Decomposing Adaptive LLM Team Gains with a Frozen-Team Counterfactual

Anonymous authors
Paper under double-blind review

Abstract

Adaptive LLM teams combine two interventions: a fixed collaboration policy and a search procedure that adapts it. Standard evolved-versus-single comparisons conflate their contributions. We introduce a three-arm evaluation that measures *Team Lift* (frozen team minus single-agent policy) and *Evolution Lift* (evolved team minus the same frozen starting team) under a common allocated token budget. On a static SWE-derived patch-output proxy—scored from generated patch text without running repository tests—the evolved team exceeds the single agent by 6.875 percentage points across the observed runs. Arithmetically, this total consists of 5.350 points of Team Lift and 1.525 points of Evolution Lift; on GPQA, the corresponding decomposition is $3.0 = 3.8 - 0.8$ points. Exploratory controls show that fixed policies with calibrated or deterministic selection can reach or exceed the evolved policy's proxy score, while cold-start traces motivate broader proposal coverage. Rerun-aware and execution-based analyses distinguish uncertainty over search trajectories from validity of the evaluation endpoint. These results motivate a practical standard for collaboration search: report whether adaptation adds repeatable, task-valid value beyond a strong frozen policy instead of attributing the full team gain to evolution.

1 Introduction

Adaptive LLM teams are increasingly designed as searchable systems: roles, communication links, aggregation rules, and prompts are updated from execution experience (Hao et al., 2026; Hu et al., 2025b; Lu et al., 2024). Their usual success criterion is simple—the final evolved team should outperform a single agent. Yet that comparison changes two things at once. It introduces a multi-role collaboration scaffold and then adapts the scaffold. It may also redistribute a fixed budget across search, critique, refinement, and final execution. Consequently, an evolved-system gain is a composite quantity rather than direct evidence that structure search added value.

We make this composition explicit. For the complete arm policies evaluated in this paper,

$$\underbrace{\mu_{\text{evolve}} - \mu_{\text{single}}}_{\text{total evolved-system lift}} = \underbrace{\mu_{\text{frozen}} - \mu_{\text{single}}}_{\text{Team Lift}} + \underbrace{\mu_{\text{evolve}} - \mu_{\text{frozen}}}_{\text{Evolution Lift}}. \tag{1}$$

The missing counterfactual is the frozen team: it shares the evolved arm's starting collaboration structure but does not search over structures. Team Lift therefore measures the operational difference between the specified frozen-team and single-agent policies, while Evolution Lift measures the incremental difference made by the tested search policy. These are policy contrasts, not a mechanism-pure causal partition, but they answer the evaluation question that an evolved-versus-single comparison leaves open.

We instantiate this comparison with SwarmEvolve, a controlled testbed that represents a team as prompted roles, typed communication channels, and an aggregation procedure. A failure-guided search proposes guarded structural mutations on development tasks. The evolved team, its frozen starting team, and a self-improving single agent receive the same tasks, backbone, tools, and per-arm allocated token budget; adaptation and scored execution

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.

draw from one ledger, and held-out outcomes never enter search. The design turns a common headline comparison into two separately interpretable targets while retaining a concrete adaptive-team system on which to study search behavior.

Our contributions are:

- **A two-lift evaluation framework.** We formalize total evolved-system gain as Team Lift plus Evolution Lift and identify the frozen starting team as the essential counterfactual for crediting structure search.
- **A controlled three-arm testbed.** We compare structure evolution, the same team held fixed, and single-agent self-improvement under aligned tasks, development/test separation, a common backbone and evaluator, and equal allocated-token accounting.
- **A diagnostic empirical study of adaptive-team gains.** Across SWE and GPQA, the decompositions show that total team lift and incremental search lift can differ materially. On the SWE proxy, the +6.875-point total decomposes into +5.350 points of Team Lift and +1.525 points of Evolution Lift. Exploratory fixed policies attain comparable or higher cell means, while cold-start traces motivate proposal coverage as a hypothesis for improving collaboration search.

The resulting picture is constructive. A team can deliver useful operational lift, but search earns separate credit only for the increment beyond a strong fixed policy. Reporting both lifts changes which systems count as progress, which baselines are necessary, and where adaptive-team research should invest: in repeatable search gains, proposal mechanisms that can invent missing structure, and evaluators that measure the intended task outcome.

2 Related Work

2.1 Search over Agent Teams

Recent systems optimize prompts, roles, workflows, and communication structures from execution experience. EvoMAC (Hu et al., 2025b) updates prompts and workflow connections through textual backpropagation; MorphAgent (Lu et al., 2024) adapts decentralized profiles, roles, and capabilities; and AgentEvolver (Zhai et al., 2025) studies self-questioning, navigation, and attribution. Meta-Team (Hao et al., 2026) searches across individual behavior, coordination, and team organization, while Evo-Bench (Huang et al., 2026) evaluates whether LLMs can improve agent harnesses. AFlow (Zhang et al., 2025), ADAS (Hu et al., 2025a), G-Designer (Zhang et al., 2024), and Multi-Agent Design (Zhou et al., 2026) likewise search program- or graph-defined agent systems. These methods establish collaboration search as a meaningful design problem. Our focus is its evaluation: whether improvement belongs to the searched policy or to the fixed scaffold and refinement machinery that the final system also contains.

2.2 Controls for Multi-Agent Gains

Multi-agent systems can gain from specialization, critique, parallel sampling, and aggregation, but can also fail through system specification, inter-agent alignment, and verification (Cemri et al., 2026). Under matched thinking-token budgets, single-agent systems may match or outperform multi-agent systems on multi-hop reasoning (Tran & Kiela, 2026). A strong single-agent control is therefore necessary, but it answers whether the *complete team policy* helps. A frozen-team control answers the complementary question of whether adaptation improves on the same starting scaffold. Our three-arm design combines both controls and adds rerun-aware analysis for the stochastic search procedure.

2.3 Benchmarking Adaptive Agent Systems

SWE-bench Verified (Jimenez et al., 2024; OpenAI, 2024) and GPQA-Diamond (Rein et al., 2023) provide contrasting settings for adaptive-agent evaluation: structured generation with

Contrast	Reported quantity	Supported policy-level interpretation
evolve−single	Total adaptive-team lift	Value of the complete adaptive-team policy relative to the single-agent policy.
frozen−single	Team Lift	Value of the specified fixed-team policy relative to the single-agent policy.
evolve−frozen	Evolution Lift	Incremental value of the tested search policy from the same starting team.

Table 1: Counterfactual ladder for adaptive-team evaluation. Each contrast supports a distinct policy-level interpretation; reporting all three prevents total team lift from being mistaken for search lift.

repository context and exact-choice reasoning. We use them to test whether separating total team lift from incremental search lift remains informative across different output and scoring structures.

3 Three-Arm Evaluation Design

3.1 Identification Target

Let μ_{arm} denote performance over an aligned task set after averaging the observed runs of a complete arm policy. We evaluate three arms: (i) an *evolve* policy that searches over collaboration structures, (ii) a *frozen* policy that retains the evolve arm's initial team, and (iii) a *single* policy that performs prompt self-improvement without a team. The conventional evolve–single contrast decomposes as

$$\Delta_{e-s} = \underbrace{\mu_{\text{frozen}} - \mu_{\text{single}}}_{\Delta_T \text{ (Team Lift)}} + \underbrace{\mu_{\text{evolve}} - \mu_{\text{frozen}}}_{\Delta_E \text{ (Evolution Lift)}}. \tag{2}$$

We formally test total evolved-system lift Δ_{e-s} and incremental search lift $\Delta_{e-f} = \Delta_E$; Δ_T is reported descriptively. Because each arm includes its own refinement policy, the terms are operational policy contrasts rather than mechanism-pure causal effects. Their value is comparative: the frozen arm holds the starting scaffold fixed while removing structure search. Table 1 makes the resulting claim ladder explicit: each stronger claim requires a stronger counterfactual.

3.2 SwarmEvolve Testbed

We study adaptation of an explicit collaboration structure

$$G = (R, C, \Pi), \qquad R = \{r_i = (\mathit{name}, \mathit{duty}, \pi_i)\}, \quad C \subseteq R \times R \times T, \tag{3}$$

where R is a roster of prompted roles, C contains directed channels with types $T = \{\texttt{delegate}, \texttt{review}, \texttt{broadcast}, \texttt{merge}\}$, and Π specifies decomposition, aggregation, and a round cap. Executing backbone B with G returns an answer, a message trajectory, and measured token cost. The reported system evolves prompts and orchestration, not model weights, and uses no external tools.

The takeover initialization G_0 is a planner $\rightarrow$ implementer $\leftrightarrow$ reviewer pipeline with the implementer as aggregator and two interaction rounds. The cold-start initialization contains one role and no channels. Structures must reference existing roles, contain no duplicate channels, satisfy $|R| \leq 20$, and use $1 \leq n_{\max} \leq 6$.

For each adaptation round, SwarmEvolve executes the current structure on a fixed four-task development subset and extracts rule-based failure signals. A signal selects one applicable mutation: add, remove, merge, or specialize a role; add, remove, or retype a channel; or alter aggregation or the round cap. Candidates are pruned and checked for valid task-specific output extraction. The candidate is accepted when

$$\widehat{u}_{\text{dev}}(G') \geq \widehat{u}_{\text{dev}}(G) - \epsilon, \qquad \epsilon = 0.02. \tag{4}$$

The random-mutation control retains the representation, development executions, budget, and structural guards but samples valid operators without failure conditioning and accepts them unconditionally. The single-agent arm updates only its prompt; self-improvement notes are capped at 800 characters and use the same non-degradation rule. Full operator definitions and guards appear in the Appendix.

3.3 Allocated-Budget Comparison

For n test tasks and per-task allocation T_{task}, every arm receives $T_{\text{total}} = nT_{\text{task}}$. Adaptation and test execution share one ledger, $T = T_{\text{adapt}} + T_{\text{exec}}$, with at most 35% assigned to adaptation. The frozen arm retains G_0 and uses residual budget for refinement passes with prior role outputs as context; the single arm uses prompt self-improvement and refreshed execution. We match allocated tokens across arms and report realized consumption separately. Call-granularity accounting permits at most one final-call overshoot; the Appendix gives the full refresh and ledger rules.

3.4 Evaluation Protocol and Inference

The evaluation uses two task families, 50 aligned held-out tasks per cell, two budget tiers (60k and 150k tokens per task), and at least three stochastic seeds. The task families, three-arm design, and two contrasts preceded held-out access; the equal-weight low/high aggregation and four-hypothesis familywise analysis were finalized after initial inspection and are reported as post hoc.

Cell estimates average seeds within task and use 10,000 paired percentile bootstrap resamples. For familywise inference, aligned task effects are computed after seed averaging, low and high tiers receive equal weight, and two-sided p-values use a null-centered paired bootstrap with the +1 correction followed by Holm step-down (Holm, 1979). A separate task×seed crossed bootstrap resamples paired run clusters to examine generalization to a fresh search trajectory. Invalid or missing answers score zero; only recorded infrastructure failures and protocol violations are excluded. The Appendix reports exclusions, protocol changes, realized-token sensitivity, and all cell-level estimates.

4 Experiments

4.1 Experimental Setup

Tasks. We use disjoint 20-example development and 50-example test subsets. The code family is sampled from SWE-bench Verified (Jimenez et al., 2024; OpenAI, 2024). Its primary endpoint is a static SWE-derived patch-output proxy, scored without running repository tests: a patch block containing the reference-derived target cue scores 1, a syntactically present patch without that cue scores 0.5, and an absent patch scores 0. GPQA-Diamond (Rein et al., 2023) uses deterministic extraction and exact matching of the final A–D choice.

Backbones and runs. Primary runs use the non-public endpoint alias `deepseek-v4-pro`; exact weights and revision are unavailable outside the execution environment. Robustness runs use reproducible Qwen3-8B (Yang et al., 2025) served with vLLM (Kwon et al., 2023). The three arms share task splits, executor, per-call 8k-token cap, and allocated total budget. Main cells contain five seeds except SWE-high, which contains four. Realized use and all sensitivity analyses appear in the Appendix.

4.2 Decomposing the Observed Team Gain

Figure 4 reports the two formal contrasts. On the SWE-derived proxy, the equal-weight evolve–single estimate is +6.875 percentage points (95% CI $[+2.575, +11.675]$; raw $p = 0.00380$; Holm-adjusted $p = 0.0152$). GPQA yields +3.0 points ($[-0.4, +6.8]$; adjusted $p = 0.289$).

Table 2 exposes the composition hidden by those total contrasts. On SWE, evolve–single is +6.875 points, while frozen–single is +5.350 points and evolve–frozen is +1.525 points; the

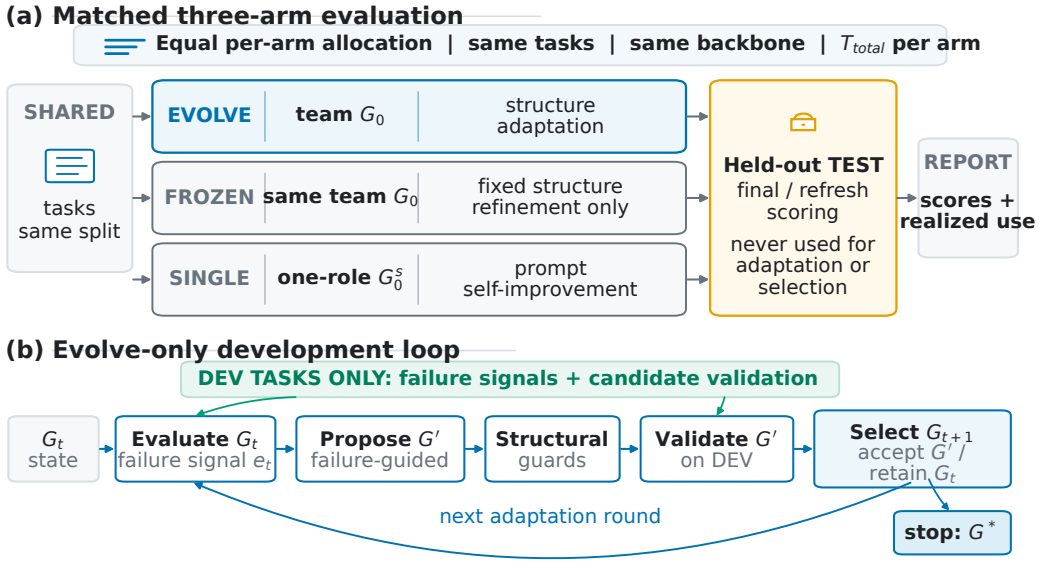

Figure 1: Fair-comparison design and the SwarmEvolve adaptation loop. (a) Evolve and frozen start from the same team structure G_0, whereas the single-agent arm starts from a one-role system. Each arm receives the same tasks, backbone, and per-arm allocated budget T_{total}, and is scored on the same held-out test tasks using the same evaluator. Test outcomes do not enter adaptation or selection. (b) On development tasks only, SwarmEvolve extracts failure signals, proposes and guards a candidate structure, validates it, and either accepts the candidate or retains the current structure; the selected state enters the next round until termination yields G^*.

Task family	Total lift	Team lift	Evolution lift
SWE output proxy	$+6.875$ $[+2.575, +11.675]$	$+5.350$	$+1.525$ $[-2.550, +5.775]$
GPQA	$+3.000$ $[-0.400, +6.800]$	$+3.800$	-0.800 $[-3.600, +1.800]$

Table 2: Equal-weight low/high point-estimate decomposition in percentage points. Bracketed intervals are per-comparison, unadjusted 95% percentile intervals for the two formally analyzed contrasts; Team Lift is the descriptive arithmetic remainder.

frozen–single contrast is therefore 78% as large as the total-lift point estimate. On GPQA, the corresponding decomposition is $+3.0 = +3.8 - 0.8$ points. The empirical interpretation changes with the contrast: a positive adaptive-team lift does not by itself quantify the incremental value of search. The task-conditional interval is positive for SWE total lift, whereas the Evolution Lift intervals include zero in both task families; the post-hoc and rerun-aware scope of these intervals is specified in Section 3.4 and the Limitations.

The aggregate values conceal a useful pattern across budget cells. As Table 3 shows, Team Lift is positive in all four cells, whereas Evolution Lift changes sign. At SWE-low the two components are similar ($+3.2$ and $+3.8$ points); at SWE-high, frozen–single is $+7.5$ points whereas evolve–frozen is -0.75 points. GPQA shows the same separation in a different form: the frozen team is above single at both budgets, while search moves the score down at low budget and up at high budget. These cell means are descriptive, but they reveal why a single evolve–single average cannot diagnose whether collaboration search is working.

Fresh-search and task-validity checks. Task-conditional inference treats the observed search runs as fixed. When the bootstrap also resamples adaptation runs, the SWE evolve–single interval widens to $[-0.7, +14.9]$ points, and both evolve–frozen intervals include zero (Appendix Table 7). Separately, an exploratory chroot-based audit executes fail-to-pass

Task	Budget	Total lift	Team lift	Evolution lift
SWE proxy	low	+7.00	+3.20	+3.80
	high	+6.75	+7.50	−0.75
GPQA	low	+2.00	+5.20	−3.20
	high	+4.00	+2.40	+1.60

Table 3: Descriptive point-estimate decomposition by task family and allocated budget tier, in percentage points. No cell-level multiplicity-adjusted claim is attached to these values.

System	Main mechanism	SWE-low	SWE-high	High tokens/task
single	self-improvement	.188	.173	140k
frozen	fixed team	.220	.247	156k
evolve	team + search	.258	.240	151k
B1	self-evaluated sampling	.187	.213	69k
B2	critique–revision	.237	.240	77k
B3	dev-calibrated selection	.317	.300	171k[a]
B4	sequential refinement	.190	.210	60k
B5	parallel samples + pick	.327	.273	59k
B6	generic team scaffold	.240	.243	88k

Table 4: Exploratory fixed-policy decomposition on the SWE output proxy ($n = 3$ per B1–B6 condition). [a]B3-high realizes one permitted final-call overrun, about 14% above its 150k allocation. These are diagnostic comparisons, not confirmatory rankings.

tests for 48 of 50 SWE-high tasks and 1,008 evaluations. Among outputs assigned proxy score 1.0, 9.9% pass; evolve and single execution rates are 3.1% and 2.6%, with six versus five discordant successes (McNemar $p = 1.0$). These checks answer different questions: the former quantifies fresh-run uncertainty, while the latter calibrates the SWE-high proxy endpoint rather than re-estimating the low/high family contrast as software-repair performance.

4.3 Which Fixed Policies Reach the Adaptive-Team Scores?

The three-arm comparison identifies the missing counterfactual; stronger fixed policies help diagnose what the total contrast may contain. We therefore ran six exploratory, allocated-budget-matched SWE controls with three seeds each (Table 4; full definitions and realized costs in Appendix A.7). Several fixed policies attain comparable or higher cell means. A critique–revision pair (B2) matches evolve at the high tier (.240); a development-calibrated selector (B3) exceeds evolve at both tiers (.317/.300 versus .258/.240); and a non-communicating pseudo-team with deterministic selection (B5) does likewise (.327/.273). A generic-prompt scaffold (B6) is close to the frozen reference, whereas self-evaluated best-of-N (B1) remains near the single reference. These comparisons make selection quality and fixed collaboration policies necessary controls when attributing a score gain to structure search.

The realized-cost view reinforces this diagnosis. Figure 2 plots the reference arms and B1–B6 against tokens actually consumed per task. At the low tier, B5 attains .327 at about 61.7k tokens per task, compared with evolve at .258 and about 63.3k. At the high tier, B5 reaches .273 at about 59k, while B3 obtains the highest observed proxy score (.300) at about 171k after its permitted final-call overshoot. The point is not that one exploratory baseline is universally best. Rather, the score region attributed to an adaptive team is reachable by qualitatively different fixed policies, and some of those policies trade score for realized cost differently. A search method should therefore be compared not only with a weak single policy but with the fixed-policy frontier relevant to its claimed mechanism.

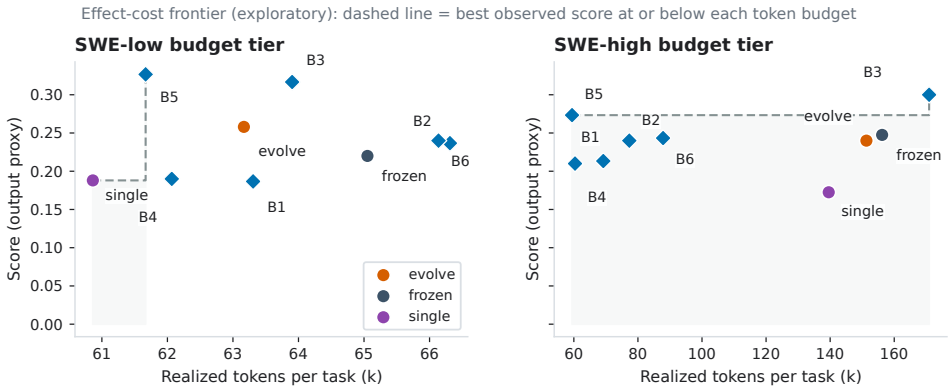

Figure 2: Exploratory score–cost view on the SWE output proxy. Circles are the seed-averaged reference arms and diamonds are B1–B6, plotted against realized tokens per task. The dashed envelope is the best observed score at or below each cost; it is a diagnostic frontier, not a statistical ranking.

4.4 A Proposal-Coverage Hypothesis for Cold-Start Search

Cold-start traces reveal a concrete search-design challenge. Guided mutation scores below random mutation in all three SWE seeds. Of 30 guided proposals, 26 change aggregation and four repair output formatting; none introduce a missing role. Random search spans six operator types and, in one trace, adds a reviewer that doubles the development score from .25 to .50. A small proposal×acceptance factorial is consistent with this pattern: the random-minus-guided proposal estimate is +10.0 points (bootstrap 95% CI $[+3.2, +17.2]$; permutation $p = .055$), whereas the acceptance estimate is −3.3 points ($p = .56$). With only three seeds per cell and a post-inspection completion of the factorial, this is a design hypothesis rather than a general mechanism claim. These exploratory runs motivate a proposal-coverage hypothesis: when failure guidance can name only existing components, search may repeatedly edit aggregation while under-proposing missing roles or channels. Proposal mixtures that add explicit structure-level exploration are therefore a concrete design direction for future evaluation.

Figure 3 separates this cold-start failure from the broader question of whether guidance is useful. Guided proposals trail random proposals in all three SWE cold-start seeds (mean −13.3 points), tie on average for GPQA cold start, and show mixed signs from the richer takeover initialization. The dependence on initialization is informative: trajectory-local guidance has more structural vocabulary when useful roles and channels already exist. From a one-role seed, the same localization mechanism can repeatedly optimize the only visible component instead of expanding the collaboration graph. This motivates proposal mixtures that combine failure-conditioned exploitation with explicit structure-level exploration.

Dependence on initialization and backbone. Within the tested grid, arm ordering and selected structures change with initialization, task family, and backbone. Takeover initialization benefits more from guided proposals than cold start, and Qwen3-8B does not preserve the primary-backbone ordering. These boundary results strengthen the case for reporting both lifts per setting rather than treating one evolved-system number as a property of collaboration search in general.

5 Discussion

The decomposition suggests design and reporting principles for adaptive LLM teams.

Evaluate the increment, not only the assembled system. An evolved team packages a starting scaffold, refinement policy, sampling and aggregation choices, and a search

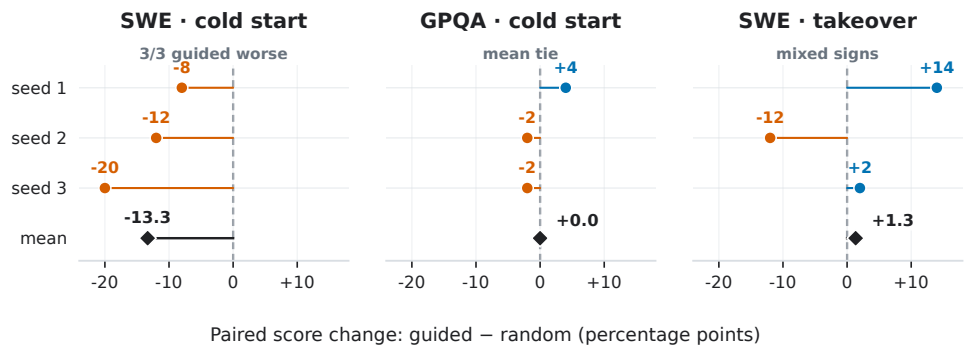

Figure 3: Paired guided-minus-random score changes by seed in exploratory high- budget runs. Guidance is consistently lower from the SWE one-role cold start, but not uniformly lower across tasks and initializations. The pattern motivates an initialization-dependent proposal-coverage hypothesis rather than a general claim that random mutation is superior.

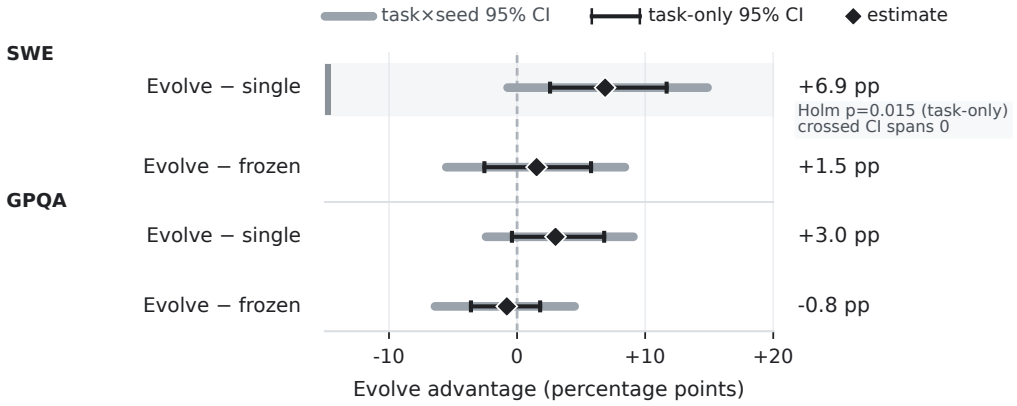

Figure 4: Equal-weight low/high contrasts for the four-test family: evolve−single and evolve−frozen per task family. Dark bars: task-conditional per-comparison 95% percentile intervals; light bars: task×seed rerun-aware intervals (10,000 resamples); diamonds: point estimates; gray band: the single task-conditional rejection. SWE metric: derived output proxy; unique adaptive runs 5 low / 4 high (GPQA 5/5).

procedure. Evolve–single measures the value of that package. Evolve–frozen measures the difference between the tested search policy and the frozen policy from the same starting team. Both are useful, but they support different claims. Future comparisons should therefore report a ladder of increasingly strong counterfactuals: single-agent self-improvement, a frozen starting team, and fixed critique/sampling/selection policies before attributing residual gain to adaptation.

Search may benefit from broader structural reachability. The cold-start traces illustrate why local failure guidance can be informative yet incomplete. A signal attached to an existing component naturally proposes repairs to that component; it may never invent the reviewer, verifier, or channel whose absence caused the failure. These traces suggest that collaboration search may benefit from proposal coverage in addition to an acceptance rule. Structure-agnostic role invention, counterfactual channel proposals, or proposal mixtures with explicit exploration are promising ways to expand that reachable set. Our small factorial motivates these interventions; it does not yet rank them.

Use evidence matched to the deployment claim. Task-conditional intervals describe the observed search runs, whereas rerun-aware intervals address a newly sampled trajectory. Allocated-budget matching controls opportunity, whereas realized tokens describe actual

cost. Likewise, a static patch-output proxy measures format and target cues, whereas fail-to-pass execution measures repair correctness. A convincing deployment case should align all three layers: the evolved policy should exceed a strong fixed policy, the increment should survive fresh searches, and the evaluator should measure the outcome that deployment values. This standard turns the two-lift framework from a reporting convention into a practical decision rule for whether adaptation is worth its added search and operational complexity.

A minimal reporting protocol. Specify the complete policies being compared; report evolve–single, frozen–single, and evolve–frozen on aligned tasks; separate held-out-task uncertainty from search-rerun uncertainty; and validate the endpoint at the level required by the deployment claim. This compact checklist prevents scaffold, selection, budget, and evaluator effects from collapsing into one "evolution" number.

Optimize for repeatable incremental value. The relevant objective is the increment over the strongest fixed policy at comparable opportunity and cost, together with its probability of recurring after a fresh search. Proposal policies should be judged by the useful structures they make reachable, acceptance policies by whether development data distinguish those proposals, and the complete search by its distribution of Evolution Lift across reruns. A strong frozen team is therefore the platform from which meaningful adaptation begins.

6 Conclusion

Adaptive-team gains are composite. By introducing a frozen-team counterfactual, we separate the operational value of using a team from the incremental value of searching over that team. In our testbed, the +6.875-point SWE output-proxy estimate decomposes into +5.350 points of Team Lift and +1.525 points of Evolution Lift. In exploratory runs, fixed policies with calibrated or deterministic selection reach or exceed the evolved policy's proxy score, while cold-start traces motivate proposal coverage as a concrete search-design hypothesis. The broader lesson is a positive evaluation standard: collaboration evolution should be credited when it adds repeatable, task-valid value beyond a strong frozen policy. Reporting Team Lift and Evolution Lift separately makes that standard measurable.

Limitations

Metric validity. The primary SWE measure is a non-execution proxy for patch formatting, target-file cues, and reference-derived keywords, not repair correctness. The exploratory fail-to-pass audit covers only the executable SWE-high subset and therefore does not convert the primary family analysis into an execution-based evaluation. GPQA uses exact letter matching; when `FINAL:` is missing, the parser takes the first standalone A–D token, affecting 40.6% of single-arm records. Both evaluators remain imperfect measurements of the intended task outcomes.

Statistical scope. The estimands were specified before test access, but the final low/high aggregation and familywise bootstrap were finalized after initial result inspection; familywise inference is therefore post hoc. With 3–5 seeds per condition, rerun-aware intervals characterize task/run sensitivity rather than establish absence of an effect. The three-seed proposal×acceptance factorial is hypothesis-generating. B1–B6 match allocated—not necessarily realized—compute and do not exhaust longer refinement chains, larger candidate pools, or additional selectors.

External validity. The evidence spans two task families, two backbones, two budget tiers, and a tool-free harness. Cross-backbone changes in arm ordering do not identify capability as the moderator. Results may differ with execution-enabled environments, tools, other models, richer teams, or alternative search and credit assignment. Our conclusion is specific: adaptive-team evaluations should separate Team Lift from Evolution Lift.

Reproducibility Statement

Reproducibility statement. All experiment configurations, per-run token ledgers, per-task scores, mutation trajectories including rejected candidates, and the analysis scripts used for the reported results are included in an anonymous supplementary package, submitted with the paper where available; the paper itself is self-contained. Sampling is deterministic, with fixed seeds and manifests containing SHA-256 hashes of the source data files. The prespecified data-collection protocol, post-hoc analysis boundary, and run-exclusion criteria are documented in the Appendix. The framework depends only on PyYAML and supports OpenAI-compatible endpoints. Exact reruns of the primary-backbone cells require access to the original `deepseek-v4-pro` endpoint. The Qwen3-8B robustness cells provide an independently reproducible instantiation using the released configurations and require one 80 GB GPU with vLLM 0.8.5.post1, PyTorch 2.6.0, Transformers 4.51.3, and Tokenizers 0.21.1.

Ethics Statement

Ethics statement. The study uses SWE-bench Verified and GPQA-Diamond under their applicable access and license terms. It involves no human subjects and no personal or sensitive data. Released artifacts exclude restricted benchmark content and are distributed under the applicable data and model licenses.

Broader impact. The results caution against attributing an evolved team's advantage over a single agent to evolution without a fixed-team control. If replicated with execution-based repair metrics, this distinction could help practitioners avoid search and coordination overhead when structural adaptation adds no repeatable value, reducing financial and energy costs. Observed variation across tasks, backbones, and initial structures also argues for validating incremental evolution lift in the intended deployment setting rather than assuming that collaboration gains transfer.

AI Use Statement

Generative AI tools assisted with refining the conceptual framing and research hypotheses; providing feedback on experimental design and methodology; implementing, refactoring, and testing code; scheduling and monitoring experiments; analyzing execution traces and interpreting results; creating and revising figures and tables from logged artifacts; searching and summarizing literature; packaging reproducibility artifacts; and drafting, translating, and editing manuscript text. We did not use generative AI to create synthetic datasets or to formulate or prove mathematical theorems; tasks involving human subjects or qualitative human data were not applicable. The authors reviewed all AI-assisted work through code tests, source and citation checks, scripted recomputation of reported quantities, cross-checks against frozen result artifacts, and visual inspection of figures and the compiled manuscript. The authors take full responsibility for the final text, claims, and artifacts, including material produced with generative-AI assistance.

References

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *Advances in Neural Information Processing Systems*, 38, 2026.

Zhezheng Hao, Tianfu Wang, Huanshuo Dong, Ziyan Liu, Hong Wang, Xiankun Lin, Qiang Lin, Can Wang, Hande Dong, and Jiawei Chen. Evolve as a team: Collaborative self-evolution for llm-based multi-agent systems. *arXiv preprint arXiv:2605.29790*, 2026.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025a.

Yue Hu, Yuzhu Cai, Yaxin Du, Xinyu Zhu, Xiangrui Liu, Zijie Yu, Yuchen Hou, Shuo Tang, and Siheng Chen. Self-evolving multi-agent collaboration networks for software development. In *International Conference on Learning Representations*, volume 2025, pp. 23007–23039, 2025b.

Lisheng Huang, Chen Yang, Hao Zhou, Huatong Song, Zongchao Chen, Ran Le, Yang Song, Wayne Xin Zhao, and Tao Zhang. Evo-bench: Can language models improve agent harness? *arXiv preprint arXiv:2608.09096*, 2026.

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.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Siyuan Lu, Jiaqi Shao, Bing Luo, and Tao Lin. Morphagent: Empowering agents through self-evolving profiles and decentralized collaboration. *arXiv preprint arXiv:2410.15048*, 2024.

OpenAI. Introducing SWE-bench Verified, August 2024. URL `https://openai.com/index/introducing-swe-bench-verified/`. Developed in collaboration with the SWE-bench authors.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Dat Tran and Douwe Kiela. Single-agent llms outperform multi-agent systems on multi-hop reasoning under equal thinking token budgets. *arXiv preprint arXiv:2604.02460*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, et al. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*, 2024.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Arik. Multi-agent design: Optimizing agents with better prompts and topologies. In *International Conference on Learning Representations*, volume 2026, pp. 15844–15872, 2026.

A Appendix

A.1 Operational Details

The default structure contains three roles (planner, implementer, reviewer), three channels (planner→implementer delegation, implementer→reviewer review, and reviewer→implementer delegation), implementer aggregation, and two rounds. The cold-start structure contains one role, no channels, and two rounds. Boundary variants preserve the role prompts and aggregation while changing only the declared factor: sparse has one channel, dense has all six directed role pairs, and the round variants set the cap to one or three.

The implemented mutation family includes role addition, removal, merging, and specialization; channel addition, removal, and retyping; and procedure changes to aggregation and round count. Mutations are followed by pruning, output-format repair, and validation of role references, channel endpoints, duplicate channels, aggregation targets, and role/round bounds. Guided candidates and single-agent prompt updates are accepted when their four-example development score is no more than 0.02 below the rolling score. Random mutation retains the same development cost and guards but is accepted unconditionally. Every run stores the resolved configuration, per-task scores, token ledger, trajectories, mutation log, accepted structure snapshots, and rejected candidates.

A.2 Data, Budgets, and Scoring

The sampling manifest records sorted-ID sampling with fixed seeds: 20260907 for SWE-bench Verified and 20260909 for GPQA-Diamond. Each family has 20 development and 50 test instances, with disjoint identifiers. Development examples are used for adaptation and candidate selection; test examples are used only for scored execution and refresh passes. GPQA options are deterministically shuffled and the stored gold is an A–D letter. The SWE metric is the three-level patch proxy used in this study (1/0.5/0), not test-suite execution; this construct distinction applies to all SWE results in the paper.

For each arm, low and high cells allocate $50 \times 60{,}000 = 3.0$M and $50 \times 150{,}000 = 7.5$M tokens. Prompt and completion tokens from adaptation and execution share the same ledger. Charging is record-then-check, producing bounded call-granularity overshoot. Across the 19 primary runs, within-run arm spread in realized tokens ranges from 0.32% to 36.29% (median 5.89%); 18 runs are at most 14.82%. The SWE-high seed-3 outlier is included in the analysis.

A.3 Per-Seed Primary Results

Table 5: Primary-backbone scores by seed. Entries are evolve/frozen/single.

Cell	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5
SWE-low	.24/.19/.17	.32/.21/.12	.22/.23/.23	.23/.24/.21	.28/.23/.21
SWE-high	.29/.25/.13	.26/.24/.19	.20/.28/.21	.21/.22/.16	–
GPQA-low	.68/.72/.64	.66/.72/.60	.68/.72/.70	.70/.76/.66	.74/.70/.76
GPQA-high	.74/.66/.60	.70/.66/.68	.74/.70/.70	.68/.66/.66	.64/.74/.66

The paired analysis first averages available seeds within each task. The four cell-level Δ_{e-s} means and unadjusted percentile intervals are $+.070[+.024, +.118]$, $+.0675[+.005, +.1325]$, $+.020[-.024, +.064]$, and $+.040[-.016, +.100]$ in the table order above. For Δ_{e-f} they are $+.038[-.010, +.086]$, $-.0075[-.0725, +.060]$, $-.032[-.080, +.020]$, and $+.016[-.028, +.064]$. These cell CIs are descriptive and are not members of the Holm family.

A.4 Familywise Inference

The four-test family is two primary estimands (Δ_{e-s} and Δ_{e-f}) × two task families (SWE and GPQA). For each family, low/high aligned task-level contrasts are equally weighted after

seed averaging. The aggregation was finalized after initial result inspection; the associated familywise inference is therefore post hoc. Table 6 reports null-centered paired-bootstrap tests and Holm sequential step-down decisions. Percentile CIs are unadjusted. Paired t results are shown only as reference checks; they do not determine any primary familywise decision.

Table 6: Task-family contrasts: equal-weight low/high estimands, null-centered paired bootstrap (10,000 resamples, +1 correction), Holm sequential step-down. p_t denotes the paired-t reference analysis. [a]Rejected under task-conditional inference only; the rerun-aware interval spans zero.

Family	Contrast	Δ pp [95% CI]	Raw p	Holm p	p_t	Decision
SWE	e−s	$+6.875[+2.575, +11.675]$	0.00380	0.0152	0.00459	reject (task-cond.)[a]
GPQA	e−s	$+3.000[-0.400, +6.800]$	0.0962	0.289	0.113	retain
SWE	e−f	$+1.525[-2.550, +5.775]$	0.468	0.936	0.482	retain
GPQA	e−f	$-0.800[-3.600, +1.800]$	0.568	0.936	0.577	retain

A.5 Task×Seed Crossed Sensitivity (Exploratory)

The task-conditional familywise analysis treats the observed evolution runs as fixed and bootstraps tasks only. Table 7 reports, for each contrast, a task-only percentile interval (recomputed by the same script; endpoints sit within ≈0.2 pp of the null-centered task-only intervals in Table 6) and a crossed interval that additionally resamples the adaptation runs: within each tier, seed clusters are drawn with replacement to the original run count and applied jointly to both arms, with the resample deviation rescaled by $\sqrt{k/(k-1)}$ so the seed component keeps its unbiased sampling variance (10,000 resamples; the task resample is drawn once per iteration and shared across tiers because both budgets use the same 50 aligned tasks). The crossed model answers a different question: whether the contrast would survive re-running the whole random adaptation process.

Family	Estimand	Task-only 95% CI	Task×seed 95% CI
SWE	evolve−single	$[+2.5, +11.6]$	$[-0.7, +14.9]$
SWE	evolve−frozen	$[-2.6, +5.8]$	$[-5.5, +8.4]$
GPQA	evolve−single	$[-0.4, +6.8]$	$[-2.4, +9.1]$
GPQA	evolve−frozen	$[-3.8, +1.8]$	$[-6.4, +4.5]$

Table 7: Percentile bootstrap intervals (percentage points). Task-only CIs condition on the observed evolution runs (SWE: 5 low / 4 high; GPQA: 5/5); crossed CIs resample seed clusters with replacement to the original run count within each tier, paired across arms, modeling re-evolution. The SWE evolve−single contrast is positive under task-conditional inference but its crossed interval still crosses zero ($P(\text{diff} \le 0) \approx .04$), so the task-conditional finding applies to the observed runs rather than to independent re-evolutions. Script: `analyze_crossed_bootstrap.py`.

Equivalence margin sensitivity (exploratory). Assessing whether the frozen scaffold captures most of the measurable team benefit requires a pre-defined practically negligible margin, an equivalence/non-inferiority test, and an interval entirely inside that margin. We define the margin as 5 pp—the between-seed spread we observe on the SWE boundary grid (≈ 5 pp), i.e. the smallest gain distinguishable from run-to-run adaptation noise in this harness (the one-task granularity at $n = 50$ is 2 pp). A bootstrap two-one-sided test at $M = 5$ pp holds for evolve−frozen under task-conditional inference on GPQA ($[-3.8, +1.8] \subset [-5, +5]$) but not on SWE (upper bound $+5.8 > 5$) and not under the rerun-aware crossed intervals: SWE $[-5.5, +8.4]$ and GPQA $[-6.4, +4.5]$ would require margins $\ge 8.4/6.4$ pp, comparable to or larger than the study's only significant effect (evolve−single, +6.875 pp). Accordingly, these crossed intervals do not establish an evolution advantage that generalizes across reruns

in the studied settings. This exploratory sensitivity was specified after inspection of the primary results and lies outside the formal familywise analysis. The analysis is implemented in `analyze_equivalence_margin.py`.

A.6 Exploratory Validation of the SWE Output Proxy

To characterize the derived SWE output proxy, we analyze all saved answers from the SWE-high primary cells, the tier containing the main contrast. Table 8 recomputes, per task-evaluation record, the proxy level with the evaluator used in the reported experiments (`swarmevolve.evaluator.PatchExists`) and the corresponding keyword rule (the first `+++ b/` file of the gold patch in its original order), and measures file-level recall/precision against the gold patch's full changed-file set plus per-hit-file line-level Jaccard overlap with the gold hunks (multi-file gold patches contribute every hit file). The recomputation is deterministic across `PYTHONHASHSEED` values, with stable file selection and trajectory ordering. Recomputing with this evaluator reproduces the recorded scores on 1,697/1,700 records; the three discrepant records are truncated, budget-exceeded attempts whose saved partial answers still contain a patch block—recorded as failures (score 0) at run time—and are re-scored here for this diagnostic. The file-hit layer is near-exact at the file level (recall/precision ≥ 0.955), but about one in four (evolve) to one in five (single) score-1.0 outputs touches the correct file with *zero* line-level overlap with the gold hunks, so the proxy overstates fix quality relative to executed fail-to-pass tests; score-0.5 outputs are essentially file-misses. n counts task-evaluation records (final pass plus refresh passes across the four included seed runs with a summary), *not* independent experimental units; the primary sampling unit for run-level summaries is the seed run, whose summary adopts one score per seed–task pair.

Arm	Proxy level	n	File rec.	File prec.	P(line overlap>0)
evolve	1.0 (file hit)	132	0.972	0.955	0.765
evolve	0.5 (patch only)	47	0.000	0.000	0.000
evolve	0.0 (no block)	421	0.000	0.000	0.000
single	1.0 (file hit)	175	0.973	0.977	0.800
single	0.5 (patch only)	67	0.007	0.015	0.000
single	0.0 (no block)	858	0.000	0.000	0.000

Table 8: SWE output-proxy overlap diagnostics (n = task-evaluation records, final and refresh passes across the four included seed runs; n is not an independent-unit count—the run-level summary unit is the seed run). Fields: proxy level, record count, file-level recall/precision, and the fraction of score-1.0 records with positive line-level overlap. The analysis uses `analyze_proxy_overlap.py`; per-item mismatch explanations are provided in `proxy_overlap_audit_20260923.json`.

A.7 Seed-Level Dispersion

Table 13 reports sample SDs (ddof=1) across seed-level task means. These summarize run-to-run dispersion and are not the sampling unit for the task-conditional bootstrap. As a sensitivity, seed-level paired tests of the evolve–single contrast give $t = 1.95$ ($p\approx0.12$, SWE-low, n=5) and $t = 1.92$ ($p\approx0.15$, SWE-high, n=4): the task-conditional rejection is a statement about the task distribution conditional on the realized evolution runs; treating the evolution procedure itself as the sampling unit, the same contrast is not significant at n=4–5 seeds. Accordingly, the estimand is seed-averaged task-level performance conditional on the observed evolution runs, rather than performance over independent re-evolutions.

Additional design details. (i) The single agent's self-improvement notes are capped at 800 characters to match the scale of the structure-side mutation budget (one guarded operator per round, role prompts of comparable length), so neither arm can accumulate an unbounded private context; the cap binds rarely in practice. (ii) Evolution and self-improvement validate on the first four of the twenty development tasks: four tasks keep one validation pass below 10% of the evolution-phase budget share, and the subset is fixed across

rounds for comparability; the remaining sixteen dev tasks stay unseen by the adaptation loop. (iii) SWE-high has four rather than five seeds because available compute did not support the fifth run. (iv) The pre-run capability probe is a fixed prompt set scored once before primary data collection to identify the API-model version. All analyzed primary runs occurred within a single execution window after this probe. (v) The acceptance slack ϵ=0.02 is smaller than the smallest nonzero development-score change observed (0.125, one half-credit on one of four tasks; all 59 proposals across the four SWE boundary variants move in multiples of 0.125), so it admits score ties (24/59, the most common single outcome) and rejects every strict degradation; any $\epsilon \in (0, 0.125]$ yields identical accept/reject decisions at this granularity, placing the selected value within a decision-invariant range.

Realized-token direction. Table 9 reports per-arm realized tokens from the run ledgers. Beyond-±2% evolve–single gaps occur in SWE-high seed 3 (single-side early exhaustion, +51.5%) and SWE-low seeds 3 and 5 (evolve-side, +10.3%/+11.2%). Excluding those seeds raises the evolve–single score contrast from 6.75 to 9.33 pp (SWE-high, n=3) and from 7.00 to 9.67 pp (SWE-low, n=3). Together with the flat budget response on SWE (single .188 → .172 when allocated tokens double), these sensitivities do not support realized-use spread as the primary explanation for the reported SWE contrast.

Cell	Seed	evolve (M tok)	single (M tok)
SWE-high	1	7.53	7.67
SWE-high	2	7.66	7.62
SWE-high	3	7.54	4.97
SWE-low	1	3.03	3.04
SWE-low	2	3.02	3.06
SWE-low	3	3.36	3.04
SWE-low	4	3.02	3.03
SWE-low	5	3.38	3.04
GPQA-high	mean	7.63	7.64
GPQA-low	mean	2.98	3.08

Table 9: Per-seed realized tokens (SWE, where the contrast is rejected) and per-arm means (GPQA) from `ledger.json`. The remaining seed (SWE-high seed 4, not shown) lies within ±2%.

Cell	Arm	Calls /run	Tokens /task	Tokens /run	Pipeline wall-h	Recorded API charge (USD)	Score
SWE-low	evolve	469	63k	3.16M	8.1	0.00	0.258
SWE-low	frozen	459	65k	3.25M	8.1	0.00	0.220
SWE-low	single	387	61k	3.04M	8.1	0.00	0.188
SWE-high	evolve	1110	151k	7.57M	15.4	0.00	0.240
SWE-high	frozen	1019	156k	7.81M	15.4	0.00	0.247
SWE-high	single	883	140k	6.98M	15.4	0.00	0.173
GPQA-low	evolve	721	60k	2.98M	9.5	0.00	0.692
GPQA-low	frozen	700	64k	3.21M	9.5	0.00	0.724
GPQA-low	single	664	62k	3.08M	9.5	0.00	0.672
GPQA-high	evolve	1799	153k	7.63M	19.3	0.00	0.700
GPQA-high	frozen	1624	154k	7.69M	19.3	0.00	0.684
GPQA-high	single	1609	153k	7.64M	19.3	0.00	0.660

Table 10: Cost and score by cell and arm: LLM calls per run, realized tokens per task and per run (per-run means over unique runs; 50 tasks per run), pipeline wall-clock hours (all arms execute within one run), recorded API charge, and the score of the listed arm. Recorded monetary API charge is zero; compute and latency costs remain non-zero. Generated by `make_cost_table.py`.

Oracle per-task selection upper bound for the single arm (exploratory). Selecting, per task, the best of the independent single-arm runs gives SWE-low 0.380 / SWE-high

0.290 / GPQA-low 0.840 / GPQA-high 0.820, versus evolve means 0.258/0.240/0.692/0.700 and single means 0.188/0.173/0.672/0.660. This oracle is not deployable: it uses test outcomes, combines multiple independently budgeted runs, and is not cost-matched to one evolved run. It quantifies selection headroom only and should not be interpreted as an achievable baseline.

Exploratory allocated-budget-matched baseline decomposition. The single-agent arm uses a minimal refinement design, so the evolve−single gap may arise from multi-sampling, critique, or selection rather than team structure. Table 11 reports six exploratory follow-up baselines that probe these components on the same SWE task sets, budgets, and backbone (three seeds each; allocated-budget matched, realized token use reported per row): best-of-N single-agent sampling with self-evaluated selection (B1), a critique–revision pair (B2), best-of-N with a selector calibrated on the development split (B3), sequential self-refinement of a single agent (B4), a pseudo-team of four parallel non-communicating solos with a deterministic pick rule (B5), and the hand-designed scaffold with generic role prompts (B6). Several non-evolutionary baselines reproduce the observed score gap at the level of cell means: the critique pair matches evolve on SWE-high (.240 vs. .240) and the generic-prompt scaffold matches the frozen team (.243 vs. .247). Parallel sampling with selection exceeds evolve at both tiers under the same allocation (B3 .317/.300 low/high; B5 .327/.273, all three B5-high seeds $\geq$ .26), while self-evaluated selection alone (B1) does not. This pattern suggests that selection quality may contribute more than sampling volume in these runs. Sequential revision without sampling (B4) stays near the single arm. Together with the rerun-aware intervals (Section A.5), these comparisons are consistent with team and sampling/selection components contributing to the task-conditional SWE advantage; they do not isolate a causal contribution from structure evolution. Scripts: `analyze_baseline_campaign.py`; `baseline_campaign_20260924.json`.

Baseline (n=3)	Mechanism	SWE-low	SWE-high	T/task, high (k)
single (reference)	one self-improving agent	.188	.173	140
frozen (reference)	hand-designed team	.220	.247	156
evolve (reference)	team + adaptation	.258	.240	151
B1 best-of-N (self-eval)	sampling	.187	.213	69
B2 critique–revision	critique	.237	.240	77
B3 best-of-N (dev selector)	selection	.317	.300	171[a]
B4 sequential refinement	revision only	.190	.210	60
B5 pseudo-team (no comm.)	samp.+selection	.327	.273	59
B6 generic scaffold	topology	.240	.243	88

Table 11: Exploratory follow-up baseline decomposition on SWE (same 50-test tasks, budgets, and backbone as EXP-100; n=3 seeds per baseline; these analyses are outside the primary familywise analysis). Realized token use per task at the high tier from run ledgers (budget cap 150k/task; some baselines stop below it when their refinement passes terminate on a truncated round). [a]B3 exceeds the cap by $\approx$ 14% (the ledger allows one-call overshoot; its per-call prompts are the largest).

Exploratory fail-to-pass execution on the executable subset. The SWE outcome is a patch-format/file-target proxy, not executed repair. To assess the proxy's construct validity, we re-evaluated all SWE-high answers of the three reference arms and baselines B2/B3/B5 (21 arm–seed combinations, 50 tasks). We used the task-specific environment images, evaluation scripts, and log grader generated by the SWE-bench package available in our execution environment. Because the execution node lacks container mount namespaces, each image was exported to a root filesystem and its evaluation script was run through `chroot`. This path uses the same task scripts and grading interface while replacing the Docker runtime with `chroot`. The auxiliary patch-application procedure is specified in the released script. 48/50 tasks were executable; two xarray instances were excluded because image export produced broken hardlinks. Table 12 summarizes four diagnostics.

Diagnostic	Result (SWE-high executable subset, 1008 evaluations)
Proxy–execution agreement	accuracy .812 carried by negatives; proxy 1.0: 9.9% pass fail-to-pass (precision .099, recall .741; 20 TP / 183 FP / 7 FN / 798 TN)
Resolved rate by proxy level	$1.0 \to 9.9\%$, $0.5 \to 1.5\%$, $0.0 \to 0.8\%$
Arm execution rates (seed-averaged)	evolve .031, frozen .000, single .026, B2 .035, B5 .056, B3 .021
Execution contrasts	evolve−single: 6 vs. 5 pairs, McNemar p=1.00; evolve−frozen: 6 vs. 0, p=0.0312 (uncorrected, exploratory); evolve−B3: 3 vs. 3, p=1.00; evolve−B5: 3 vs. 8, p=0.227
Repair-success comparison	evolve−single: no detected difference; low-base-rate exploratory result

Table 12: Fail-to-pass execution on the executable subset (SWE-bench evaluation scripts and grader in a chroot runtime). This exploratory follow-up has low base rates, so the contrasts carry substantial uncertainty. The proxy serves as a diagnostic overlap measure rather than a measure of repair success. Scripts: `extract_execution_answers.py`, `run_execution_validity.py`, `analyze_execution_validity.py`.

Table 13: Seed-level sample SD by budget cell. Columns are arm scores followed by contrast SDs.

Cell	evolve	frozen	single	Δ_{e-s}	Δ_{e-f}
SWE-low	0.041	0.020	0.044	0.080	0.050
SWE-high	0.042	0.025	0.035	0.070	0.053
GPQA-low	0.030	0.022	0.061	0.037	0.041
GPQA-high	0.042	0.036	0.037	0.060	0.068

For the auxiliary common-seed family summaries, the SWE SDs are .037722/.015546/.032275 for evolve/frozen/single and .067992/.050394 for $\Delta_{e-s}/\Delta_{e-f}$ (four common seeds). The corresponding GPQA SDs are .013416/.013416/.038471 and .040620/.019235 (five common seeds). These are auxiliary summaries only.

A.8 Equal-Weight Cross-Family Summary

Equal weighting of the SWE and GPQA task-family point estimates gives descriptive arm means .472500/.468875/.423125 for evolve/frozen/single. The corresponding contrasts are +4.9375 pp for evolve–single and +0.3625 pp for evolve–frozen. This cross-family summary is descriptive only; no additional pooled hypothesis test is performed.

A.9 Cold-Start, Ablation, and Backbone Tables

Table 14: Mechanism probes at the high budget. Each entry lists seeds 1/2/3.

Initialization	Family	Guided	Random mutation
Single role	SWE	.14/.16/.25	.22/.28/.45
Single role	GPQA	.74/.72/.66	.70/.74/.68
Pipeline	SWE	.32/.30/.30	.18/.42/.28

The backbone experiment uses the same 20/50 splits and equal-allocation scheduler. Its realized-token spread ranges from 0.63% to 4.61%. The combination of a near-tie on SWE and a consistent GPQA deficit for evolution is evidence against a backbone-independent benefit.

Table 15: Qwen3-8B robustness means over three seeds (evolve/frozen/single).

Family	60k	150k
SWE proxy	.667/.657/.633	.643/.650/.647
GPQA	.320/.473/.467	.320/.460/.427

A.10 Failure Case: Cold-Start Guided Mutation

In the cold-start condition (EXP-200, SWE family), the guided pipeline consistently underperformed random mutation across all three seeds. We examined the mutation logs to identify the failure mode. The guided experience extractor attributes failures to the *only existing role*, producing mutations that repeatedly re-prompt that single role or retune the aggregation policy rather than adding complementary capabilities. In seed 1, the first two mutations both apply `op_patch_format` to the single lead role, and the round-1 candidate drops the dev score from 0.25 to 0. By contrast, in seed 3 the random control applies `op_add_reviewer` in round 1—a mutation the guided pipeline never proposes in any seed—raising the dev score from 0.25 to 0.5. This is a concrete instance of the failure-localization signal being miscalibrated without an existing scaffold: the signal can only refer to components that exist, so from a single-role seed it over-specializes rather than explores. This contrast is consistent with undirected mutations discovering complementary roles that the signal does not propose. This analysis motivates research on structure-agnostic credit assignment—signals that can reason about roles and channels that do not yet exist.

Operator distributions are consistent with the proposed mechanism. Across the three cold-start SWE seeds, the guided pipeline proposed `op_change_aggregation` in 26 of 30 mutations (87%) and `op_patch_format` in the remaining 4. The random-control pipeline proposed six distinct operators across 17 mutations: `op_add_role`, `op_add_reviewer`, `op_specialize`, `op_merge_roles`, `op_change_max_rounds`, and `op_change_aggregation`. The observed distribution is consistent with the extractor concentrating proposals on aggregation-related changes from a single-role seed, which may restrict structural exploration from this initialization. The acceptance criterion blocks many of these narrow proposals for degradation, further reducing the guided arm's effective search breadth relative to random exploration.

Exploratory proposal×acceptance factorial. Table 16 combines the pre-inspection EXP-200/EXP-220 cells with the exploratory EXP-236 run, which supplies the previously missing random+conditional cell through a direct experiment (three seeds, same cold-start seed, budget, and tasks). The exploratory factorial estimates a larger proposal effect than acceptance effect, with a substantial proposal-by-acceptance interaction; uncertainty remains large at three seeds per cell. The proposal main effect (random−guided, averaged over acceptance) is +0.100 [exact stratified permutation $p = 0.0550$; bootstrap 95% CI $[0.032, 0.172]$], the acceptance main effect is −0.033 [$p = 0.560$; bootstrap 95% CI $[-0.105, 0.035]$], and the interaction is −0.127 [exact within-cell residual permutation $p = 0.158$; bootstrap 95% CI $[-0.270, 0.007]$]. Uniform random proposals with no filtering score highest (.317), signal-guided proposals lowest (.153–.183), and an additive-prior proposal distribution recovers most of the gap (.263; .280 with an eight-task validation subset). The acceptance filter interacts with proposal quality rather than acting uniformly: it slightly helps the weakest (guided) proposals (.153 → .183) and hurts uniform-random proposals (.317 → .220). Neither tested acceptance policy closes the gap for signal-guided proposals; the larger recoverable component in this setting is associated with the proposal distribution.

A.11 Prespecified Data Collection and Exploratory Analyses

Table 17 separates runs specified before result inspection from exploratory follow-ups. Prespecification applies to data collection, not to the post-hoc low/high aggregate inference. Retrospective readouts of prespecified runs remain exploratory; *trajectory analysis* uses existing runs without new executions, and *terminated* designs yield no data.

Proposal	Acceptance	Score	Source
Prespecified experimental cells:			
guided	cond.	0.183 (n=3)	EXP-200 evolve arm
guided	uncond.	0.153 (n=3)	EXP-220
random	uncond.	0.317 (n=3)	EXP-200 random arm
Exploratory experimental cell:			
random	cond.	0.220 (n=3)	EXP-236
Exploratory additive-prior proposals:			
additive prior	cond.	0.263	EXP-230
additive, 8-task val.	cond.	0.280	EXP-237
Observational log counterfactual (not an experimental cell):			
random	"cond."	29% would-reject	EXP-200 logs

Table 16: Proposal×acceptance factorial at SWE high budget, cold-start single-role G_0. The three EXP-200/220 cells were specified before result inspection. The random+conditional cell comes from the later direct EXP-236 experiment; because acceptance changes subsequent trajectories, it is distinct from the observational log counterfactual. The additive-prior EXP-230/237 rows are also exploratory follow-ups. All cells n=3 seeds; scripts: `analyze_factorial.py`, `analyze_exp230.py`.

Exp.	Family / budget	Design	Analysis class
EXP-100	SWE+GPQA, low/high	evolve / frozen / single, 4–5 seeds	prespecified run; post-hoc inference
EXP-200	SWE+GPQA, high	cold start: guided vs. random mutation, 3 seeds	prespecified run; exploratory readout
EXP-210	SWE, high	takeover ablation: guided vs. random, 3 seeds	prespecified run; exploratory readout
EXP-220	SWE, high	guided proposals, unconditional acceptance, 3 seeds	prespecified run; exploratory readout
EXP-110	SWE, both tiers	budget-illusion sensitivity	terminated
EXP-230	SWE, high	additive-prior proposals, 3 seeds	exploratory follow-up
EXP-236	SWE, high	random proposals, enforced acceptance, 3 seeds	exploratory follow-up
EXP-237	SWE, high	additive prior, 8-task validation, 3 seeds	exploratory follow-up
EXP-300	SWE+GPQA, high	four boundary variants, 3 seeds	exploratory follow-up
EXP-401	SWE, low/high	six allocated-budget-matched baselines, 3 seeds	exploratory follow-up
EXP-200 logs	SWE, high	random-arm acceptance counterfactual	trajectory analysis

Table 17: Experiment and analysis classification. EXP-110 was terminated with no data; exploratory follow-ups and trajectory analyses are outside the primary familywise analysis.

A.12 Analysis Inclusion and Reproducibility

Two preliminary three-arm runs are excluded because infrastructure and pipeline failures made their outputs unsuitable for analysis: a run-ID collision, faulty candidate acceptance/output extraction, and test-set leakage into adaptation. The analyzed runs separate development and test files and use the guarded acceptance path. Primary data collection uses 60k/150k token tiers because the initial 30k/90k tiers could not accommodate a complete multi-role pass under the 8k call cap. Incomplete boundary cells and the unfinished budget-illusion sensitivity yield no evidence. The boundary variants are exploratory analyses defined after inspection of the primary results.

Source-data manifests and hashes, resolved configurations, per-run summaries, token ledgers, mutation trajectories, and the analysis code used to generate the reported tables and figures are provided in an anonymous supplementary package, submitted with the paper where available. The familywise bootstrap uses 10,000 resamples with a fixed random seed. Run records include the API-model identifier and call window; the Qwen environment uses vLLM 0.8.5.post1, PyTorch 2.6.0, Transformers 4.51.3, and Tokenizers 0.21.1. Raw benchmark bodies and credentials are omitted from all released materials.

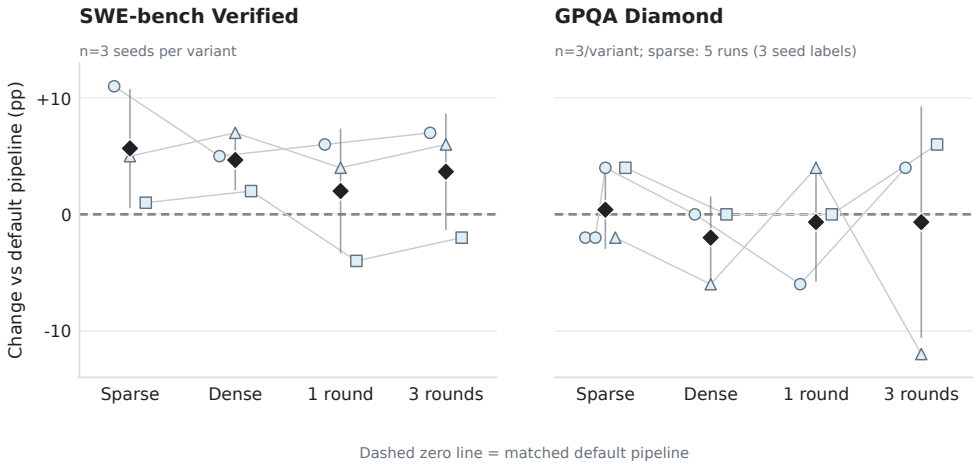

Figure 5: Boundary experiments (exploratory follow-up; n=3 runs per cell except GPQA sparse: five runs with three unique seed labels). Markers show individual runs against the matched default-pipeline reference (seed shape: circle / triangle / square; thin lines join the same seed across variants; diamonds and bars give cell mean ± sample SD). On SWE no variant leads in all three seeds and on GPQA every cell mean stays within two points of the reference—run-to-run variance dominates structural variation at this grid size.

Where the boundary differences live (task level). Across the SWE boundary runs (three seeds, same 50 test tasks), 24–29 tasks score zero under *every* structural variant and the full-score (proxy level 1.0) set shrinks from 9 to 2 tasks by seed 3 (7 of the seed-1 full scores are Django repository tasks); the variants differ mainly on the remaining 13–20 borderline tasks, up to 10 of which receive the partial proxy score. Structure choice therefore changes the fate of borderline patch-format hits rather than unlocking new task classes. Notably, the sparse variant's mean score *within the borderline band* is stable across all three seeds (0.500/0.500/0.525); its overall variation ($0.350 \to 0.290 \to 0.250$) tracks the shared floor and the collapse of the full-score set, which shift the band boundary itself—the same floor-drift pattern coincides with the round variants' seed-3 drops. A similar pattern appears in the main experiment's SWE-high evolve arm (full-score tasks $13 \to 12 \to 9 \to 8$ over its four seeds, mean $.29 \to .21$). Because seeds run in chronological order, slow backbone drift cannot be fully excluded for the multi-day EXP-100 window (GPQA-high shows a weaker version of the same tail decline, $.74 \to .64$); however, the boundary grid's declines occur within a single day, which argues against drift as the sole cause. By repository (seed 1), sparse attains full scores on 9.5/26 Django tasks but 0.5/6 SymPy and 0/3 scikit-learn tasks.

Evolved final structures. Table 18 records the final accepted structure of the evolution loop for each completed boundary run (seed 1). The pattern is task-conditional: on SWE-bench every variant converges to a communication-light structure (at most two channels; the sparse variant keeps exactly one), while on GPQA-Diamond the sparse-seeded loop *adds* verification roles and channels (four roles, five channels, vote aggregation) and the dense-seeded loop *prunes* (six channels to three, ending reviewer-led)—opposite paths converging on similar scores (0.680/0.704). This is consistent with channel overhead being a salient cost in SWE traces (though not in score rankings, which seed variance dominates) but not on GPQA-Diamond. The endpoints are partly seed-stable: rounds3 ends at the identical three-

role / two-channel / implementer structure in all three seeds; the SWE sparse variant ends at three roles / one channel in seeds 1–2 (final aggregator differs) but *grows* to five roles / four channels at seed 3; the GPQA dense-seeded loop ends at or below its six initial channels in all three seeds (three, five, and six); SWE rounds1 diverges (three roles/two channels at seed 1 vs. four roles/four channels at seed 2); the SWE dense seed 2 prunes one channel (six to five; roles unchanged). Variants define *initializations*, not enforced constraints—one run (GPQA one-round seed 1) raised its round cap from 1 to 2—and SWE structure variance across seeds mirrors the score variance.

Cell	Final roles	Ch.	Mutations	Final aggregation
SWE sparse	planner, implementer, reviewer	1	4	planner
SWE rounds1	planner, implementer, reviewer	2	5	concat
SWE rounds3	planner, implementer, reviewer	2	2	implementer
GPQA sparse	planner, reviewer, verifier_1, verifier_2	5	5	vote
GPQA dense	reviewer, verifier_1	3	7	reviewer
GPQA rounds1	reviewer (+1)	3	16^*	reviewer
SWE dense	planner, implementer, reviewer	6	1	implementer

Table 18: Structure evolution in the completed seed-1 boundary runs (`G_round_*.json` snapshots). The SWE dense seed-1 run accepted only a patch-format mutation (procedure-level; structure unchanged). *The GPQA one-round run is an outlier in accepted-mutation count (16) and escaped its round cap (1→2). All seeds start from the same three-role pattern; the SWE sparse seed has one channel (mutations only rotate aggregation), the round-count seeds have three channels (both variants prune to two), the GPQA sparse seed grows from one to five channels, and the GPQA dense seed prunes from six to three. "Mutations" counts accepted mutations; rejected candidates are logged separately.

Boundary per-seed scores. Table 19 reports per-seed evolve scores for each boundary cell; all cells are complete (n=3 per cell; GPQA sparse has five runs over three unique seeds, listed by their seed-mean). The default two-round pipeline reference is the EXP-100 evolve-arm mean at the matched high budget tier (SWE 0.240, n=4; GPQA 0.700, n=5).

Task family	Variant	Seeds (1/2/3)	Mean
SWE	sparse	.350/.290/.250	.297
SWE	rounds1	.300/.280/.200	.260
SWE	rounds3	.310/.300/.220	.277
SWE	dense	.290/.310/.260	.287
SWE	pipeline (EXP-100 ref)	0.240 (n=4)	
GPQA	sparse	$.700^{\times 3}$/.680/.740	.704
GPQA	dense	.700/.640/.700	.680
GPQA	rounds1	.640/.740/.700	.693
GPQA	rounds3	.740/.580/.760	.693
GPQA	pipeline (EXP-100 ref)	0.700 (n=5)	

Table 19: Boundary grid per-seed evolve scores (exploratory follow-up; all cells $n \geq 3$, sparse five runs). $^{\times k}$ marks k replicate runs of the same seed (the GPQA sparse cell has three independent seed-1 runs, .680/.680/.740, shown as their mean .700). Both families vary channel density and round count.