

Trace-GRAF: Trace-Guided Granularity Routing for Sequential Agentic Workflows

Anastasiia Linich^{1,2} Artyom Sosedka^{1,3} Semen Budennyi^{1,4}

¹Sber AI, Moscow, Russia

²Moscow Institute of Physics and Technology (National Research University),
Dolgoprudny, Moscow Region, Russian Federation

³National University of Science and Technology MISIS, Moscow, Russia

⁴Artificial Intelligence Research Institute, Moscow, Russia

Correspondence: asya.more.collab@gmail.com

Abstract

Multi-agent LLM results can conflate decomposition with roles, communication, parallelism, and extra compute. We study strictly ordered tool-use workflows of dependent state-changing steps, where segmentation may also support auditing and handoff constraints. We introduce Trace-GRAF, a granularity selector that aligns one single-agent trace to target workflow steps, converts order violations and unrelated drift into split pressure, and selects the smallest granularity on the minimum-pressure plateau. It executes balanced contiguous segments because boundary ablation finds no significant success difference from semantic segmentation; pressure-minimizing partitions characterize cuts. Across held-out cohorts spanning two benchmarks and three executors, fixed decomposition sometimes improves and sometimes reduces success, while greater granularity both rescues failures and regresses solved runs. When SA success is observable, the pressure route preserves successful runs by construction. Across validation resplits on TerminalWorld, the calibrated extension’s mean recovery is 2.3–3.0% for GPT-OSS, 0.0–0.1% for Gemini, and 2.3–4.9% for DeepSeek. Simple fixed fallbacks match much of the observed gain. Trace-GRAF therefore provides a risk-aware routing diagnostic with benchmark- and split-dependent gains.

1 Introduction

LLM agents increasingly execute state-changing workflows through external tools (Wang et al., 2023; Yao et al., 2023; Schick et al., 2023). Multi-agent systems can help, but they often change decomposition, roles, communication, memory, parallelism, and compute simultaneously (Li et al., 2023; Wu et al., 2023; Hong et al., 2024). This makes the effect of decomposition itself difficult to isolate.

We isolate a narrower question: when should a strictly ordered workflow be split across sequen-

tial execution contexts? Terminal and enterprise operations often contain state changes where later actions depend on earlier state (Chu et al., 2026; Trivedi et al., 2024). Here parallelism is not the explanation; the engineering question is whether shorter local contexts improve reliability enough to offset handoff risk and cost.

The answer need not be monotonic. More contexts may rescue skipped, reordered, or drifting executions, but handoffs can remove useful context, add cost, and regress solved tasks. Sequential decomposition is therefore a routing problem over whether and how finely to segment another execution.

This paper proposes Trace-GRAF: *Trace-Guided Granularity Routing*. Code is available online.¹ The key observation is that a single-agent trace can contain deployment-time diagnostic signal. A failed run may reveal where the executor skipped, reordered, repeated, or drifted away from target workflow steps. Trace-GRAF aligns single-agent action blocks to target workflow steps, computes the minimum achievable within-segment pressure for each tested granularity, and uses those costs to select the number of balanced contiguous execution contexts. The semantic-versus-balanced ablation informs the choice of the simpler balanced execution protocol without establishing equivalence, while the pressure-minimizing partitions characterize where the trace evidence places cuts. We also evaluate validation-calibrated abstention as an optional launch-suppression extension.

We use the following research questions as structural anchors:

RQ1. When does sequential multi-agent decomposition improve final task success over a single tool-using agent?

¹github.com/khilling2/Trace-GRAF

RQ2. Is increasing the number of agents monotonic, or do rescues and regressions coexist?

RQ3. Does boundary construction materially change success, and do pressure-minimizing boundaries resemble semantic or balanced partitions?

RQ4. Can trace pressure support post-SA routing beyond simple fixed controls, and what does validation calibration add?

Our contributions are: (i) a controlled formulation of sequential decomposition granularity for strictly ordered agentic workflows; (ii) Trace-GRAF, a trace-guided selector that converts a failed single-agent execution trace into a granularity for balanced contiguous execution; (iii) an exact dynamic-programming pressure objective and an analysis of its induced boundaries; (iv) an evaluation of an optional validation-calibrated abstention extension, including its split sensitivity; and (v) a two-benchmark, three-executor evaluation with cross-judge sensitivity and cost controls.

2 Background and Related Work

Tool-use and multi-agent workflows. Tool-using agents couple language models with external actions (Yao et al., 2023; Schick et al., 2023); multi-agent frameworks add roles and communication (Li et al., 2023; Wu et al., 2023; Hong et al., 2024); and workflow optimizers search over agent structure (Zhuge et al., 2024; Zhang et al., 2024). These systems motivate decomposition but combine it with other interventions. We hold tools, evaluator, task order, workflow cards, and environment protocol fixed. Granularity changes the number of sequential contexts, segment-local prompts, action caps, and handoffs.

Decomposition and adaptive inference. Step-wise and decomposed inference expose intermediate structure during reasoning (Wei et al., 2022; Zhou et al., 2023; Khot et al., 2023; Prasad et al., 2024). Trace-GRAF instead selects granularity for an existing ordered executable workflow using observed execution behavior.

Process-grounded benchmarks. Terminal-World provides ordered shell workflows with deterministic validation (Chu et al., 2026); App-World provides stateful API tasks with unit-test evaluation (Trivedi et al., 2024). Together they

support the required ordered-process, trace, and final-state analyses in two tool environments.

3 Problem Formulation

Let a task x have an ordered workflow $w_1, \dots, w_n$. A granularity k corresponds to a contiguous partition

$$P_k = \{[a_1, b_1], \dots, [a_k, b_k]\},$$

where $a_1 = 1$, $b_k = n$, and $a_{r+1} = b_r + 1$. The $k = 1$ case is single-agent execution. For $k > 1$, each sequential agent receives the original task, its assigned workflow segment, and the shared environment state plus a mechanically rendered hand-off transcript from earlier segments.

For task x and candidate granularity k , let

$$Y_x(k) \in \{0, 1\}$$

denote final task success, and let $C_x(k)$ denote execution cost. The evaluated Trace-GRAF route uses

$$\pi(x) = (k_x, B_{x,k_x}),$$

where $B_{x,k}$ is the deterministic balanced-contiguous partition. We evaluate this post-SA granularity choice under regression risk and execution cost. The pressure route keeps $k = 1$ after observed SA success. After observed SA failure, it chooses among $k \in \{2, 4, 6, 8\}$ using trace-derived pressure. An optional validation-calibrated extension can suppress a launch when its candidate-reliability evidence is weak. For analysis, we compare against single-agent execution, fixed k policies, the best fixed fallback selected on validation, and a per-task upper-bound oracle over the tested configurations.

Because Y_x is binary, rerunning a solved task cannot improve its score; retaining $k = 1$ therefore guarantees non-regression while avoiding another launch. This guard assumes that task success is observable after SA execution, as it is through the benchmark evaluator; deployment requires an equivalent outcome verifier.

The selector optimizes trace pressure, while $C_x(k)$ measures the resulting policy’s execution cost.

For an adjacent increase $i \rightarrow j$, a *rescue* is $Y_x(i) = 0, Y_x(j) = 1$, while a *regression* is $Y_x(i) = 1, Y_x(j) = 0$. These policy-level outcomes quantify the net benefit and risk of moving to finer sequential decomposition; they do not isolate handoffs from the other protocol changes induced by k .

The setting assumes an ordered target process, such as a runbook, standard operating procedure, approved plan, or validated historical trace. Benchmark reference scripts serve as evaluation proxies for leakage-controlled workflow cards. Open-ended tasks without a stable intended procedure are outside scope.

4 Method

4.1 Overview

Trace-GRAF has three stages. First, it aligns the single-agent trace to the ordered workflow. Second, it turns the alignment into a split-pressure matrix and computes the minimum within-segment pressure $J(k)$ for each candidate k . Third, after SA failure, it selects the smallest tested granularity attaining the minimum pressure and executes that granularity as balanced contiguous segments. The boundary ablation in Section 6.3 motivates the simpler balanced protocol but does not establish equivalence to semantic segmentation; the dynamic-programming partitions characterize cut structure. Trace-GRAF uses trace pressure without validation outcomes. The optional calibrated-abstention extension adds target-domain validation evidence. The structured alignment judge is the only routing-overhead LLM call; pressure construction, selection, and calibration are deterministic.

4.2 Single-Agent Trace Alignment

Given a single-agent run, split the trace into action blocks $a_1, \dots, a_m$. In TerminalWorld, an action block is centered on one shell call plus the surrounding assistant message and tool observation. A structured LLM judge aligns each action block to each target workflow step and returns

$$A \in \{\text{full}, \text{partial}, \text{unrelated}\}^{m \times n}.$$

`full` means the action directly attempts the workflow step, whether or not the attempt succeeds. `partial` means related preparation, verification, repair, or partial overlap. `unrelated` means no meaningful correspondence. We convert the categorical matrix to

$$M_{ai} = \begin{cases} 1, & A_{ai} = \text{full} \\ 0, & A_{ai} = \text{partial} \\ -1, & A_{ai} = \text{unrelated}. \end{cases}$$

Partial following is deliberately neutral in the main pressure formula: it is evidence of local relation, but not by itself a reason to split.

4.3 Split-Pressure Matrix

Trace-GRAF uses two principles. First, workflow steps should not be merged when the single-agent trace violates their relative order. Second, steps should not be merged when unrelated actions occur between their full-following attempts. Let $F = \mathbb{K}[M = 1]$ be the full-following matrix, and let $T_{ab} = \mathbb{K}[a < b]$ be the strict order matrix over action blocks. The observed precedence matrix is

$$R = F^\top T F,$$

where R_{ij} counts how often a full-following action for step i occurs before a full-following action for step j .

Define an unrelated-action indicator

$$e_a = \mathbb{K} \left[\max_i M_{ai} < 0 \right].$$

Let $p_i = \min\{a : F_{ai} = 1\}$ be the first full-following position for step i , when it exists. For $i < j$, define trace drift

$$E_{ij} = \sum_{a=\min(p_i, p_j)+1}^{\max(p_i, p_j)-1} e_a$$

when both positions exist, and $E_{ij} = 0$ otherwise. The split-pressure matrix is

$$S_{ij} = R_{ji} + E_{ij}, \quad i < j,$$

with $S_{ji} = S_{ij}$ and $S_{ii} = 0$. The term R_{ji} penalizes observed order violations; E_{ij} penalizes unrelated drift between workflow-aligned actions.

4.4 Pressure-Minimizing Partitions

For a fixed number of execution contexts k , the pressure analysis computes

$$P_k^* = \arg \min_{P: |P|=k} \sum_{[a,b] \in P} \sum_{a \leq i < j \leq b} S_{ij}.$$

Equivalently,

$$P_k^* = \arg \min_{P: |P|=k} \frac{1}{2} \langle S, X(P) \rangle,$$

where $X(P)_{ij} = 1$ if steps i and j are assigned to the same segment. Because partitions are contiguous, this objective is solved exactly by dynamic programming over prefix endpoints; Appendix D gives the recurrence and tie rule. Let

$J(k) = J(P_k^*)$ be the minimum within-segment pressure and

$$\rho(k) = \frac{J(1) - J(k)}{J(1) + \epsilon}$$

be pressure reduction, with $\rho(k) = 0$ for all k when $J(1) = 0$. This convention avoids creating artificial pressure gains when the single-agent trace has no within-workflow split pressure.

For the primary post-SA pressure route, let $\mathcal{K}_{\text{MAS}} = \{2, 4, 6, 8\}$. If SA succeeds, an absorbing guard retains $k = 1$. Otherwise, Trace-GRAF selects

$$k_P = \min \left(\underset{k \in \mathcal{K}_{\text{MAS}}}{\text{argmin}} J(k) \right)$$

and executes the corresponding balanced-contiguous partition B_{k_P} . Because nonnegative within-segment pressure is non-increasing as more cuts are allowed, this rule selects the smallest tested k on the minimum-pressure plateau; it has no learned or validation-calibrated complexity penalty. The optimizer P_k^* supplies $J(k)$ and the boundary partition used in the structural analysis. The primary routing, cross-judge recovery, significance, and cost results use the pressure-selected k_P with balanced contiguous execution.

4.5 Optional Extension:

Validation-Calibrated Abstention

The optional abstaining selector tests whether target-domain validation can suppress weak launches. Calibration uses validation tasks whose SA run failed. For each such task t and candidate $k \in \{1, 2, 4, 6, 8\}$, it computes $P_{t,k}^*$, $J_t(k)$, and $\rho_t(k)$, and evaluates the balanced-contiguous outcome $Y_t(k)$. Here $k = 1$ retains the observed SA outcome without another launch, whereas $k > 1$ denotes a balanced-contiguous rerun. Candidate reliability is estimated from a Bayesian posterior over success on this SA-failed validation slice. If candidate k solves s_k of n_k available tasks in that slice, we use

$$p_k \mid \mathcal{V} \sim \text{Beta}(s_k + 1, n_k - s_k + 1)$$

and define $\pi_k = Q_{0.25}(p_k)$, the conservative posterior quartile. This avoids overconfidence from the small validation split.

The selector also includes abstention, denoted $\perp$. Abstention is correct on validation when no

executable k solves the task:

$$Y_t(\perp) = 1 - \max_{k \in \{1, 2, 4, 6, 8\}} Y_t(k).$$

Here $Y_t(\perp)$ labels a correct decision to withhold an automated launch, not benchmark task success; deployment routes this outcome to human review. For every candidate $c \in \{1, 2, 4, 6, 8, \perp\}$, including abstention, we estimate a posterior and conservative reliability $\pi_c = Q_{0.25}(p_c)$. For executable candidates, the task-specific routing signal is $r_t(k) = \rho_t(k)$. For abstention, it is an unsolvability score

$$r_t(\perp) = \frac{1}{2}(u_t^{\text{miss}} + u_t^{\text{extra}}),$$

where u_t^{miss} is the fraction of workflow steps with no full-following action and u_t^{extra} is the fraction of action blocks unrelated to every workflow step. Both are computed from the same numeric alignment matrix. The calibrated selector scores

$$\text{Score}_t(c) = \pi_c(1 + r_t(c)).$$

If the single-agent run solves the task, the absorbing guard selects $k = 1$. Otherwise, the extension chooses the highest-scoring k candidate, breaking ties toward smaller k , unless abstention has strictly higher score. Appendix B reports validation-split sensitivity; Appendix C gives implementation details.

5 Experimental Setup

Benchmark and task selection. The 8–18-step cohorts contain 107 TerminalWorld tasks and 105 AppWorld tasks. Fixed validation sets contain 30 and 28 tasks, respectively, leaving 77 held-out test tasks per benchmark and executor. Workflow cards are generated from reference scripts or train/dev solution-code proxies after removing executable details. Appendix C describes the leakage controls.

Execution conditions. The matrix uses openai/gpt-oss-20b, google/gemini-2.5-flash, and deepseek/deepseek-v4-flash on both benchmarks. Within each benchmark, SA and MAS share the tool interface, evaluator, environment-reset policy, and workflow cards. MAS adds sequential contexts, segment-local prompts and action caps, and mechanical handoffs.

Each SA trace is aligned independently by all three models. The native-judge setting uses the executor as judge; the cross-judge sweep propagates judge substitution through routing. Validation-only controls select a fixed fallback granularity without trace pressure.

Run configuration. Every cell in the fixed balanced task/model/granularity matrix contains one execution at temperature 0. Action budgets, timeouts, handoff limits, exact prompts, and provider details appear in Appendix C.

Policies and metrics. We compare the single-agent baseline, balanced fixed policies with $k \in \{2, 4, 6, 8\}$, and a per-task upper-bound oracle over those fixed configurations. For post-SA routing, Trace-GRAF uses minimum partition costs to select k_P , then executes that granularity as balanced contiguous segments. We compare it with always launching $k = 2$ after SA failure and with SA+BESTFALLBACK, which selects the highest-success $k \in \{2, 4, 6, 8\}$ on validation, breaking ties toward smaller k . Validation resampling evaluates the optional calibrated-abstention extension. The main outcome is attempted-run success: `completed_success` divided by all attempted runs. Adjacent- k regressions and rescues measure the net outcome change from finer decomposition. We also report approximate policy token cost. Boundary analyses compare semantic and balanced execution and characterize the pressure-minimizing cuts relative to both references.

Reported percentages are point estimates. For the principal routing outcome we use two-sided 90% Wilson intervals. Paired comparisons against SA+BESTFALLBACK use a one-sided exact McNemar test for improvement. We apply Holm correction across the nine executor-judge comparisons and report the smallest unadjusted and adjusted values.

6 Results

6.1 RQ1: Sequential Decomposition Can Help, but Depends on the Executor

Table 1 reports the complete held-out fixed-policy matrix. On TerminalWorld, GPT-OSS rises from 9.1% under SA to 31.2% at $k = 8$, and Gemini rises from 27.3% to 32.5% at $k = 2$; every fixed MAS configuration is below DeepSeek’s 49.4% SA result. AppWorld is more conservative: all fixed MAS configurations have lower observed

success than SA for GPT-OSS and Gemini, while DeepSeek changes from 35.1% to 36.4% at $k = 2$. Per-task oracles remain above every fixed policy in all six benchmark/executor cells. Thus, the answer to RQ1 is conditional: sequential decomposition can improve observed success for some executors, but the direction and useful granularity are benchmark- and model-dependent.

6.2 RQ2: Rescues and Regressions Coexist Across Granularities

The final two columns of Table 1 pool adjacent increases $1 \rightarrow 2$, $2 \rightarrow 4$, $4 \rightarrow 6$, and $6 \rightarrow 8$ within each held-out benchmark/executor cell. TerminalWorld rescue rates are 18.6–20.6%, while regression rates are 33.1–42.7%. AppWorld rescue is lower at 1.4–7.9%; regression ranges from 6.5% for DeepSeek to 70.0% for Gemini. Thus, the answer to RQ2 is that rescues and regressions coexist in every cell; increasing k is not a monotonic refinement.

6.3 RQ3: No Significant Difference Is Detected Across Boundary Schemes

The boundary ablation compares semantic and balanced contiguous segmentation in 24 benchmark/executor/ k settings using two-sided exact McNemar tests. No comparison reaches $p < .05$ (smallest $p = .063$); paired coverage is incomplete in some semantic cells, so the result does not establish equivalence. Trace-GRAF adopts balanced contiguous execution as the simpler protocol. Appendix Table 4 characterizes the pressure-minimizing cuts relative to semantic and balanced references. Across 1,848 task partitions, the DP cuts are closer to semantic in 59.5%, closer to balanced in 17.9%, and tied in 22.6%. The pressure objective selects granularity and supplies boundary structure for analysis, while execution uses the balanced protocol for which the ablation detected no significant success difference.

6.4 RQ4: Pressure Routing Is Benchmark- and Judge-Dependent

Table 2 reports realized post-SA recovery for both benchmarks. In the native-judge setting, Trace-GRAF recovers 28.6/19.6/17.9% of SA failures for GPT-OSS/Gemini/DeepSeek on TerminalWorld, versus 3.6/0.0/6.0% on AppWorld. The corresponding 90% Wilson intervals for overall Trace-GRAF success are 26.8–44.4/32.8–50.9/49.1–67.2% on TerminalWorld and 23.3–

Benchmark	Executor	SA	$k=2$	$k=4$	$k=6$	$k=8$	Oracle	Rescue	Regress.
TerminalWorld	GPT-OSS	9.1	27.3	29.9	27.3	<u>31.2</u>	49.4	18.6	37.5
TerminalWorld	Gemini	27.3	<u>32.5</u>	24.7	31.2	31.2	50.6	18.7	42.7
TerminalWorld	DeepSeek	<u>49.4</u>	40.3	44.2	39.0	39.0	68.8	20.6	33.1
AppWorld	GPT-OSS	<u>28.6</u>	19.5	19.5	20.8	15.6	36.4	7.9	42.6
AppWorld	Gemini	<u>16.9</u>	5.2	1.3	2.6	3.9	19.5	1.4	70.0
AppWorld	DeepSeek	35.1	<u>36.4</u>	35.1	32.5	31.2	40.3	2.0	6.5

Table 1: Held-out fixed-policy success and adjacent-granularity transitions (%). Each success cell contains $N = 77$ tasks. Oracle is the per-task upper bound over $k \in \{1, 2, 4, 6, 8\}$; the best value is bold and the second-best value is underlined. Rescue and regression pool 308 adjacent pairs per row and condition on a failed and solved lower- k run, respectively.

Benchmark	Executor	SA	SA $\rightarrow k=2$	BestFB	GRAF	Recover
TerminalWorld	GPT-OSS	9.1	29.9	<u>33.8</u>	35.1	28.6
TerminalWorld	Gemini	27.3	41.6	<u>37.7</u>	41.6	19.6
TerminalWorld	DeepSeek	49.4	58.4	58.4	58.4	17.9
AppWorld	GPT-OSS	28.6	32.5	32.5	<u>31.2</u>	3.6
AppWorld	Gemini	16.9	16.9	16.9	16.9	0.0
AppWorld	DeepSeek	35.1	<u>37.7</u>	<u>37.7</u>	39.0	6.0

Table 2: Held-out routing on 77 test tasks per benchmark/executor cell (%). SA $\rightarrow k=2$ always runs two agents after SA failure; BestFB is the validation-selected SA+BESTFALLBACK. GRAF uses the native judge to select k from pressure and executes balanced contiguous segments at that k . Among the four comparable policy columns, the best value is bold and the second-best distinct value is underlined. Recover is GRAF success among SA-failed tasks. Successful SA runs retain $k = 1$, so non-regression is 100% by construction.

Bench.	Executor	N	Exact	No-full	Recovery
TW	GPT-OSS	107	58.2	86.2	25.7–28.6
TW	Gemini	107	58.0	73.0	19.6–23.2
TW	DeepSeek	107	46.7	82.1	15.4–20.5
AW	GPT-OSS	105	37.2	82.7	1.8–5.5
AW	Gemini	105	41.2	72.1	0.0–1.6
AW	DeepSeek	105	58.6	79.0	0.0–6.0

Table 3: Cross-judge agreement and recovery (%). TW/AW denote TerminalWorld/AppWorld. N is the full cohort used for agreement; Recovery is evaluated on the held-out split. Exact is mean per-trace all-three exact-row agreement; No-full is the pooled share without full/non-full disagreement; Recovery ranges over judges.

40.4/11.0–25.0/30.3–48.3% on AppWorld. Relative to SA+BESTFALLBACK, the Trace-GRAF point estimate differs by at most 3.9 accuracy points and is lower in the AppWorld GPT-OSS cell. The fixed post-SA $k = 2$ control is also competitive: Trace-GRAF has a higher point estimate in two cells, the same estimate in three, and a lower estimate in one.

Table 3 tests the reliability of the alignment component itself. Mean per-trace all-three exact-row agreement ranges from 37.2–58.6%, whereas 72.1–86.2% of pooled action rows contain no cell-level disagreement on full versus non-full. Judge substitution changes recovery but preserves

the benchmark contrast: TerminalWorld ranges reach 15.4–28.6%, while AppWorld remains at 0.0–6.0%. Agreement is higher under the coarser full/non-full comparison used by split pressure, although the two columns use different aggregation schemes and judge choice remains a material source of routing variance.

Statistical comparison. For each executor, we pool its 77 TerminalWorld and 77 AppWorld test tasks and compare every alignment-judge Trace-GRAF route with SA+BESTFALLBACK. Among these nine paired comparisons, the smallest unadjusted one-sided exact McNemar value is $p = .073$ for the Gemini executor with GPT-OSS judge, with 9 of 12 discordant pairs favoring Trace-GRAF. This result does not survive Holm correction across the nine-test family ($p_{\text{adj}} = .657$); hence the sweep provides no multiplicity-corrected evidence that Trace-GRAF outperforms SA+BESTFALLBACK.

Appendix B reports calibration, action-count, and cost controls. Calibration is split-sensitive and mostly abstention-oriented; action count does not distinguish successful granularities in the reported comparisons. Trace-GRAF has lower estimated cost than SA+BESTFALLBACK in two of six benchmark/executor cells and higher esti-

mated cost in the other four (Table 5). RQ4 therefore finds benchmark- and judge-dependent recovery, limited gains beyond simple controls, and no multiplicity-corrected evidence of superiority.

7 Conclusion

Fixed sweeps show that decomposition both rescues and regresses. An absorbing guard preserves successful SA runs; after failure, TraceGRAF uses trace-derived pressure, without validation outcomes, to select k for balanced contiguous execution. Recovery is stronger on TerminalWorld, simple fallbacks match much of the observed gain, and corrected comparisons provide no evidence of superiority. The calibrated extension is abstention-oriented and split-sensitive. Trace pressure therefore provides a routing diagnostic with benchmark-dependent gains.

Limitations

The results are benchmark-dependent and apply to ordered workflows with contiguous boundaries rather than open-ended, branching, or parallel tasks. Judge choice can affect routes, and the calibration extension is sensitive to validation composition; $Q_{0.25}$ remains a conservative design choice. Incomplete paired coverage limits the semantic-versus-balanced comparison in some cells, and one execution represents each fixed balanced task/model/granularity cell. Token counts are approximate, and alignment overhead can approach the cost of a fallback rerun. Appendix B reports the corresponding analyses.

Ethical Considerations

The experiments use public benchmark tasks and generated execution traces rather than private user data. The primary ethical and operational concerns are benchmark license compliance, accidental disclosure through saved traces, and harm from executing unsafe or unintended actions. Automated terminal agents can modify files, run commands, and interact with services, so deployment should include sandboxing, least-privilege permissions, approval boundaries, audit logging, and rollback procedures. Any released traces should remove secrets, credentials, private paths, personal information, and benchmark artifacts whose license does not permit redistribution.

References

- Zhaoyang Chu, Jiarui Hu, Xingyu Jiang, Pengyu Zou, Han Li, Chao Peng, Peter O’Hearn, Earl T. Barr, Mark Harman, Federica Sarro, and He Ye. 2026. [TerminalWorld: Benchmarking agents on real-world terminal tasks](#). *Preprint*, arXiv:2605.22535.
- Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. [MetaGPT: Meta programming for a multi-agent collaborative framework](#). In *International Conference on Learning Representations*.
- Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. [Decomposed prompting: A modular approach for solving complex tasks](#). In *International Conference on Learning Representations*.
- Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. [CAMEL: Communicative agents for “mind” exploration of large language model society](#). *Preprint*, arXiv:2303.17760.
- Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2024. [ADaPT: As-needed decomposition and planning with language models](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4226–4252.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). In *Advances in Neural Information Processing Systems*.
- Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. [AppWorld: A controllable world of apps and people for benchmarking interactive coding agents](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076, Bangkok, Thailand. Association for Computational Linguistics.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2023. [A survey on large language model based autonomous agents](#). *Preprint*, arXiv:2308.11432.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. [AutoGen: Enabling next-gen LLM applications via multi-agent conversation](#). *Preprint*, arXiv:2308.08155.

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

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2024. [AFlow: Automating agentic workflow generation](#). *Preprint*, arXiv:2410.10762.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). In *International Conference on Learning Representations*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. [Language agents as optimizable graphs](#). *Preprint*, arXiv:2402.16823.

complete three-executor, two-benchmark analysis from the native alignment judge.

A Boundary-Location Ablation

Execution-boundary ablation. The routing evaluation uses balanced contiguous execution. Across the paired semantic-versus-balanced comparison, all 24 two-sided exact McNemar tests have $p \geq .063$ after rounding to three decimals. Paired coverage is complete for most Terminal-World cells but incomplete for some DeepSeek and AppWorld cells. No statistically significant success difference is detected between semantic and balanced boundaries, so Trace-GRAF uses the simpler balanced contiguous protocol.

Pressure-boundary analysis. For each held-out task, Trace-GRAF DP boundaries are compared with the semantic and balanced contiguous references by $\delta(b, c) = \sum_i |b_i - c_i|/k$. A lower value means that the DP partition is closer to that reference. The semantic reference is selected exhaustively from Claude Sonnet 4.5 split-quality annotations by maximizing the sum of split qualities minus 0.25 times the range of segment sizes and minus the number of single-card segments; the balanced reference uses contiguous segment sizes that differ by at most one. Table 4 reports the

Benchmark	Executor	k	Partitions	δ_{sem} mean $\pm$ SD	δ_{bal} mean $\pm$ SD	Sem. closer	Median sem./bal.
TerminalWorld	GPT-OSS	2	77	2.006 $\pm$ 1.087	2.084 $\pm$ 0.894	44.2	2.000/2.000
TerminalWorld	GPT-OSS	4	77	2.536 $\pm$ 1.273	2.633 $\pm$ 1.206	45.5	2.250/2.750
TerminalWorld	GPT-OSS	6	77	1.952 $\pm$ 1.150	2.457 $\pm$ 1.104	80.5	1.667/2.333
TerminalWorld	GPT-OSS	8	77	1.208 $\pm$ 1.292	1.744 $\pm$ 1.345	68.8	0.875/1.625
TerminalWorld	Gemini	2	77	2.195 $\pm$ 0.932	2.221 $\pm$ 0.767	40.3	2.000/2.000
TerminalWorld	Gemini	4	77	2.597 $\pm$ 1.250	2.734 $\pm$ 1.164	49.4	2.250/2.750
TerminalWorld	Gemini	6	77	2.035 $\pm$ 1.214	2.548 $\pm$ 1.082	80.5	1.667/2.333
TerminalWorld	Gemini	8	77	1.232 $\pm$ 1.320	1.787 $\pm$ 1.393	70.1	0.875/1.625
TerminalWorld	DeepSeek	2	77	1.773 $\pm$ 1.125	1.786 $\pm$ 0.933	36.4	2.000/1.500
TerminalWorld	DeepSeek	4	77	2.416 $\pm$ 1.221	2.487 $\pm$ 1.198	42.9	2.250/2.500
TerminalWorld	DeepSeek	6	77	1.939 $\pm$ 1.124	2.405 $\pm$ 1.095	79.2	1.667/2.333
TerminalWorld	DeepSeek	8	77	1.185 $\pm$ 1.278	1.692 $\pm$ 1.349	67.5	0.750/1.625
AppWorld	GPT-OSS	2	77	1.188 $\pm$ 0.911	1.091 $\pm$ 0.715	31.2	1.000/1.000
AppWorld	GPT-OSS	4	77	1.416 $\pm$ 0.811	1.477 $\pm$ 0.943	44.2	1.250/1.250
AppWorld	GPT-OSS	6	77	1.320 $\pm$ 0.838	1.758 $\pm$ 0.755	76.6	1.000/1.500
AppWorld	GPT-OSS	8	77	0.744 $\pm$ 0.883	1.317 $\pm$ 1.081	70.1	0.375/1.250
AppWorld	Gemini	2	77	1.935 $\pm$ 1.021	1.968 $\pm$ 0.745	41.6	2.000/2.000
AppWorld	Gemini	4	77	2.338 $\pm$ 0.940	2.510 $\pm$ 0.941	57.1	2.250/2.500
AppWorld	Gemini	6	77	1.738 $\pm$ 1.064	2.284 $\pm$ 0.798	84.4	1.500/2.333
AppWorld	Gemini	8	77	0.948 $\pm$ 1.131	1.544 $\pm$ 1.280	71.4	0.500/1.625
AppWorld	DeepSeek	2	77	1.929 $\pm$ 0.992	1.922 $\pm$ 0.735	37.7	2.000/2.000
AppWorld	DeepSeek	4	77	2.195 $\pm$ 0.903	2.347 $\pm$ 0.925	54.5	2.250/2.250
AppWorld	DeepSeek	6	77	1.677 $\pm$ 0.884	2.219 $\pm$ 0.701	83.1	1.500/2.333
AppWorld	DeepSeek	8	77	0.914 $\pm$ 1.036	1.519 $\pm$ 1.212	71.4	0.500/1.625

Table 4: Boundary-location analysis for both benchmarks and all executors. “Sem. closer” is the percentage of partitions with $\delta_{\text{sem}} < \delta_{\text{bal}}$. Every cell contains all 77 held-out tasks.

B Routing Cost and Additional Controls

Token cost. Table 5 estimates tokens as $\lceil 4 \text{ words}/3 \rceil$ from saved raw traces and native judge prompts/responses. It is a tokenizer-free comparison, not provider billing. For GRAF, the judge-derived pressure selects k , and the execution cost is that of balanced contiguous execution at the selected k . Trace-GRAF has a lower cost point estimate than SA+BESTFALLBACK in two of six benchmark/executor cells (both on TerminalWorld) and a higher estimate in the other four. The 90% bootstrap interval for the GPT-OSS TerminalWorld contrast crosses zero. Wall-clock coverage is insufficient for a uniform latency comparison.

Optional calibration-extension sensitivity. For the calibrated abstention extension, we repeat calibration and routing for 1,000 seeded random validation splits, holding the $Q_{0.25}$ rule fixed. With 30 validation tasks, mean TerminalWorld recovery under GPT-OSS/Gemini/DeepSeek judges is 2.3/3.0/2.7% for the GPT-OSS executor, 0.0/0.0/0.1% for Gemini, and 4.9/2.3/3.3% for DeepSeek; every AppWorld mean-recovery cell rounds to 0.0%. On TerminalWorld, reducing validation size from 40 to 10 lowers modal-route stability from 93.3–95.2% to 79.8–82.3% (GPT-OSS), 99.9% to 91.7–93.5% (Gemini), and 85.5–88.8% to 62.7–67.6% (DeepSeek). The calibrated extension’s result on a single held-out split is therefore sensitive to calibration composition. This sen-

sitivity does not apply to Table 2, which reports the validation-free Trace-GRAF route.

Lightweight trace baseline. We test the number of executable actions in the original SA trace as a simple alignment-free feature. Across success-by-granularity comparisons, no non-exploratory 90% paired-bootstrap interval for the median action-count difference excludes zero. Trace length alone therefore does not distinguish successful granularities in this analysis; learned multi-feature routing remains future work.

Benchmark	Executor	BestFB vs. SA	GRAF vs. SA	GRAF vs. BestFB	90% CI
TerminalWorld	GPT-OSS	+240.0	+217.1	-6.8	[-14.6, +1.6]
TerminalWorld	Gemini	+279.7	+132.2	-38.9	[-54.6, -20.9]
TerminalWorld	DeepSeek	+71.5	+91.1	+11.4	[+9.0, +14.1]
AppWorld	GPT-OSS	+74.6	+166.9	+52.9	[+41.1, +65.1]
AppWorld	Gemini	+111.2	+134.2	+10.9	[+6.5, +16.5]
AppWorld	DeepSeek	+134.3	+168.5	+14.6	[+11.5, +18.4]

Table 5: Approximate policy token-cost change (%). The final column gives the task-level percentile-bootstrap interval for Trace-GRAF versus BestFB. All aggregates use 77 held-out tasks per benchmark/executor cell.

C Reproducibility and Prompt Templates

Execution record. Each cell in the fixed balanced matrix contains one execution. Executor and alignment calls used temperature 0. The provider defaults supply `top_p` and the output-token cap. Single-agent TerminalWorld runs allowed 80 tool actions and single-agent AppWorld runs allowed 60; both had 600 seconds per task, with 30 seconds per TerminalWorld tool call and 60 seconds per AppWorld tool call. Tool observations were truncated to 6,000 characters. Multi-agent runs on both benchmarks shared an 80-action total budget, with per-segment caps of 40, 20, 13, and 10 for $k \in \{2, 4, 6, 8\}$, respectively. Mechanical handoffs retained at most 20,000 transcript characters, with individual assistant, stdout, and stderr excerpts capped at 2,000 characters. Alignment calls had a 180-second timeout, strict schema validation, and one schema-repair attempt.

Calls used an OpenRouter-compatible endpoint. The model identifiers are the three identifiers reported in Section 5. Provider routing determines the backend and revision, which remains a source of replication variance even with temperature zero.

Executor system prompt. The shared core prompt was:

You are executing a workflow in a sandboxed task environment.

You will receive: (1) the original task; (2) workflow cards, where each card contains a local State and Goal; and (3), optionally, execution history from earlier work on this task.

Use the available tools to inspect and modify the environment. The workflow cards are the task guidance. They may describe goals without giving exact commands. When you believe the provided workflow cards have been completed, respond with a brief final message.

Benchmark-specific tool instructions were appended to this core prompt. The user message

supplied the original task, prior mechanical transcript, and the cards assigned to the current segment. Non-final segments were asked to name the variables and critical values required by the next segment.

Workflow cards. Workflow-card generation converts ordered reference steps into execution-neutral `local_state` and `local_goal` fields. The anti-leakage prompt excludes commands, exact API calls, helper variables, and solution-code structure while retaining task-critical entities and required literal values. Execution agents receive only these state-goal cards. In deployment, the cards can be derived from reviewed procedures rather than benchmark solutions.

Trace-alignment system prompt. The alignment judge was told that it was aligning a single-agent execution trace to an ordered target workflow and must return a dense $M \times N$ categorical matrix. The three labels were defined as follows:

`full_following`: the action directly follows or attempts the step; a failed command can still receive this label because intent, not success, is being classified.

`partial_following`: the action partially follows, prepares, verifies, repairs, or retries the step, or is semantically close but incomplete.

`unrelated`: the action has no meaningful correspondence to the step or belongs to a different part of the workflow.

The remaining instructions were: do not require an exact command; do not judge whole-task success; do not reward successful execution; treat generic exploration as unrelated unless it materially supports a step; allow one action to align with multiple steps; prefer `unrelated` in the absence of clear evidence; and return only the prescribed JSON schema.

The alignment user template recorded the task and run identifiers, executor, run status, M , N , original task, ordered workflow

cards (`source_step`, `local_state`, and `local_goal`), and each action block’s command or code, exit code, timeout flag, wall time, stdout, and stderr. Saved prompts and raw responses therefore expose the exact instance presented to each judge.

Deployment protocol. An operator first supplies a reviewed ordered procedure and a deterministic success check. The system runs SA once; a successful run is retained. Under TraceGRAF, a failed trace is aligned to the procedure and the smallest tested granularity on the minimum-pressure plateau is selected. The ordered procedure is divided into balanced contiguous segments at that k , which execute sequentially in a fresh or rolled-back environment. The pressure-minimizing partition accompanies the route as a boundary-structure diagnostic. The calibrated-abstention extension combines pressure reduction with candidate reliabilities estimated on a fixed target-domain validation set and may abstain. Each segment receives its local cards, shared state, and a bounded mechanical transcript; abstention routes the task to human review. Both protocols use runbooks or approved plans rather than hidden gold solutions.

Artifact coverage. The accompanying code and data release contains 1,908 alignment matrices: 636 native-judge matrices and 1,272 cross-judge matrices. Each record is keyed to a unique run identifier and retains the rendered prompt, raw response, validated alignment, and derived pressure features. Validation sampling used seed 20260618. The calibration-split sensitivity analysis used base seed 20260802 with deterministic offsets by benchmark, executor, judge, and validation size. The release also contains execution summaries, validation/test assignments, balanced-contiguous outcomes, and reconstructed pressure, semantic, and balanced boundaries used in the reported analyses.

D Pressure Partition Details

The dynamic program computes segment costs

$$Q(a, b) = \sum_{a \leq i < j \leq b} S_{ij}$$

and then minimizes

$$D[r, b] = \min_{a \leq b} (D[r - 1, a - 1] + Q(a, b))$$

for r segments ending at workflow step b . Backpointers recover the contiguous partition used in the boundary-structure analysis. The implementation scans predecessors from left to right and updates only on strict improvement, so an exact tie retains the first predecessor encountered and favors earlier cuts; it applies no balance or short-segment penalty.