BEYOND SAMPLING MORE: SCALING LLM AGENTS THROUGH SKILL-CONDITIONED EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Repeated sampling is a simple and effective form of test-time scaling for LLM agents, but every candidate trajectory is drawn from the same procedural context. We study reusable *skills*—packaged procedural guidance—as a source of structured exploration, and ask how a fixed rollout budget should be divided between breadth across skills and depth within a single skill. From a provenance-tracked pool of 4,802 open-source skills, we build Skill-Diverse Best-of-N (SD-BoN), a label-free candidate generator that selects eight skills with a relevance–diversity rule and allocates sixteen rollouts to each. Across four models on AppWorld, WebArena, GAIA, and τ-bench, SD-BoN with flat majority voting improves over similarity Top-1 by 8.33 points for GPT-5.6-sol on AppWorld and by 6.96–7.51 points on the remaining benchmarks; over all 16 model–benchmark pairs, the mean gain is 7.26 points (range 5.22–9.57). A label-aware pilot-ranking diagnostic, which consumes 1,024 success-labeled screening rollouts per task and model, yields a smaller mean allocation gain of 1.85 points. Our analysis separates candidate coverage from selected success: broader skill exploration reliably enlarges the set of successful candidates, but it also widens the gap that a practical selector must close. Evaluating both quantities, rather than coverage alone, is essential for judging skill-conditioned test-time scaling.

1 INTRODUCTION

Language model agents complete tasks through sequences of decisions, tool calls, and interactions with external environments, and in benchmarks such as AppWorld and WebArena success depends on the executed procedure rather than on a merely plausible final response (Trivedi et al., 2024; Zhou et al., 2024). Test-time sampling improves success by generating multiple trajectories and selecting among them (Cobbe et al., 2021); either way, performance rests on two pillars—useful candidates and an effective selection rule.

A natural way to change the candidate distribution is to vary the prompt, yet rewording an instruction, proposing a different high-level plan, and supplying a different concrete execution procedure are distinct interventions. A reusable skill can provide tool-specific conventions, prerequisite checks, action ordering, and completion checks—operational details that change which trajectories an agent can actually execute. Since a skill supplied as text is itself a prompt, the scientific question is what its *content* contributes, not whether a file labeled "skill" is intrinsically better than one labeled "prompt."

We therefore ask two questions in sequence. First, *when does variation across reusable skills provide more useful exploration than paraphrases or model-generated strategies?* Because skills may simply be longer, contain more relevant knowledge, or be better selected, we pair the head-to-head comparison with length and content controls designed to discriminate among these explanations. Second, *how should a fixed execution budget $B = KM$ be divided across K skills with M executions each?* Increasing K expands the set of attempted procedures but reduces per-skill repetition and can make final selection harder. Throughout, we distinguish *oracle coverage*—whether any candidate succeeds—from *selected success*, the success of the candidate actually returned; an allocation can improve the former while degrading the latter. Our experiments separate two regimes that are never pooled: *label-free* comparisons retrieve skills from the task instruction alone, while

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

pilot-ranked comparisons additionally consume success-labeled screening rollouts and serve as a label-aware diagnostic, not a deployable baseline, with the screening cost reported separately.

Our findings are as follows. Single-skill coverage saturates early: most gains arrive before 16 rollouts, and 73–83% of initial failures remain unsolved even at 128 rollouts. Redistributing the same budget across distinct retrieved skills outperforms paraphrases and generated strategies in coverage, a direction that survives length matching, and an information ablation attributes roughly three points of coverage to concrete procedural detail. Our label-free relevance–diversity generator, SD-BoN, improves selected success over similarity Top-1 on every model–benchmark pair, while the label-aware allocation sweep traces an inverted-U curve. Finally, a coverage–selection decomposition shows that beyond the peak, additional breadth keeps producing successful candidates that practical selectors fail to return.

The paper contributes (i) a provenance-tracked open-source skill corpus with audit metadata and qualitative examples of retrieved procedural patterns; (ii) SD-BoN, a label-free, fixed-budget protocol for skill-conditioned exploration built on a relevance–diversity selector, evaluated against compute-matched baselines and a clearly separated label-aware ranking diagnostic; and (iii) a cross-model analysis of coverage, selection loss, and allocation that identifies when breadth across skills helps and when it merely creates successful candidates that a selector cannot reliably use.

2 Skill Curation and Single-Skill Scaling

2.1 Collection and Curation

We define a skill as a reusable package of procedural guidance, optionally accompanied by examples or scripts, that conditions agent execution. We collected 5,771 skill files with full provenance metadata from the public `VoltAgent/awesome-agent-skills` index at a pinned revision; after deduplication, a combined static and semantic audit, and a final human review pass, the frozen experiment pool contains 4,802 unique skills, shared by every model and benchmark and executed only inside benchmark sandboxes without host credentials. Skills flagged for leakage, real credentials, prompt injection, or non-repairable unsafe behavior are excluded; because the cleaning policy itself includes decontamination requirements, a manual spot check of 100 keep-labeled skills found no benchmark contamination, and human review of every review-labeled skill removed 28 more for quality failures, including residual contamination. Because dependency checks were documentary and the keep-side contamination check is sampled rather than exhaustive, the audit establishes neither runtime compatibility nor a contamination guarantee. The index is heterogeneous rather than benchmark-specific. Appendix A details the seven-step audit and cleaning pipeline, eligibility routing, stage counts, and worked cleaning and rejection cases (Table 5).

2.2 Skill Selection and Evaluation Protocol

Retrieval and pilot ranking. We embed the concatenation of each skill's normalized name, description, and instruction body with `BAAI/bge-m3` (Multi-Granularity, 2024); for each task x, cosine similarity against the public task instruction retrieves the 128 highest-scoring skills, forming $\mathcal{S}_x$. Retrieval never sees reference answers, evaluator state, or execution outcomes. In the pilot-ranked condition, each retrieved skill receives $R = 8$ screening executions. With benchmark success indicator $y_x(\tau) \in \{0, 1\}$, its pilot score is

$$\widehat{q}_x(s) = \frac{1}{R} \sum_{r=1}^{R} y_x(\tau_{x,s,r}^{\text{pilot}}), \qquad s_x^\star = \arg\max_{s \in \mathcal{S}_x} \widehat{q}_x(s). \tag{1}$$

Ties are resolved by retrieval similarity and then a fixed skill identifier; the ranking is model-specific and shared across budgets. Note that $s_x^\star$ maximizes the *observed* pilot score, not necessarily the true success probability, and the procedure is *label-aware*: it consumes $128 \times 8 = 1{,}024$ success-labeled screening rollouts per task and model. All pilot-ranked results use fresh evaluation trajectories, exclude pilot successes, and account the screening cost separately.

Evaluation settings and metrics. We evaluate AppWorld (168-task `test_normal` split), WebArena (812-task test set), GAIA (165-task validation set), and τ-bench (115-task retail domain)

with GPT-5.6-sol (xhigh), Claude Sonnet 4.5 (extended thinking), DeepSeek-V4 (thinking), and Kimi K3 (high); benchmark versions, decoding settings, and interaction limits appear in Appendix B. Each trajectory runs in an isolated copy of the initial task environment; failures and agent timeouts count as unsuccessful, and infrastructure failures are retried once with the same seed. The no-skill baseline keeps the same base instructions and tool interface but omits the retrieved skill. For same-skill scaling we evaluate $N \in \{1, 4, 8, 16, 32, 64, 128\}$ under the fixed skill $s_x^\star$. For task set $\mathcal{D}$, oracle coverage is

$$C(N) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbb{I}\big[\exists i \le N : y_x(\tau_{x,s_x^\star,i}) = 1\big], \tag{2}$$

empirical $\mathrm{pass@}N$ over one fixed candidate pool per task—an evaluation reference, not the success rate of a practical selector, and distinct from τ-bench's consistency-oriented pass^k metric (Yao et al., 2024).

Statistical protocol. We draw one 128-trajectory pool per task and condition and evaluate nested prefixes in a fixed rollout order, so coverage is non-decreasing in N. Every deterministic aggregate is an integer task count divided by $|\mathcal{D}|$, differences are computed from counts before rounding, and finer values arise only in explicitly labeled averages over selection seeds, pilot subsets, or models. We report point estimates without bootstrap intervals; because the same tasks are shared across models, cross-model means are descriptive equal-weight aggregates. Appendix B specifies seeds, retry policy, and the repeat structure.

2.3 Coverage Saturation and Failure Recovery

Figure 1 plots single-skill coverage for all 16 model–benchmark pairs; the full data appear in Table 8 (Appendix D). Gains concentrate early: GPT-5.6-sol on AppWorld improves by 6.55 percentage points from $N = 4$ to $N = 16$ but by only 0.60 points from $N = 16$ to $N = 64$, and the increment from 64 to 128 rollouts is at most 0.61 points across all pairs. We use *saturation* to describe these diminishing observed gains—not a proven asymptotic ceiling, and not a decline in selected success.

Recovery of initial failures. Marginal gains can shrink either because coverage approaches one or because repeated sampling rarely rescues the remaining failures. The conditional recovery rate distinguishes the two:

$$Q_N = \frac{C(N) - C(1)}{1 - C(1)}, \qquad C(1) < 1. \tag{3}$$

For nested candidate sets, Q_N is the fraction of first-rollout failures for which at least one of the remaining $N - 1$ rollouts succeeds. Aggregated over models (Table 9, Appendix D), increasing the budget from 1 to 128 adds roughly 12.7–13.7 percentage points of mean coverage yet recovers only 17.2–27.0% of initially failed model–task cases: 73.0–82.8% of initial failures remain uncovered, so saturation is not explained by coverage approaching 100%. Persistent failures alone do not demonstrate dependent executions either—task heterogeneity, without any within-task dependence, already produces flat aggregate curves (Appendix E). The operational question is whether a *different* skill changes success probabilities on tasks that repeated same-skill execution rarely recovers.

3 Skill-Conditioned Exploration

We proceed in two steps. Sections 3.1 and 3.2 ask whether retrieved skills add useful procedural content beyond alternative prompts, and which part of that content matters—without any success-labeled screening. Sections 3.3–3.5 then study how a collection of such procedures should be executed and selected over, distinguishing pilot-ranked diagnostics from label-free comparisons. Throughout, C denotes oracle coverage and P the success of an existing candidate returned by a fixed selector; formal definitions appear in Section 3.3.

3.1 Skills versus Prompt Diversity

We first compare sources of procedural variation on AppWorld with GPT-5.6-sol at $B = 128$: one retrieved skill with 128 rollouts; eight meaning-preserving paraphrases of that skill with 16 rollouts

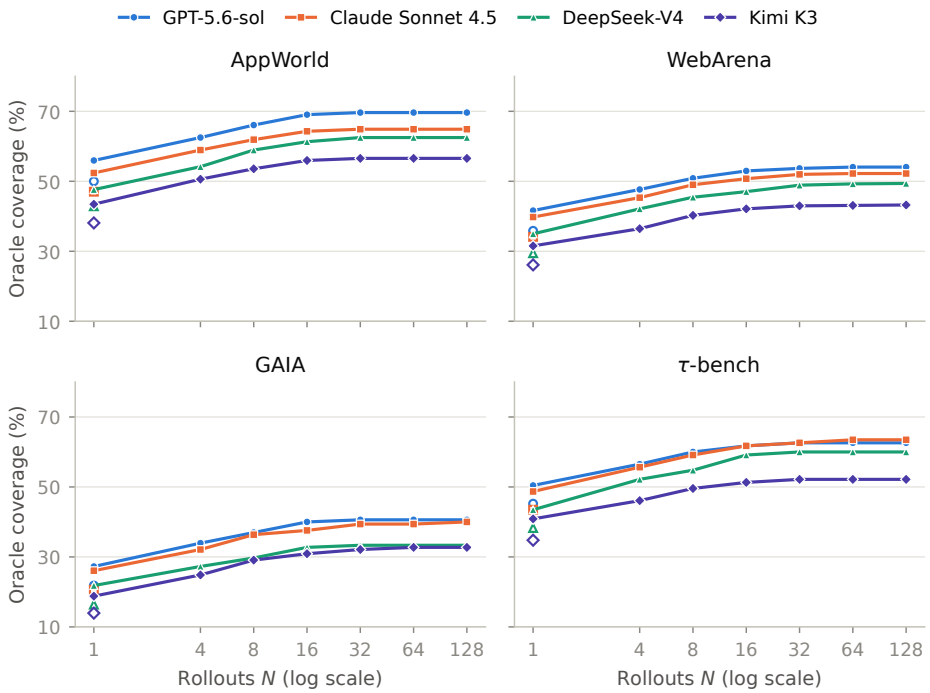

Figure 1: Single-skill oracle coverage against the rollout budget N (log scale), plotted from Table 8. Open markers at $N{=}1$ mark the no-skill pass@1 baseline. Gains concentrate before $N{=}16$, and every curve flattens far below full coverage.

each; eight task-conditioned strategies generated by the evaluated model itself with 16 rollouts each; and eight distinct retrieved skills with 16 rollouts each. All conditions use task-only similarity selection without success-labeled pilots, frozen prompts, and identical primitive tool access and execution limits. Distinct skills achieve the highest coverage and selected success, exceeding paraphrases by five covered tasks and three selected successes (2.98 and 1.79 points) and generated strategies by three covered tasks and one selected success (1.79 and 0.60 points; Table 10, Appendix D). The advantage is thus larger at generation than at final selection. Skill prompts are also longer, and the selected-success margin over generated strategies is a single task—this comparison motivates, but cannot substitute for, the controls that follow.

Comparison at matched instruction length. Table 11 (Appendix D) repeats the four generation strategies under a common instruction-length constraint of 4,096 tokens (± 64), with the rollout budget, base model, task set, tool access, outcome canonicalizer, and selector all fixed; rewriting and length choices were frozen on development tasks without evaluation feedback, and no executable capability is exclusive to the skill condition. The primary contrast—distinct skills versus generated strategies—yields three additional covered tasks and two additional selected successes (1.79 and 1.19 points) under paired task-level differences. The direction thus survives length matching, although the margins involve few tasks; the independent-run and cross-setting repeats in Appendix F quantify their stability.

3.2 What Does Procedural Content Contribute?

A task-specific strategy can state *what* to achieve without specifying *how* to execute it. We hypothesize that a useful skill supplies concrete intermediate decisions—tool and argument conventions, prerequisites, action ordering, and completion checks—and that variation across such procedures changes which failures are recoverable. This is a hypothesis about the information available to the agent, not about the intrinsic virtue of a skill file: a generated prompt could in principle contain the same information.

The factorial control in Table 12 (Appendix D) starts from the same eight similarity-retrieved source skills per task and crosses procedural detail with presentation: the *concrete* condition retains all applicable operational details, the *abstract* condition retains only the goal and high-level subgoals, and each appears either as structured steps or as continuous prose preserving the same instructions. All forms are normalized to the length constraint of Section 3.1 with identical tool and code access; transformations were frozen before evaluation and audited without success labels (Appendix F). Averaged over the two presentations, concrete detail adds 2.98 points of coverage and 2.38 points of selected success, positive under both presentations—consistent with the hypothesized role of operational information. Because the abstract condition deliberately deletes operational instructions, this is an information ablation on the retained bundle; it credits no individual instruction, nor does it show that a skill *format* is superior to a prompt.

A gain at fixed $K = 8$ could also reflect better individual procedures rather than a larger benefit from diversity. A targeted extension therefore repeats the concrete and abstract structured conditions at $K = 1$ with the same total budget: the two $K = 1 \to 8$ gains differ by 0.60 percentage points, a small interaction that identifies the breadth-versus-quality contrast without excluding a pure quality explanation. Together, the two controls establish that the observed advantage survives length matching and depends on concrete procedural content, while leaving the finer mechanism open. The remaining experiments ask how to allocate executions across the available procedures.

3.3 Skill Breadth under a Fixed Rollout Budget

A skill selector chooses $\mathcal{A}_{x,K} \subseteq \mathcal{S}_x$ with K distinct skills, and we allocate M rollouts to each:

$$B = KM, \qquad \mathcal{T}_x^{K,M} = \{\tau_{x,s,m} : s \in \mathcal{A}_{x,K},\ m = 1, \dots, M\}. \tag{4}$$

The pilot-ranked diagnostic uses the ranking of Section 2.2 and majority voting over all B candidates, so $K = 1$ recovers the setting of Section 2. Let h return an existing candidate $\widehat{\tau}_{x,h}$ from a pool. Its selected success rate and the pool's oracle coverage are

$$P_h(K, M) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} y_x(\widehat{\tau}_{x,h}), \tag{5}$$

$$C(K, M) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbb{I}\left[\exists \tau \in \mathcal{T}_x^{K,M} : y_x(\tau) = 1\right]. \tag{6}$$

Abstention counts as failure. Because selectors return existing candidates, $P_h \leq C$ within a pool, and the selection gap $G_h = C - P_h$ is the fraction of tasks with a successful candidate that was not selected. Environment-state evaluation is performed on the selected candidate's isolated execution, never on merged side effects. Here B counts evaluation rollouts.

Allocation sweep. At $B = 128$ we sweep $K \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ with $M = 128/K$. The primary $K = 8$ operating point was fixed on a 40-task AppWorld development split before any test task was run; the test-set sweep is shown to trace the allocation curve, but no test-set maximum informs the reported deployable setting. Figure 2 plots the label-aware GPT-5.6-sol diagnostic; the label-free SD-BoN comparison in Table 2 remains the practical headline. The 8×16 allocation improves majority-vote success over 1×128 by 2.38, 2.46, 1.82, and 2.61 points on AppWorld, WebArena, GAIA, and τ-bench (mean 2.32; GPT rows of Table 14, Appendix D); each benchmark attains its maximum at $K = 16$ (GAIA also ties at $K = 32$) and declines beyond it as each skill receives fewer repetitions—sharply on AppWorld, mildly on the other three benchmarks. The inverted-U pattern indicates a finite-budget allocation trade-off, though not by itself success complementarity: increasing K also introduces lower-ranked skills, and these estimates condition on one pilot ranking.

Table 14 (Appendix D) extends the comparison to every model and benchmark: the $K = 8$ gain is positive in all 16 model–benchmark pairs (1.21–2.61 points; benchmark-averaged gains of 2.32, 1.77, 1.74, and 1.58 for GPT-5.6-sol, Claude Sonnet 4.5, DeepSeek-V4, and Kimi K3), so the detailed GPT curve is representative rather than exceptional. Repeating the sweep for $B \in \{16, 32, 64, 128\}$ (Table 15), the cross-skill gain is positive at every tested budget (1.19–2.38 points): skill breadth remains useful across rollout budgets. Allocations selected retrospectively over test data are reported only as oracle envelopes and never enter this comparison.

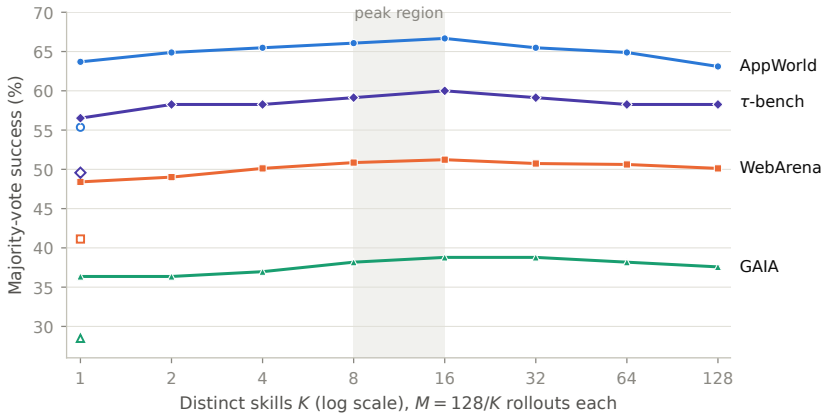

Figure 2: Label-aware allocation curves for GPT-5.6-sol at $B = 128$: majority-vote success as the fixed budget is split into K skills with $M = 128/K$ rollouts each. Open markers at K=1 show the no-skill 128-rollout baseline; the $K \in \{1, 8, 32, 128\}$ points also appear in Table 14. Each benchmark attains its maximum at $K = 16$; GAIA also ties at $K = 32$.

Task recovery and coverage loss. An aggregate gain does not reveal whether cross-skill sampling expands coverage or merely exchanges one set of solved tasks for another. With A_x and D_x indicating oracle success under same-skill and cross-skill sampling, we report the rescue rate $r_K = \Pr(A_x{=}0, D_x{=}1)$, the loss rate $\ell_K = \Pr(A_x{=}1, D_x{=}0)$, their difference $\Delta C_B(K) = r_K - \ell_K$, and the conditional rescue rate $r_K / \Pr(A_x{=}0)$ among tasks uncovered by the same-skill pool. The detailed decomposition (Table 16, Appendix D) uses GPT-5.6-sol on AppWorld, where we retain per-task trajectory identities. Net gains are positive but the lost fractions are nonzero, indicating task turnover rather than uniform dominance: at $K = 8$, 11 rescued and three lost tasks yield a net gain of eight tasks (4.76 points).

3.4 Label-Free Selection and Skill Composition

Selection policies. Fixing majority voting, we compare three label-free policies on the same retrieved pool: **Random-K**, uniform sampling without replacement; **Similarity Top-K**; and **Similarity-weighted sampling**, sequential softmax sampling over retrieval scores with temperature $\beta = 0.10$ chosen on a disjoint validation split (Equation (11), Appendix C). Randomized policies average over 10 selection seeds; at $K = 128$ all policies select the same set. We further introduce a label-free relevance–diversity selector. Starting from the 32 highest-similarity skills, it greedily adds

$$s_j = \arg\max_{s \in \mathcal{R}_j} \left[\lambda r(x, s) + (1 - \lambda) \min_{s' \in \mathcal{A}_j} d(z_s^{\text{text}}, z_{s'}^{\text{text}}) \right], \tag{7}$$

where r is normalized retrieval similarity, z^{text} is a static signature of tool names, resource domains, and procedural operators extracted from the skill body, d is cosine distance, and $\lambda = 0.70$ is fixed across all reported comparisons. Penalizing the minimum distance to the selected set is the classical maximum-marginal-relevance (MMR) rule (Carbonell & Goldstein, 1998). The resulting generator—MMR selection of $K = 8$ skills with $M = 16$ executions each at $B = 128$, using no success labels, pilot traces, or benchmark answers—is what we call **Skill-Diverse Best-of-N (SD-BoN)**; unless stated otherwise, it is paired with flat majority voting over all B candidates. We also test a hierarchical skill vote that first takes the modal outcome within each skill and then gives each skill one vote, preventing a single skill's unusually concentrated outcome distribution from dominating the pooled vote (Equation (10), Appendix C).

Similarity-weighted sampling improves modestly over uniform selection, whereas MMR adds 4.35–5.95 points over Similarity Top-K at the same rollout budget, and the hierarchical vote a further 0.37–1.74 points (Table 1); the all-model comparison below retains flat voting. Since MMR changes the composition of the skill set, this comparison does not isolate any individual instruction. Pilot

Table 1: Label-free skill-selection and voting ablation: selected success (%) at $B = 128$, $K = 8$ for GPT-5.6-sol on each benchmark's full test set. Random-K and Similarity-weighted rows average ten selection seeds; Similarity Top-K and MMR are deterministic.

Policy	AppWorld	WebArena	GAIA	τ-bench
Random-K	58.33	44.52	31.45	53.04
Similarity Top-K	59.52	45.69	32.73	53.91
Similarity-weighted	59.88	45.94	32.48	54.52
SD-BoN (MMR, flat vote)	65.48	50.62	37.58	58.26
SD-BoN + hierarchical vote	**66.67**	**50.99**	**38.79**	**60.00**

Table 2: Label-free SD-BoN (MMR selection, flat majority voting) at $B = 128$. P_1, P_8 are majority-vote success, C_8 is oracle coverage of the same MMR candidate pool, and $G_8 = C_8 - P_8$ is its selection gap (all in %). The static selector is shared across models; no success-labeled pilot is used.

Model	Benchmark	Top-1 P_1	MMR P_8	Gain (pp)	MMR C_8	MMR G_8
GPT-5.6-sol	AppWorld	57.14	65.48	8.33	73.81	8.33
	WebArena	43.10	50.62	7.51	58.13	7.51
	GAIA	30.30	37.58	7.27	44.85	7.27
	τ-bench	51.30	58.26	6.96	66.96	8.70
Claude Sonnet 4.5	AppWorld	54.76	61.31	6.55	69.05	7.74
	WebArena	42.36	49.01	6.65	55.54	6.53
	GAIA	29.09	36.97	7.88	44.24	7.27
	τ-bench	49.57	57.39	7.83	66.09	8.70
DeepSeek-V4	AppWorld	50.60	59.52	8.93	66.67	7.14
	WebArena	37.56	45.44	7.88	53.20	7.76
	GAIA	24.85	30.30	5.45	36.97	6.67
	τ-bench	45.22	54.78	9.57	62.61	7.83
Kimi K3	AppWorld	46.43	53.57	7.14	61.31	7.74
	WebArena	34.48	40.15	5.67	46.55	6.40
	GAIA	21.21	28.48	7.27	36.36	7.88
	τ-bench	42.61	47.83	5.22	55.65	7.83

ranking, for its part, relies on noisy eight-trial estimates and does not optimize mixture voting, so it is a label-aware diagnostic rather than a performance upper bound.

Does breadth help without success-labeled screening? The primary test pairs Similarity Top-1 with Similarity Top-8 under identical base instructions, tools, rollout budgets, and majority voting (Table 13, Appendix D): selected success improves by 2.38–2.61 points and coverage by 3.45–5.22 points across the four benchmarks, so the allocation benefit survives removal of task-success feedback. Under matched total generation attempts (1,152 per task), SD-BoN's flat vote comes within 0.87 points of the label-aware pilot-ranked configuration on every benchmark while using only 128 label-free rollouts, and the hierarchical vote exceeds it on every benchmark (Table 17, Appendix D).

Across all models (Table 2), the SD-BoN gain over similarity Top-1 ranges from 5.22 to 9.57 points and averages 7.26 points with the 16 cells weighted equally; percentages and gains are computed from integer counts before rounding. The pilot-ranked mean gain of 1.85 points uses a different Top-1 baseline, so the difference between the two gains does not isolate a selector effect.

Coverage belongs to a candidate pool. The C_8 and G_8 columns of Table 2 refer to the same SD-BoN trajectories as P_8; selection gaps of 6.40–8.70 points show that the SD-BoN result does not require near-perfect voting. The constraint $P \leq C$ applies only within a candidate pool: pilot-selected single-skill coverage is no upper bound on coverage from different skills, and the known-probability bound of Appendix E concerns a true per-task maximizer, not an eight-trial pilot winner. In the observed pools, SD-BoN's selected success stays below the pilot-selected single-skill coverage and its coverage does not exceed the pilot Top-8 coverage. Intuitively, pilot ranking optimizes estimated single-trajectory success while MMR penalizes overlap in procedural signatures: complementary procedures improve coverage, but final voting can still discard successful candidates. Reporting C_8 and G_8 together separates the two effects. Two further diagnostics appear in Appendix D: at matched

Table 3: AppWorld trajectory selection at $B = 128$ for GPT-5.6-sol. Each column shares exactly the same candidates across selectors.

Selector	$K = 1$	$K = 8$	$K = 32$	$K = 128$
Majority voting	63.69	66.07	65.48	63.10
LLM reward judge	64.88	68.45	66.67	64.88
Verifier	66.07	69.64	68.45	67.26
Oracle coverage	69.64	74.40	75.60	74.40

sorted pilot-score profiles, skill sets with higher behavioral diversity add 1.40 points of coverage and of selected success on a matched AppWorld subset (Table 18), and growing the ranking evidence from $R = 1$ to $R = 8$ pilot trials shrinks—but does not eliminate—the oracle-coverage advantage of spreading rollouts (Table 19), consistent with skill breadth partly compensating for uncertain ranking.

3.5 From Candidate Coverage to Selected Success

We now hold the pilot-ranked skill sets and generated trajectories fixed and apply each selector to the same candidate pools, isolating selection from generation.

Selectors. A benchmark-specific canonicalizer g_x maps each candidate to an observable task-relevant outcome. Majority voting selects the most frequent outcome and a representative trajectory from it (Equation (9), Appendix C); unparseable outcomes abstain but consume budget, and ties use fixed seeded rules. The LLM reward judge (GPT-5.6-luna at xhigh reasoning effort) assigns a single 0–1 score from the task instruction, visible trajectory, tool outputs, and final response; the verifier applies benchmark-specific observable completion checks and passes only unresolved semantic constraints to the same model. Neither selector sees reference answers, hidden evaluator state, or the benchmark success label; full specifications appear in Appendix C.

Every practical selector benefits from $K = 8$ (Table 3): gains over $K = 1$ are 2.38 points for voting and 3.57 for both the LLM reward judge and the verifier, while oracle coverage gains 4.76 points. The verifier closes more of the selection gap without reaching the oracle, and all selectors degrade at $K = 128$. Table 21 (Appendix D) confirms the pattern for all models: the verifier improves over majority voting on every model–benchmark pair, and the oracle coverage increase consistently exceeds the selected-success increase. The coverage-to-selection distinction is therefore not specific to GPT-5.6-sol, even though the failure diagnostics below use AppWorld for that model.

Coverage-to-selection conversion. The exact decomposition

$$\Delta P_{h,B}(K) = \underbrace{r_K - \ell_K}_{\Delta C_B(K)} - \Delta G_{h,B}(K), \qquad G_h = C - P_h, \tag{8}$$

shows that a broader skill set improves final success exactly when its net coverage gain exceeds the increase in selection gap. This is an accounting identity rather than a causal theorem: it separates failure to generate a successful candidate from failure to select an available one. Figure 3 visualizes the divergence on AppWorld (Table 20, Appendix D): additional skills shrink the largest wrong-outcome group, yet the selection gap and the mass of mixed outcome groups grow. Increasing K from 8 to 32 raises coverage by 1.19 points but lowers voting success by 0.60 points—the selection gap grows by three tasks, more than offsetting the two-task coverage gain—so reducing agreement on wrong outcomes is insufficient to improve final success.

The same bottleneck appears for all four models on AppWorld (Table 22, Appendix D): from $K = 8$ to $K = 32$, oracle coverage changes by 0.00 to $+1.79$ points while majority-vote success falls by 0.60–1.79 points and the selection gap grows by 0.60–3.57 points (all computed from paired integer counts). These changes involve few tasks, and the three-run comparison in Appendix F keeps execution variation separate from prompt-construction variation; the diagnostics describe the divergence without establishing a causal fragmentation mechanism. The observed mismatch between coverage-maximizing and success-maximizing allocations (Appendix E) argues for choosing breadth for the *deployed selector*, not for oracle coverage.

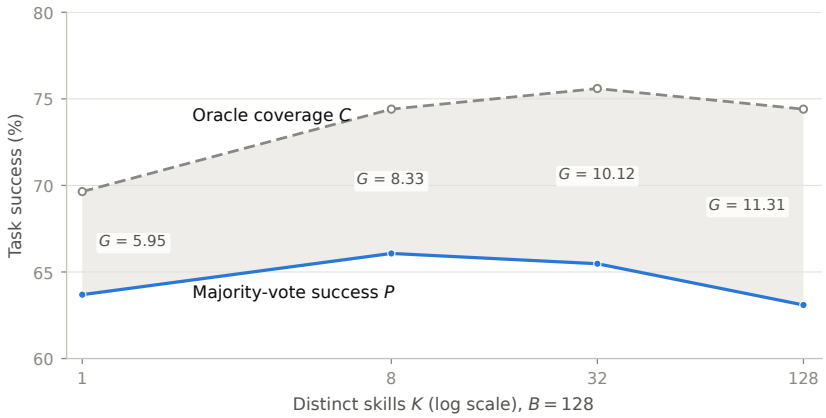

Figure 3: Coverage–selection divergence on AppWorld for GPT-5.6-sol at $B = 128$ (Table 20). From $K = 8$ to $K = 32$, oracle coverage C continues to rise while majority-vote success P falls, so the selection gap $G = C - P$ (shaded) widens: broader skill sets produce more successful candidates that the selector fails to return.

4 Related Work

Test-time sampling, selection, and diversity. Sampled solutions are selected by learned outcome and process verifiers (Cobbe et al., 2021; Lightman et al., 2024), aggregated by self-consistency and its open-ended generalization (Chen et al., 2023), or explored by deliberate tree search (Yao et al., 2023). The value of additional samples depends on the proposal distribution, verification method, and problem difficulty (Snell et al., 2024): repeated sampling solves many additional problems when a reliable filter exists (Li et al., 2022; Brown et al., 2024), and voting ensembles improve with size (Li et al., 2024), but additional samples can also stabilize a wrong answer—which motivates our paired recovery reporting over fixed nested pools. Diversity has been injected through varied prompts and weighted voting (Li et al., 2023), complementary demonstrations (Ye et al., 2023), and iterative self-feedback (Madaan et al., 2023). A skill encoded as instructions is itself a form of prompting, so our length and content controls frame the skill advantage as an information ablation, with skill count and executions per skill as separate allocation variables.

Agents, reusable procedures, and memory. Tool-using agents interleave reasoning with environment actions (Yao et al., 2022; Schick et al., 2023), and procedural knowledge has been reused through executable skill libraries (Wang et al., 2023), trajectory exemplars (Zheng et al., 2024), stacked policies (Sodhi et al., 2023), induced workflows (Wang et al., 2024), and stored interaction traces (Shinn et al., 2023)—typically optimizing a memory across tasks. We instead use a frozen public skill pool, vary only which procedures are explored at test time, and write no outcome back: our claim concerns compute allocation across reusable procedures, not lifelong memory formation. AppWorld, WebArena, GAIA, and τ-bench (Trivedi et al., 2024; Zhou et al., 2024; Mialon et al., 2024; Yao et al., 2024), like related agent evaluation suites (Deng et al., 2023; Xie et al., 2024; Jimenez et al., 2024; Liu et al., 2024), score executed procedures rather than short textual answers, motivating our benchmark-specific outcome representations.

5 Conclusion

Single-skill sampling saturates well below full coverage, and redistributing the same budget across distinct retrieved skills recovers tasks that repetition does not. The label-free SD-BoN generator improves selected success over similarity Top-1 by 7.26 points on average across 16 model–benchmark cells (range 5.22–9.57), while the label-aware pilot-ranking diagnostic shows a smaller allocation gain at substantial screening cost. The coverage–selection decomposition explains why candidate quality, skill composition, and final selection must be evaluated together: beyond a moderate breadth, exploration keeps producing successful candidates that practical selectors fail to return.

LIMITATIONS

The method leaves four design choices open for future work. First, SD-BoN fixes one diversity rule—MMR over static text signatures with $\lambda = 0.70$—and one allocation—$K = 8$, $M = 16$; learned diversity criteria and per-task adaptive breadth are natural extensions. Second, the label-free and label-aware regimes use different rankings and baselines, so their difference does not isolate a selector effect; a shared-candidate design that varies only the selector could test this directly. Third, the content comparisons operate on retained instruction bundles; instruction-level interventions could localize which operational details drive the gain. Finally, outcome canonicalization and majority voting rely on the verifiable task-level success criteria that all four benchmarks provide; extending skill-conditioned exploration to open-ended, rubric- or judge-scored tasks remains open.

ETHICS STATEMENT

The study evaluates agents on public benchmarks inside isolated sandbox copies of each task environment, without host credentials, and skills are inserted as delimited untrusted context. The audit pipeline excludes skills flagged for prompt injection, credential exposure, or unsafe behavior, but it does not certify the remaining pool as universally safe, and scoring a selected isolated execution does not show that state-changing actions can be safely replayed in a live environment. Public availability of a skill is not redistribution permission; any release must respect source licenses (Appendix A).

REPRODUCIBILITY STATEMENT

Section 2.2 specifies retrieval, pilot ranking, the evaluated benchmarks and models, and the statistical protocol; every deterministic percentage corresponds to an integer task count over the stated denominators. Appendix A records the seven-step audit and cleaning pipeline, the stage counts in Table 4, and worked cleaning and rejection cases; Appendix B gives benchmark versions, decoding settings, interaction limits, and the retry policy; Appendix C specifies outcome canonicalization, voting rules, and the judge and verifier interfaces.

AI USE STATEMENT

Large language models were used to assist manuscript writing, editing, and numerical consistency checks. Models are also explicit components of the method: GPT-5.6-sol performs the semantic corpus audit (Appendix A), the evaluated models generate all trajectories, and GPT-5.6-luna serves as the reward judge and semantic verifier (Appendix C). The authors remain responsible for all content; model-assisted consistency checks do not replace author verification of tables, claims, and citations.

REFERENCES

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Jaime G Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, volume 98, pp. 290941–291025, 1998.

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. *arXiv preprint arXiv:2311.17311*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *arXiv preprint arXiv:2402.05120*, 2024.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 5315–5333, 2023.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Multi-Linguality Multi-Functionality Multi-Granularity. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Paloma Sodhi, SRK Branavan, Yoav Artzi, and Ryan McDonald. Step: Stacked llm policies for web actions. *arXiv preprint arXiv:2310.03720*, 2023.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 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)*, pp. 16022–16076, 2024.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Veselin Stoyanov, Greg Durrett, and Ramakanth Pasunuru. Complementary explanations for effective in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4469–4484, 2023.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations*, volume 2024, pp. 19036–19066, 2024.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

A Corpus Provenance and Eligibility

The skill pool is produced by seven ordered steps:

1. **Collection and provenance pinning.** Every listed skill is downloaded with its listing URL, resolved URL, source commit, and per-file SHA-256 digest recorded, yielding 5,771 raw files.
2. **Exact deduplication.** Content hashing removes 590 byte-identical duplicates, leaving 5,181 unique contents; each duplicate file keeps a pointer to its canonical copy.
3. **Static scan.** Deterministic rules detect malformed frontmatter, missing license metadata, absent local references, oversized skills, secret-like strings, dangerous commands, prompt-injection phrases, and benchmark- or answer-like language, flagging 4,138 of the 5,181 unique contents; the findings are attached to the audit input.
4. **Isolated semantic audit.** Each unique content is audited in a disposable session (GPT-5.6-sol at xhigh reasoning effort) against a fixed JSON schema: a `keep`/`review`/`reject` decision, a quality score, domain tags, boolean flags for answer leakage, benchmark-specific content, prompt injection, secrets or credentials, unsafe or privileged behavior, and private-context requirements, a dependency list, and a per-file edit record. The audit receives the complete skill up to a 280,000 character bound together with the static findings and provenance metadata; truncated items cannot enter the automatic pool. This stage returns 4,713 `keep`, 117 `review`, and 351 `reject` decisions.
5. **Minimal cleaning.** Permitted edits remove promotional or duplicated text, normalize metadata and punctuation, and replace credential-like example values with explicit placeholders such as `<DB_PASSWORD>`; technical procedures are never silently rewritten and missing dependencies are never invented. Raw originals and edit records are retained alongside every cleaned file.
6. **Eligibility routing.** Automatic use requires a `keep` decision, quality score 3, no static pool blocker, and no semantic flag; this stricter routing maps the 4,713 keep decisions to 75 materialized automatic files.
7. **Human verification.** Because the cleaning policy includes decontamination requirements, two manual checks follow the semantic audit: a random spot check of 100 keep-labeled skills found zero benchmark contamination, and a human review of all 117 review-labeled skills removed 28 for unresolved quality failures, including residual contamination. The 89 surviving review-labeled skills and the 4,638 non-automatic keep decisions are review-gated, giving 4,727 unique review files.

Dependency checks throughout are documentary—commands are not executed and dependencies are not installed—so the audit establishes neither runtime compatibility nor universal safety, and the keep-side contamination check is sampled rather than exhaustive. Unresolved licensing or other review flags are preserved rather than treated as clearance to use or redistribute a skill.

The final experiment manifest is the 4,802-row `usable_manifest.jsonl` file, containing 75 automatic and 4,727 review-gated skills. All experiments preserve the status as a covariate and run skills inside benchmark sandboxes. The manifests, audit JSON, model responses, raw files, cleaned files, and cleaning changes are retained under `data/raw`, `data/audit`, and `data/cleaned`. Table 4 summarizes the stage counts, Table 5 illustrates the cleaning edits and rejection decisions with cases drawn from the recorded audit artifacts, and Table 6 shows representative retrieved procedural patterns.

Public availability is not redistribution permission. A release must exclude or separately gate artifacts whose licenses do not permit redistribution, even when those artifacts were available for internal evaluation.

B Execution and Statistical Settings

The evaluated configurations are `gpt-5.6-sol` with xhigh reasoning, `claude-sonnet-4.5` with extended thinking, `deepseek-v4` with thinking enabled, and `kimi-k3` with high reasoning.

Table 4: Collection and audit outcomes. Blocks describe different stages and must not be added together.

Stage or outcome	Count
Downloaded skill files	5,771
Exact duplicate files removed	590
Unique contents audited	5,181
Keep / review / reject	4,713 / 117 / 351
Non-rejected unique candidates	4,830
Removed by human review of review-labeled skills	28
Eligible for automatic use	75
Review-gated	4,727
Final experiment-eligible skills	4,802

Table 5: Cleaning and rejection cases from the recorded audit artifacts. Each row cites the source label from the pinned index and paraphrases the audit's recorded edit or rejection rationale. Cleaned skills retain their technical procedures; rejections remove the whole skill from the pool.

Source label	Outcome	Audit finding	Recorded action
`sanjay3290/postgres`	cleaned	Example connection string contains a concrete password value.	Replaced the example password with the explicit placeholder `<DB_PASSWORD>`; the schema-inspection and query procedures are unchanged.
`Linked-API/linkedin`	cleaned	Illustrative API token and email address in the setup snippet.	Replaced token and email examples with explicit placeholders and normalized description punctuation to ASCII.
`Rootly-AI-Labs/rootly-incident-responder`	cleaned	Promotional "Scale Beyond Manual Response" section with a demo-booking link.	Removed the promotional section; retained the incident workflow, troubleshooting guidance, MCP setup, and human approval gates unchanged.
`obra/executing-plans`	cleaned	Promotional paragraph recommending a specific agent platform.	Removed the recommendation paragraph; preserved the plan-execution workflow and safety boundaries without inventing missing metadata.
`obra/using-superpowers`	rejected	Mandates invoking the skill before any response and attempts to override default agent behavior.	Rejected as prompt injection: a self-referential instruction layer rather than a bounded reusable procedure.
`anthropics/canvas-design`	rejected	Embeds fabricated prior user instructions and a mandatory refinement step based on that fabricated context.	Rejected as prompt injection despite otherwise strong domain-specific guidance.
`anthropics/mcp-builder`	rejected	Evaluation example contains a concrete question with its answer value.	Rejected for answer leakage; the referenced local guides were also absent from the submission.
`muratcankoylan/context-compression`	rejected	Repeated benchmark-specific identifiers, exact compression ratios, and a named evaluation dataset.	Rejected for material benchmark contamination of a reusable-skill pool.

These model configurations specify the evaluation protocol. We use temperature 0.7, top-p 0.95, and a 16,384-token output limit. The system prompt, tool schemas, and benchmark harness remain fixed within a model and benchmark; the skill is inserted as a delimited untrusted procedural-context block immediately before the task instruction. AppWorld permits at most 40 API calls, WebArena 30 browser actions, GAIA 20 tool invocations, and τ-bench 30 interaction turns. Wall-clock timeout is 15 minutes per trajectory.

Infrastructure errors are retried once with the same seed; task-level errors, invalid actions, and timeouts are counted as failures. Each task and condition has one fixed 128-trajectory candidate pool. Nested-prefix estimates use the recorded rollout order, and prefix ordering does not count as an extra rollout. Every deterministic percentage corresponds to an integer task count; for example, 58.26% on τ-bench represents $67/115$. For $R \in \{1, 2, 4\}$, the sensitivity analysis uses 20 nested-subset constructions from the same eight pilot traces; $R = 8$ uses the full ranking once. Execution repeats have separate run identifiers and fixed pilot rankings. A full rerun of screening is a different condition and is not pooled with these repeats.

Table 6: Representative retrieved skill patterns and the failure mode they address. Names are normalized archetypes rather than source labels.

Pattern	Procedural content	Typical rescued failure
API preflight and verification	Inspect schema, authenticate only through the sandbox, issue the smallest mutation, then read back the changed record.	Wrong field names or successful-looking but uncommitted updates.
Browser search–filter–confirm	Search before opening a result, preserve the target identifier, apply filters before mutation, and verify the resulting page state.	Stale-page actions and edits to the wrong entity.
Structured data handoff	Parse files into a typed intermediate table, validate units and missing values, then compute and format the final answer.	Tool output that is numerically plausible but uses the wrong unit or row.

Table 7: Benchmark configuration.

Benchmark	Version	Split / subset	Tasks
AppWorld	0.1.3.post1	`test_normal`	168
WebArena	0.2.0	test	812
GAIA	2023	validation	165
τ-bench	1.0	retail	115

C Selector Specifications

For AppWorld and WebArena, outcome canonicalization uses the visible final response together with observable application state changes: affected entity type, operation type, and identifiers created, removed, or updated. For GAIA, we lowercase text, apply Unicode NFKC normalization, remove surrounding prose, canonicalize numeric formatting and units, and retain the final answer span. For τ-bench, the representation combines the visible assistant resolution, database mutation summary, and user-confirmation status. Hidden reference answers and evaluator-only state are never exposed to practical selectors.

Invalid or unparseable outputs form an abstention group and consume budget. If every output abstains, the task fails. Modal-vote ties are broken with a seed derived from the task identifier. Within a selected group, the representative is the valid trajectory with the shortest normalized action trace, with the same seed resolving ties. Mixed groups, in which canonicalized outcomes hide a success-relevant difference, are retained for the diagnostic in Table 20.

Voting rules. Majority voting selects the most frequent canonical outcome,

$$\widehat{a}_x = \arg\max_a \sum_{\tau \in \mathcal{T}_x^{K,M}} \mathbb{I}[g_x(\tau) = a], \tag{9}$$

without a strict-majority requirement. The hierarchical skill vote first takes the modal outcome within each skill and then gives each skill one vote,

$$a_s = \arg\max_a \sum_{m=1}^{M} \mathbb{I}[g_x(\tau_{x,s,m}) = a], \qquad \widehat{a}_x = \arg\max_a \sum_{s \in \mathcal{A}_{x,K}} \mathbb{I}[a_s = a], \tag{10}$$

with seeded, label-independent tie-breaking at both levels.

Skill-selection baselines. Similarity-weighted sampling draws skills sequentially without replacement with

$$p(s \mid x, \mathcal{R}) = \frac{\exp(r(x,s)/\beta)}{\sum_{s' \in \mathcal{R}} \exp(r(x,s')/\beta)}, \tag{11}$$

where $\mathcal{R}$ is the remaining skill set and the temperature $\beta = 0.10$ is chosen on a disjoint validation split. In the MMR selector of Equation (7), the first skill is the highest-similarity item; the pilot-trace signature used in the composition analysis of Appendix D.5 is not available to this label-free selector.

Table 8: Single-skill oracle coverage (%) from one fixed, nested 128-trajectory pool per task. "w/o skill" reports the no-skill pass@1 baseline. All columns under "with skill" use the same selected skill for a given task and model.

	w/o skill	With skill: oracle pass@N						
Model	$N = 1$	$N = 1$	$N = 4$	$N = 8$	$N = 16$	$N = 32$	$N = 64$	$N = 128$
AppWorld								
GPT-5.6-sol	50.00	55.95	62.50	66.07	69.05	69.64	69.64	69.64
Claude Sonnet 4.5	47.02	52.38	58.93	61.90	64.29	64.88	64.88	64.88
DeepSeek-V4	42.86	47.62	54.17	58.93	61.31	62.50	62.50	62.50
Kimi K3	38.10	43.45	50.60	53.57	55.95	56.55	56.55	56.55
WebArena								
GPT-5.6-sol	35.84	41.63	47.66	50.86	52.96	53.69	54.06	54.06
Claude Sonnet 4.5	34.11	39.78	45.32	49.01	50.74	51.97	52.22	52.22
DeepSeek-V4	29.43	34.98	42.12	45.44	47.04	48.89	49.26	49.38
Kimi K3	26.11	31.53	36.45	40.27	42.12	42.98	43.10	43.23
GAIA								
GPT-5.6-sol	21.82	27.27	33.94	36.97	40.00	40.61	40.61	40.61
Claude Sonnet 4.5	20.61	26.06	32.12	36.36	37.58	39.39	39.39	40.00
DeepSeek-V4	16.36	21.82	27.27	29.70	32.73	33.33	33.33	33.33
Kimi K3	13.94	18.79	24.85	29.09	30.91	32.12	32.73	32.73
τ-bench								
GPT-5.6-sol	45.22	50.43	56.52	60.00	61.74	62.61	62.61	62.61
Claude Sonnet 4.5	43.48	48.70	55.65	59.13	61.74	62.61	63.48	63.48
DeepSeek-V4	38.26	43.48	52.17	54.78	59.13	60.00	60.00	60.00
Kimi K3	34.78	40.87	46.09	49.57	51.30	52.17	52.17	52.17

Table 9: Recovery analysis derived from Table 8. Coverage columns are percentages; gains are percentage points. Each coverage is an unweighted mean across the four models.

Benchmark	$\bar{C}(1)$	$\bar{C}(128)$	$\Delta_{1\to16}$	$\Delta_{16\to128}$	$\bar{Q}_{128}$ (%)
AppWorld	49.85	63.39	12.80	0.74	27.00
WebArena	36.98	49.72	11.24	1.51	20.22
GAIA	23.48	36.67	11.82	1.36	17.23
τ-bench	45.87	59.57	12.61	1.09	25.30

The LLM reward judge is GPT-5.6-luna at xhigh reasoning effort and is used only as a common selector, never as the generator of the strategy baseline. Its input contains the public task, visible trajectory, tool outputs, and final response; the prompt requests a 0–1 task-completion score. The verifier first evaluates explicit observable constraints, such as required entities, attribute changes, and prohibited extra mutations, then sends only unresolved semantic constraints to the same model. Its candidate score is the mean constraint satisfaction probability. Prompts and three-shot scoring examples are fixed using development tasks and do not contain test answers.

For state-changing tasks, scoring a selected isolated execution does not show that it can be safely replayed or committed in a live environment. Deployment requires a separate execution and commitment policy. SD-BoN is evaluated only offline; it never merges state-changing actions across skills.

D Extended Results and Diagnostics

This appendix collects the full numerical results and secondary experiments referenced from Sections 2 and 3.

D.1 Single-Skill Scaling

Table 8 reports the complete single-skill coverage data plotted in Figure 1, and Table 9 the derived recovery statistics.

Table 10: AppWorld prompt-level control for GPT-5.6-sol at $B = 128$, averaged over the full 168-task set.

Generation condition	Coverage	Selected success	Mean input tokens / rollout
One skill, repeated	63.10	57.14	3,864
Same-skill paraphrases	64.29	57.74	3,902
Generated strategies	65.48	58.93	3,526
Distinct retrieved skills	**67.26**	**59.52**	4,178

Table 11: Length-controlled comparison on AppWorld with GPT-5.6-sol, $B = 128$, majority voting, and no success-labeled screening. C and P are percentages. Every condition is newly controlled; the scores in Table 10 are not reused as length-matched results.

Generation condition	C	P	Instruction tokens	Total tokens / task
One skill, repeated	62.50	56.55	4,087	824,000
Same-skill paraphrases	63.69	57.14	4,102	831,000
Generated strategies	64.88	58.33	4,090	814,000
Distinct retrieved skills	66.67	59.52	4,097	846,000

D.2 Prompt and Content Controls

Table 10 reports the uncontrolled prompt-diversity comparison of Section 3.1, Table 11 its length-matched version, and Table 12 the content–presentation factorial of Section 3.2. The distinct-skill condition of Table 10 reuses the full-test-set Similarity Top-8 skill sets and candidate bank from Table 1 rather than an independent rerun; the generated-strategy condition holds the reasoning configuration of the evaluated model fixed. In the length-controlled comparison, actual instruction length, total trajectory tokens, and strategy-construction cost are recorded separately, since matching prompt length does not by itself match total compute.

For $Y \in \{C, P\}$, the average detail contrast reported in Section 3.2 is

$$\Delta_{\text{detail}} Y = \tfrac{1}{2} \sum_{f \in \{\text{structured}, \text{prose}\}} \big(Y_{\text{concrete},f} - Y_{\text{abstract},f}\big). \tag{12}$$

In the $K = 1$ extension, the concrete and abstract structured conditions obtain selected-success values of 56.55 and 54.76, so the two $K = 1 \to 8$ gains differ by 0.60 points. On AppWorld, the label-free Top-1 versus Top-8 endpoints of Table 13 are also visible in Table 10: breadth adds seven covered tasks and four selected successes.

D.3 Allocation Diagnostics

Figure 2 in the main text plots the full label-aware allocation sweep for GPT-5.6-sol; Table 14 reports the $K \in \{1, 8, 32, 128\}$ allocations for all models, Table 15 varies the total budget, Table 13 reports the label-free Similarity Top-1 versus Top-8 endpoints of Section 3.4, and Table 16 decomposes the AppWorld coverage change into rescued and lost tasks. At $K = 8$, seven of the eleven rescued AppWorld tasks involve multiple applications and eight fall below the median pilot-score margin, though these strata alone do not establish a subgroup mechanism.

D.4 Generation-Attempt-Matched Comparison

Table 17 matches the total number of generation attempts per task. SD-BoN spends its attempts on eight distinct skills and never inspects success labels: its flat-vote variant comes within 0.87 points of the label-aware pilot-ranked row on every benchmark, and its hierarchical-vote variant exceeds that row throughout. For the generated-strategy row, the evaluated model proposes the candidate strategies and the same pilot protocol selects among them; the pilot-ranked row is retained as a label-aware diagnostic, not as evidence that labeled routing underlies the label-free gain.

Table 12: Content–presentation control, AppWorld and GPT-5.6-sol. All conditions use the same source skill sets, $K = 8$, $M = 16$, matched instruction length, and majority voting. C and P are percentages.

Procedural detail	Presentation	C	P	Instruction tokens
Concrete	Structured	66.07	58.93	4,088
Concrete	Prose	65.48	58.33	4,093
Abstract	Structured	63.10	56.55	4,090
Abstract	Prose	62.50	55.95	4,097

Table 13: Label-free allocation, GPT-5.6-sol, $B = 128$. Both conditions use similarity ranking. P_1, P_8 are percentages; differences are percentage points.

Benchmark	P_1	P_8	ΔP	ΔC
AppWorld	57.14	59.52	2.38	4.17
WebArena	43.10	45.69	2.59	3.45
GAIA	30.30	32.73	2.42	4.24
τ-bench	51.30	53.91	2.61	5.22

D.5 Quality-Matched Composition

Increasing K in a Top-K sweep simultaneously admits lower-ranked skills and reduces repetition. To examine composition in isolation, we hold $B = 128$, $K = 8$, and $M = 16$ fixed and compare eight-skill sets with matched sorted pilot-score vectors. Each skill is characterized by a normalized histogram of tool categories and adjacent tool-category transitions from its pilot traces (zero-tool executions receive a dedicated category). With $z_{x,s}$ this signature, set diversity is

$$d_x(\mathcal{A}) = \frac{2}{K(K-1)} \sum_{s<t,\ s,t\in\mathcal{A}} \left(1 - \frac{z_{x,s}^{\mathsf{T}} z_{x,t}}{\|z_{x,s}\|\|z_{x,t}\|}\right). \tag{13}$$

For each task we sample 2,000 candidate sets and retain lower- and higher-diversity sets whose sorted pilot scores differ by at most 0.125 at every position; a valid pair exists for 143 of 168 AppWorld tasks. All rows in Table 18, including the Top-8 reference (full-set value 66.07%), are recomputed on exactly this 143-task subset. Higher behavioral diversity is associated with two additional covered tasks and two additional selected successes (1.40 points each) at matched observed pilot quality. Because eight-trial scores are noisy and different skills may carry different knowledge, diversity is not thereby isolated as the sole causal factor. Note that the Similarity Top-K row of Table 1 is a label-free full-test-set condition (its AppWorld value of 59.52% covers all 168 tasks), whereas the Pilot Top-8 reference here uses a different selector recomputed on the matched subset; the two are not the same Top-8 condition.

D.6 Sensitivity to Ranking Evidence

We vary the number of pilot trials used for ranking, $R \in \{1, 2, 4, 8\}$, with the evaluation budget fixed at 128 (Table 19). Nested subsets of the eight pilot executions supply ranking evidence, evaluation candidates remain disjoint, and the subset analysis uses 20 constructions for $R \in \{1, 2, 4\}$ ($R = 8$ uses the full ranking once). More ranking evidence reduces ties and shrinks the oracle-coverage advantage of spreading rollouts, consistent with skill breadth partly compensating for uncertain ranking; a positive gap remains at $R = 8$. This diagnostic does not show that breadth beats the unknown true best skill—under fixed independent success probabilities that skill maximizes oracle coverage (Appendix E)—and conditions have unequal screening costs, so it is not an end-to-end efficiency comparison.

D.7 Selection Diagnostics

Table 20 gives the AppWorld voting diagnostics plotted in Figure 3; Table 21 extends the selector comparison to all models, and Table 22 the $K = 8 \to 32$ selection bottleneck.

Table 14: Cross-skill allocation summary for all models at $B = 128$. Entries are majority-vote task success (%). Each deterministic value is computed on one fixed task-level candidate pool; $K = 1$ and $K = 8$ are the primary comparison.

Model	Benchmark	$K = 1$	$K = 8$	$K = 32$	$K = 128$	$\Delta_{1\to 8}$
GPT-5.6-sol	AppWorld	63.69	66.07	65.48	63.10	2.38
	WebArena	48.40	50.86	50.74	50.12	2.46
	GAIA	36.36	38.18	38.79	37.58	1.82
	τ-bench	56.52	59.13	59.13	58.26	2.61
Claude Sonnet 4.5	AppWorld	60.12	61.90	61.31	60.12	1.79
	WebArena	47.78	49.51	49.01	47.29	1.72
	GAIA	35.76	37.58	36.97	35.15	1.82
	τ-bench	57.39	59.13	58.26	56.52	1.74
DeepSeek-V4	AppWorld	58.33	60.12	59.52	57.74	1.79
	WebArena	44.21	45.81	45.07	43.84	1.60
	GAIA	29.09	30.91	29.70	28.48	1.82
	τ-bench	53.91	55.65	54.78	53.04	1.74
Kimi K3	AppWorld	52.38	54.17	52.38	51.79	1.79
	WebArena	38.79	40.39	39.53	38.18	1.60
	GAIA	27.88	29.09	27.88	26.67	1.21
	τ-bench	46.96	48.70	47.83	46.09	1.74

Table 15: Majority-vote success (%) across rollout budgets for GPT-5.6-sol on AppWorld.

Allocation	$B = 16$	$B = 32$	$B = 64$	$B = 128$
Same skill ($K = 1$)	61.90	62.50	63.10	63.69
Cross-skill ($K = 8$)	63.10	64.29	65.48	66.07

E Analytical Interpretation of Allocation

Task heterogeneity versus dependent executions. Persistent failures alone do not demonstrate that successive rollouts are dependent. Suppose each task has a fixed single-rollout success probability p_x under its selected skill, with executions independent conditional on the task. Then

$$C(N) = \mathbb{E}_x[1 - (1 - p_x)^N] \le 1 - (1 - \mathbb{E}_x[p_x])^N, \tag{14}$$

where the inequality follows from concavity in p_x for $N \ge 2$. Substituting average pass@1 into a homogeneous Bernoulli formula can therefore substantially overestimate coverage even without within-task dependence.

Oracle-best single-skill bound. Fix a task x and assume each rollout succeeds independently with probability $p_{x,s}$ when using skill s. Assume rollout costs are equal and success probabilities remain fixed. Let $p_x^\star = \max_s p_{x,s}$. For any nonnegative integer allocation with $\sum_s m_s = B$,

$$\prod_s (1 - p_{x,s})^{m_s} \ge \prod_s (1 - p_x^\star)^{m_s} = (1 - p_x^\star)^B. \tag{15}$$

Therefore

$$c_x(\mathbf{m}) \le 1 - (1 - p_x^\star)^B, \tag{16}$$

with equality attained by assigning all rollouts to a maximizing skill. Skills receiving zero executions contribute a factor of one. The bound concerns per-task coverage under known probabilities. It does not establish optimality for majority voting, varying execution costs, adaptive environments, or dependent rollouts. Nor does it imply that one globally chosen skill is optimal across tasks.

What the practical comparison measures. The observed pilot winner $s_x^\star$ in Equation (1) need not maximize the unknown $p_{x,s}$. Beating that winner can therefore be consistent with the bound. An empirical advantage alone does not identify ranking noise as its cause; the tie, composition, and held-out execution analyses are diagnostics rather than a causal proof. Replacing ranking with labels from the same evaluation trajectories would invalidate the comparison.

Table 16: AppWorld task recovery for GPT-5.6-sol at $B = 128$. Rescue, loss, and net gain use all tasks as the denominator and are percentage points.

K	Rescued	Lost	Net gain	Conditional rescue (%)
4	5.36	1.79	3.57	17.65
8	6.55	1.79	4.76	21.57
16	7.74	2.38	5.36	25.49
32	8.33	2.38	5.95	27.45

Table 17: Generation-attempt comparison for GPT-5.6-sol. The first three rows consume 1,152 generation attempts per task: the no-skill row votes over all attempts, while the skill and generated-strategy rows use 1,024 routing trials and 128 fresh evaluation trials. The two SD-BoN rows use 128 label-free rollouts. Values are selected success (%).

Condition	AppWorld	WebArena	GAIA	τ-bench
No skill, 1,152 rollouts	56.55	42.86	29.70	50.43
One skill, 1,152 total attempts	63.69	48.40	36.36	56.52
Generated strategies, 1,152 rollouts	58.93	45.69	31.52	52.17
SD-BoN (flat vote), 128 rollouts	65.48	50.62	37.58	58.26
SD-BoN + hierarchical vote	**66.67**	**50.99**	**38.79**	**60.00**
Pilot-ranked skills, 1,152 total	66.07	50.86	38.18	59.13

A selector-dependent allocation objective. Let $\mathcal{K}_B$ be the tested skill counts at budget B. The empirical oracle and selected-success maximizers are sets,

$$\mathcal{K}_C^\star = \arg\max_{K \in \mathcal{K}_B} C(K, B/K), \qquad \mathcal{K}_h^\star = \arg\max_{K \in \mathcal{K}_B} P_h(K, B/K), \tag{17}$$

where sets preserve ties rather than implying a precisely identified unique optimum. These test-set maximizers are descriptive: deployable allocation choices must be fixed on separate development tasks, and we infer neither a universal optimal skill count nor a validated adaptive allocation policy from the reported sweep.

Recovery statistics. For nested candidates, the event of first-rollout success is a subset of the event of success by rollout N. Hence $C(N) - C(1) = \Pr(\text{first fails, a later candidate succeeds})$, giving Equation (3) after conditioning. In Table 9, equal weighting over model-level coverage estimates defines the averaging distribution before taking the ratio. This is not the unweighted average of per-model recovery ratios. The interpretation does not apply unchanged if $C(1)$ and $C(N)$ use different tasks, selected skills, or incompatible sampling estimators.

E.1 Illustration: Voting Can Benefit from a Weaker Skill

In a simplified task with one canonical correct answer, suppose skill A assigns probability 0.40 to that answer and 0.60 to wrong answer A. Skill B assigns 0.35 to the correct answer and 0.65 to a different wrong answer B. Equal mixing yields shares 0.375, 0.30, and 0.325: the correct answer becomes the unique mode despite skill B's lower individual accuracy. For fixed skill sets and independent draws, empirical vote shares converge to their mixture probabilities as repetitions increase. More sampling can therefore reinforce a wrong mode or stabilize a correct one. This example is mathematical, not an observed benchmark result; state-changing tasks require the outcome-equivalence rules in Appendix C.

F Additional Diagnostics

Scope of the repeat comparisons. This appendix reports contrasts across prompt construction and execution runs. These sources of variation are kept separate. The length-matched comparison varies both factors; the allocation comparison holds prompt construction fixed and varies execution runs.

Table 18: AppWorld composition control for GPT-5.6-sol at $K = 8$, $M = 16$ on the same 143-task matched subset. The first two conditions have pilot-score profiles matched within the stated tolerance; the Top-8 reference is recomputed on this subset.

Composition	Pilot mean	d_x	Coverage (%)	Selected success (%)
Matched, lower diversity	0.734	0.218	72.73	65.73
Matched, higher diversity	0.731	0.462	74.13	67.13
Pilot Top-8 reference	0.769	0.384	74.83	66.43

Table 19: Sensitivity to pilot-ranking evidence for GPT-5.6-sol on AppWorld. ΔC and ΔP compare $K = 8$ with $K = 1$.

R	Screening rollouts	Top-score tie rate	ΔC (pp)	ΔP (pp)
1	128	71.4%	6.19	3.39
2	256	58.9%	5.65	3.18
4	512	43.2%	5.00	2.80
8	1,024	31.5%	4.76	2.38

Prompt construction and repeat design. The initial controlled setting is AppWorld with GPT-5.6-sol. The transfer checks use WebArena with GPT-5.6-sol and AppWorld with DeepSeek-V4. For each setting, we use four independently constructed prompt/strategy sets and three independent execution runs per construction. Each mean gives equal weight to the $4 \times 3 = 12$ construction–run combinations and to tasks within a setting. Multiple executions of one strategy set cannot measure construction variation. Frozen task-only retrieval is distinct from a pilot-ranked comparison, which is not used for the content tests. Table 23 reports the resulting averages: the distinct-skills direction is stable in coverage across settings, while the selected-success margin remains small.

Allocation stability. The three-run AppWorld comparison uses Similarity Top-1 and Similarity Top-8, rather than the MMR selector in the main table. A mean gain of 1.98 points corresponds to ten additional successes across $3 \times 168 = 504$ task–run cases. This is a different selector comparison from the SD-BoN result. For a separate $K = 8 \to 32$ comparison, changes of three covered tasks and three fewer selected successes across 504 cases imply $(\Delta C, \Delta P, \Delta G) = (0.60, -0.60, 1.19)$ points. This comparison fixes the prompt set. Pilot-ranked repeats also fix the ranking, separating execution variation from a full rerun of screening.

Aggregation and task counts. A binary paired difference equals the number of newly successful tasks minus newly failed tasks, divided by the task count. Repeated-condition differences are computed within a task and construction–run combination before averaging. A task appearing in 12 combinations is not counted as 12 independent tasks for statistical inference. The reported means are descriptive; their positive signs alone do not establish significance.

Auditing the content intervention. A format-only rewrite must preserve instructions, including entity and tool references, without adding task-specific information. A content abstraction must remove the specified operational details while retaining the original goal and high-level subgoals. Across the 5,376 audited source–representation pairs, the instruction-preservation rate for format-only rewrites is 0.94, blind audit agreement and the proportion of rewrites needing correction are 0.86 and 0.12, and 0.68 of sources contain applicable operational details. Sources with no removable detail remain in the full-set analysis and are tracked for a separate eligible-source analysis rather than silently excluded; a null contrast is therefore not interpreted as evidence against procedural guidance when many source skills lacked that guidance initially. Rewrites are not selected by benchmark success. If supporting scripts differ between conditions, the comparison estimates a broader capability-package effect and cannot isolate textual procedural content.

Selector diagnostics. The mean number of successful groups per covered task is 1.31, 1.58, and 1.74 at $K = 1, 8, 32$, respectively, while the largest successful group contains fractions 0.71, 0.63, and 0.57 of successful candidates. Fractions 0.07, 0.11, and 0.15 of vote failures come from select-

Table 20: AppWorld voting diagnostics for GPT-5.6-sol at $B = 128$. C, P_{vote}, and G_{vote} are percentages.

K	C	P_{vote}	G_{vote}	Wrong-mode mass	Mixed-group mass
1	69.64	63.69	5.95	42.61	1.72
8	74.40	66.07	8.33	31.84	2.36
32	75.60	65.48	10.12	27.52	3.41
128	74.40	63.10	11.31	24.18	4.89

Table 21: Cross-skill selection summary at $B = 128$. Entries are percentages. The majority-vote, verifier, and oracle columns use the same candidate pools within each model and benchmark.

Model	Benchmark	Majority		Verifier		Oracle coverage	
		$K = 1$	$K = 8$	$K = 1$	$K = 8$	$K = 1$	$K = 8$
GPT-5.6-sol	AppWorld	63.69	66.07	66.07	69.64	69.64	74.40
	WebArena	48.40	50.86	51.23	53.82	54.06	58.37
	GAIA	36.36	38.18	38.79	41.21	40.61	44.85
	τ-bench	56.52	59.13	60.00	62.61	62.61	67.83
Claude Sonnet 4.5	AppWorld	60.12	61.90	63.10	65.48	64.88	69.64
	WebArena	47.78	49.51	50.37	52.83	52.22	55.79
	GAIA	35.76	37.58	38.18	41.21	40.00	44.24
	τ-bench	57.39	59.13	60.00	63.48	63.48	67.83
DeepSeek-V4	AppWorld	58.33	60.12	61.31	64.29	62.50	67.26
	WebArena	44.21	45.81	46.92	49.75	49.38	53.45
	GAIA	29.09	30.91	32.12	35.15	33.33	37.58
	τ-bench	53.91	55.65	56.52	60.00	60.00	64.35
Kimi K3	AppWorld	52.38	54.17	55.36	58.33	56.55	61.90
	WebArena	38.79	40.39	41.50	44.46	43.23	46.67
	GAIA	27.88	29.09	30.30	33.94	32.73	36.97
	τ-bench	46.96	48.70	49.57	53.04	52.17	56.52

ing a failing representative of a mixed group. These aggregates describe group fragmentation and representative selection separately; neither alone identifies the cause of an allocation curve.

G Reproducibility and Research Assistance

Reproducing the full comparison requires source and experiment manifests, immutable content hashes, audit records, cleaning diffs, retrieval indices, pilot and evaluation split identifiers, per-task trajectories, selector outputs, and aggregation scripts. Redistribution of skill bodies depends on their licenses; provenance metadata and hashes can be supplied separately when redistribution is restricted.

The local corpus manifests and audit artifacts record collection and cleaning. For benchmark experiments, candidate identities and reuse must be traceable across every table and figure. Aggregate arithmetic and cross-table consistency checks do not replace task-level validation or establish statistical significance.

LLM assistance was used in corpus auditing and manuscript editing. Numerical reporting requires independent checks of task denominators, selection identities, rounding, and the provenance of any claimed experimental result.

Table 22: AppWorld selection bottleneck across models at $B = 128$. Values are percentages; $G = C - P_{\text{vote}}$.

	Majority success		Oracle coverage		Gap G	
Model	$K = 8$	$K = 32$	$K = 8$	$K = 32$	$K = 8$	$K = 32$
GPT-5.6-sol	66.07	65.48	74.40	75.60	8.33	10.12
Claude Sonnet 4.5	61.90	61.31	69.64	70.24	7.74	8.93
DeepSeek-V4	60.12	59.52	67.26	67.26	7.14	7.74
Kimi K3	54.17	52.38	61.90	63.69	7.74	11.31

Table 23: Repeat-study averages for the length-matched distinct-skills versus generated-strategies comparison. Gains are percentage points; means average four prompt constructions and three execution runs per construction.

Benchmark	Model	ΔC	ΔP
AppWorld	GPT-5.6-sol	2.98	0.60
WebArena	GPT-5.6-sol	2.96	1.48
AppWorld	DeepSeek-V4	2.38	1.19