.5% against 45.4% for the no-controller pipeline (uncontrolled + Uniform) at 20b, a task-paired bootstrap difference of +5.1 points, CI [−0.1, +10.2]; at 120b, Risa-H (steered + Hybrid) reaches 55.3% against 53.5% for uncontrolled + Uniform (+1.8 points, CI [−2.6, +6.2]). Within the uncontrolled pools, which cost about 1/16 of the candidate tokens per step, routing arbitration still adds 2.6 and 1.3 points over Uniform and hybrid arbitration 5.6 and 1.8 points, so the terminal level is effective at a small fraction of the generation budget.

G Cross-Family Reproduction (Qwen3.6-35B-A3B)

The main grid uses `gpt-oss-20b` (24 layers, 32 experts, 4 active) and `gpt-oss-120b` (36 layers, 128 experts, 4 active). To test portability to a different routing shape, we rerun the pipeline on `Qwen3.6-35B-A3B` (40 layers, 256 experts, 8 active), which also has a hybrid attention stack. The role-gating logic, fixed write coefficients, decision-token fraction, and K=4 selector remain unchanged. Because the raw fingerprint dimension changes, role centroids are re-estimated from task-disjoint Qwen action labels under the same role definition; the refit uses role labels exclusively. Sampling follows the vendor's recommended setting (temperature 0.6, top-p 0.95, top-k 20) rather than the `gpt-oss` setting; the step budget, candidate count, and attempt count are unchanged.

G.1 Full-Benchmark Evaluation

The cross-family condition covers all 500 SWE-bench Verified tasks, with K=4 attempts and n=16 candidates per step. We report it separately from the six `gpt-oss` conditions because it uses a different model family and sampling temperature. Under the main-table requirement of at least two graded patches, 498 tasks are eligible. Across the 2,000 scheduled (task, attempt) cells, 28 produce empty patches, while the eight cells for `astropy-8707` and `astropy-8872` share the reproducible image-build incompatibility described in Appendix A.5. Every selector is computed on the same available patch pool for each eligible task.

On the full benchmark, Risa improves over Uniform by 3.5 points (95% CI [+1.9, +5.0], $p < 0.001$), while Text improves by 3.3 points ([+1.7, +4.8], $p < 0.001$). Risa matches Text, with 10 routing-only wins and 9 text-only wins (exact McNemar p=1.000). Risa-H attains the highest deployable resolved rate at 45.6%. The Uniform-to-Oracle headroom is 8.9 points, of which Risa recovers approximately 39%.

Among the 476 tasks with all four attempts graded, resolved rates are 43.0% for Uniform, 46.4% for Text, 46.6% for Risa, 47.1% for Risa-H, and 52.1% for Oracle. Risa exceeds Uniform by 3.6 points (95% CI [+2.0, +5.3]), and the Risa–Text McNemar count remains 10:9 (p=1.000).

Selector	Resolved (%)	Gain vs. Uniform
Uniform	41.7	—
Text	45.0	+3.3 [+1.7, +4.8]
RISA	<u>45.2</u>	+3.5 [+1.9, +5.0]
RISA-H	**45.6**	**+3.9**
Oracle	50.6	+8.9

Table 6 Cross-family reproduction: final-patch selection (% resolved) over K=4 separately sampled `Qwen3.6-35B-A3B` attempts on the full 500-task benchmark (498 eligible tasks). Eligibility requires at least two graded patches. Brackets are task-bootstrap 95% confidence intervals for paired gains over Uniform. Bold and underlining mark the best and second-best deployable estimates.

G.2 Generation Outcome

The condition records 1,964 completed (task, attempt) cells out of 2,000 scheduled, with a 73.5% patch yield. Stored diffs preserve the accumulated patch at budget termination. Grading these final patches yields a 42.1% attempt-level resolved rate, alongside 0.1% under stock exit-triggered scoring. Table 6 reports task-level selector outcomes from the same re-graded patch pool.

H Hyperparameters

parameter	value
attempts per task K	4
candidates per step n	16
sampling temperature	1.0, top-p 1.0 (`gpt-oss`); 0.6, top-p 0.95, top-k 20 (`Qwen3.6`)
decision-token fraction	lowest 25% of patch tokens by teacher-forced probability
step fingerprint span	tool-call segment of the candidate (fallback: trailing 64 generated tokens)
`gpt-oss` reasoning efforts	native low / medium / high setting
max new tokens per query	4096 (low/medium), 65536 (high)
step budget per attempt	50 agent steps
history pool (exploration rule)	last 64 executed actions; the $\min(3, \lvert\mathcal{H}_t\rvert)$ most similar are scored
write-rule cluster threshold	$\tau = 0.65$ (pairwise WJ)
write-rule score	$z(H) + 1.5\,z(M) + z(\bar{S}) - z(P)$; coefficients fixed at $(1, 1.5, 1, -1)$
`Qwen3.6` generation budget	8192 new tokens per query, same 50-step budget
top-k logprobs requested	20 per generated token
role-centroid fit	task-disjoint 70/30 split, nearest-centroid (cosine)
terminal fingerprint prompt	problem statement + accumulated diff, `max_tokens`=1, temperature 0
bootstrap resamples	10^4 (task-clustered for pooled cells)
agent container limits	4 CPUs, 2 GB RAM

Table 7 Final configuration. All arms of an ablation share every value except the selection rule under study.

H.1 Selection of Final Settings

Three groups of knobs exist, and each was set by a stated criterion rather than by tuning on evaluation outcomes. (i) *Budget parameters* (K=4 attempts, n=16 candidates, 50 steps) were fixed in advance according to the compute available for a full 500-task grid at three effort settings. (ii) *Routing-readout parameters* were selected on diagnostic data before the campaign, then swept afterwards as robustness checks: decision-token fraction over $\{5, 10, 25, 50, 100\}$%, write-rule threshold $\tau \in \{0.55, 0.65, 0.75, 0.85\}$, exploration neighborhood $m \in \{1, 3, 5, 10\}$ crossed with history pool $\in \{8, 64\}$, and fingerprint window over {whole span, opening 64, trailing 64, decision tokens}. The criterion was the diagnostic pairwise AUC on outcome-labeled steps and

attempt pairs, which is disjoint from the evaluation grid; Table 8 shows every swept value. (iii) *Write-score coefficients* $(1, 1.5, 1, -1)$ were fixed once on the separate 98-step mixed-outcome pilot (Section A.3.5) and remain fixed throughout evaluation. All final parameters follow these diagnostic and pilot criteria, independently of the resolved rates reported in the main table.

I Compute Infrastructure and Cost

All experiments ran on a single server, shared with other users, with 8× NVIDIA H20 (95.6 GiB each, compute capability 9.0), two Intel Xeon Platinum 8480+ CPUs (56 cores / 112 threads each, 224 logical cores) and 2.0 TB of system RAM, under Ubuntu 22.04.5 LTS (Linux 5.15.0) with NVIDIA driver 560.35.05 (CUDA 12.6) and Docker 27.5.1. The serving stack is vLLM 0.15.1 with PyTorch 2.9.1+cu128, Transformers 4.57.3 and Python 3.12.9; the agent and all analysis scripts run on the same Python version with NumPy, pandas and the official `swebench` harness. The 20B model is served with tensor parallelism 1 (one replica per GPU, data-parallel round-robin); the 120B model with tensor parallelism 2 (three GPU pairs); the cross-family `Qwen3.6-35B-A3B` condition uses eight tensor-parallel-1 replicas. SWE-bench instance images are built and graded locally in Docker, with each evaluation container limited to 8 CPUs and 16 GB RAM.

I.1 Cost of the Method

The n=16 step candidates share a prefix and are issued as a batch, so prefix caching reduces repeated prefill work. On our serving stack the measured prefix-cache hit rate is 98.9%, and the realized wall-clock latency on the same hardware for an n=16 step is $\approx 2\times$ that of a single-sample step. A medium-effort steered rollout generates $\approx$ 90K tokens including all candidate continuations ($\approx$ 360K generated tokens per task at K=4). Per-step fingerprint scoring is arithmetic over returned tensors; terminal selection adds one short prefill and one discarded token per available patch, with no additional patch generation. Representative measured generation costs (wall-clock × devices) are 22 GPU-h (`20b` low), 222 GPU-h (`20b` high), 25 GPU-h (`120b` low), and $\approx$ 208 GPU-h for a 200-task `120b`-high subset served as three 2-GPU replicas. These values provide representative reference points; high-effort cost is dominated by long-context re-prefill rather than decode.

I.2 Randomness and Seeds

Two sources of randomness exist. Trajectory sampling uses the inference server's stochastic sampler: each of the K attempts is an independent draw at the configured temperature, which is the intended behavior, since the method operates on independently sampled attempts. Consequently single-attempt numbers vary between runs, and every reported comparison is either paired on the same stored pool (all selector rows) or aggregated over K attempts and hundreds of tasks. All post-hoc randomness is seeded and reproducible from the released code: bootstrap resampling uses `numpy.random.default_rng(0)`, trajectory subsampling for step statistics uses `random.Random(0)`, the diagnostic-pool construction and the 70/30 centroid split use `seed`=42; the corresponding diagnostic-pool and split id lists are released with the code, so the exact task sets can be reproduced without rerunning the sampler.

I.3 Evaluation Metrics

The primary metric is the official SWE-bench Verified *resolved rate*: a patch counts as resolved when the instance's fail-to-pass and pass-to-pass test sets both pass inside the instance container, as decided by the benchmark harness. This community-standard criterion grounds evaluation in execution rather than surface similarity to a reference patch. A selector's score is the resolved rate of the single patch it submits per task, so all selectors are compared as realized 1-of-K decisions rather than as rankings. Three reference quantities frame those numbers: *Uniform*, the exact expectation of choosing uniformly among a task's available patches; *Oracle*, the union of successes in the pool, an upper bound on what any selector could reach; and *patch yield*, the fraction of attempts that end with a nonempty diff, which separates candidate availability from candidate choice. For diagnostic analyses on labeled steps and attempt pairs we use pairwise AUC — the probability that a resolving branch is ranked above a non-resolving one — because those comparisons are about ordering

decision-token fraction	5%	10%	25%	50%	100%
offline AUC (cross-attempt)	.644	.648	**.657**	.657	.639
realized macro-avg (%)	—	48.0	**48.2**	48.2	—

per-condition realized spread across fractions: at most 2.4 *points; the selected 25% setting attains the top macro-average.*

write-rule threshold $\tau \in \{.55, .65, .75, .85\}$: pairwise AUC .548/.572/.536/.580
step-level fraction {10, 25, 50}%: peer-similarity AUC .554/.545/.544
exploration $m \in \{1, 3, 5, 10\}$, pool 8: .546/.545/.548/.551; pool 64: .533/.537/.547/.551

Table 8 Sensitivity of the selected routing-window and neighborhood hyperparameters. Candidate count, attempt count, step budget, and the four write-rule coefficients are held fixed throughout.

candidates, not about absolute calibration. Macro-averages weight each model–effort condition equally so that the six conditions, which differ in difficulty, contribute equally.

J Additional Pilot Measurements

J.1 Window-Control Battery

Using each patch's mean WJ agreement with the other available attempts as its score, AUC for ranking a resolved patch above an unresolved one in mixed-outcome groups is, by readout window: decision tokens (lowest-probability 25%) .657; whole span .639; random 25% (3 seeds) .634; uniformly spaced 25% .620; router-weight entropy in place of token probability .620; highest-probability 25% (complement) .616. Decision tokens lead the whole-span readout by .018 AUC and the probability-complement control by .041.

J.2 Routing Saturation

The fraction of layer–expert cells receiving nonzero routing mass rises from 0.75 (spans under 64 tokens) to 0.96 (beyond a thousand). Identity AUC, which uses WJ to distinguish pairs from the same task-and-prefix generation context from pairs drawn across contexts, falls to .58 for the trailing-64 window on long spans, while whole-span, decision-token, and opening-64 readouts retain .88, .93, and .98, respectively.

J.3 Choice Concentration and Cross-Attempt Agreement

The lowest-probability quartile of final-step generations carries a median 72% of total token surprisal $\sum_t -\log p_t$ (n=658 generations). Mean cross-attempt routing agreement is 0.56 at the remaining higher-probability positions and 0.65 at decision tokens; the main-paper decile plot reports the full distribution.

J.4 Threshold Sensitivity

The selected routing-window and neighborhood hyperparameters occupy a broad stable region (Table 8). Fractions from 10–50% are stable both offline (cross-attempt ranking AUC) and *realized* (resolved rate when the terminal selector uses each fraction, macro-averaged over the six model–effort conditions); the selected 25% fraction attains the top offline AUC and realized macro-average. Across the threshold sweep, pairwise AUC remains in the compact .536–.580 range.

J.5 Novelty Across Steps and Trajectories

We bin candidates by the similarity of their routing fingerprint to the trajectory's own recent executed actions, using the deployed score (mean of the three most similar entries in a 64-action history) and, as a robustness check, the mean over the whole history.

Within a step, routing-history quintiles yield pairwise AUCs near chance, identifying history disagreement as a computation-diversity prior. Over 2,636 branch-labeled write candidates from mixed-outcome steps,

the resolving fraction (most different first) is 52.9, 47.6, 44.0, 45.7, 55.5 percent under the deployed score (pairwise AUC 0.49) and 51.6, 48.6, 50.9, 48.4, 46.4 under the whole-history mean (AUC 0.52). Figure 5 (left) visualizes this within-step outcome balance.

Across trajectories, the relationship is strong and stable under both scores. Binning 1,021 rollouts by mean history similarity gives resolve rates of 33.8, 31.9, 26.5, 21.6, 5.4 percent (AUC 0.656; 30.4 to 11.7 percent, AUC 0.611, under the whole-history mean). Thus routing-history similarity is a strong progress marker (Figure 5, right). The controller operationalizes this stable trajectory-level signal as a diversity prior; targeted intervention studies can further characterize how actively changing routing history affects downstream outcomes.

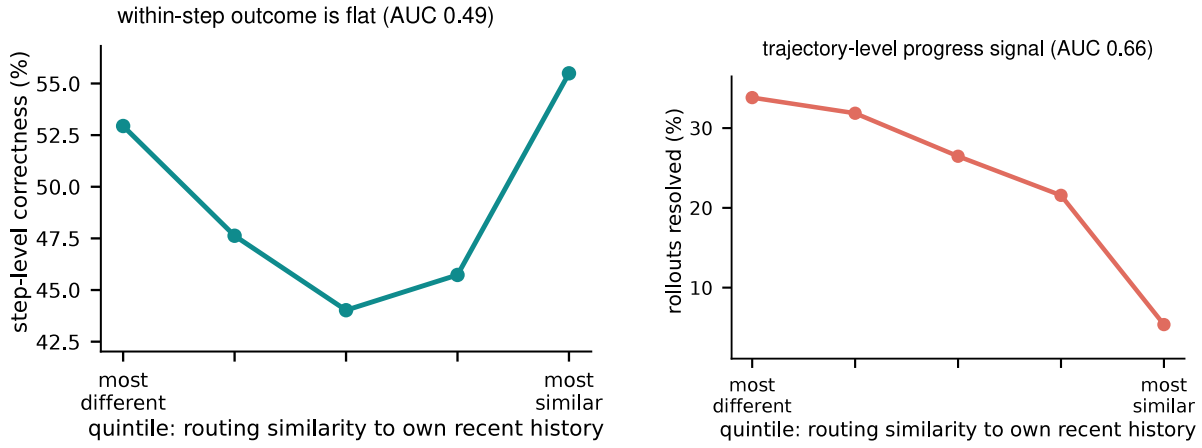

Figure 5 Similarity to one's own recent routing history. Left: within-step outcome balance supports the diversity-prior interpretation (n=2,636). Right: across rollouts, mean history similarity strongly separates resolved from unresolved attempts (n=1,021), motivating a trajectory-level diversity prior.

J.6 Steering Coverage Analysis

On the 80-task empirically hard diagnostic pool, we compare the full controller with the same agent scaffold and controller disabled. A submittable patch is an attempt ending with a nonempty final diff. Routing-guided steering raises this attempt-level yield from 79% to 94%, reducing patchless attempts from 21% to 6%. This coverage diagnostic measures candidate availability: the controller turns more otherwise stalled runs into candidates that terminal selection can compare. Task-level selector accuracy is reported separately on fixed steered pools in the main paper.

K Reproducibility Statement

Code for the agent scaffold, serving-side routing capture, selection rules, grading pipeline, and analysis scripts will be released under a research-friendly license upon publication. SWE-bench Verified is public, and the released pipeline regenerates per-candidate routing traces (expert indices and gate weights). The release will include regeneration scripts, derived pilot tables, and trace artifacts distributed under their source licenses. Analysis scripts fix their random seeds; trajectories use independent stochastic draws at the model-specific temperatures in Table 7. Reported comparisons aggregate over K attempts and use paired arms and instances where applicable.