CAUSAL IMPROVEMENT GRAPH FOR AGENTIC HARNESS OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic Harness is the runtime that constructs task context and controls execution flow, thereby shaping overall agent performance. Given a fixed model and external evaluation, automated Harness optimization seeks to improve this runtime through an iterative proposal–evaluation loop to better solve target tasks. Existing meta-harness methods mainly adopt *proposer-centric discovery*, in which an LLM-based proposer integrates accumulated experimental findings to determine subsequent Harness revisions. This places the burden of maintaining the evolving improvement state on the proposer as history expands and its underlying experimental logic becomes harder to discern. In this paper, we introduce the *Causal Improvement Graph* (CIG), a graph-governed meta-harness framework that externalizes the evolving improvement state in a persistent graph, allowing prior findings to directly govern subsequent Harness optimization through local proposer operations. CIG grows and links EVIDENCE, HYPOTHESIS, INTERVENTION, and OUTCOME nodes to represent what was observed, how it may be explained, how to test that explanation, and what the evaluation reveals. Their structural relations preserve how the improvement state changes across iterations, allowing local proposers to build directly on relations among prior findings rather than recover them from raw history. Across various agent tasks, CIG discovers stronger Harnesses than previous meta-harness baselines and remains robust to the choice of task solver and proposer. Structural ablations further support the design of an explicit improvement state with graph-governed evolution.

1 INTRODUCTION

As foundation models increasingly operate as agents, Harness design has emerged alongside model development as a major route to improving agent performance (Lee et al., 2026b; Lin et al., 2026). A *Harness* is the programmable runtime that turns model capabilities into task behavior by constructing context, mediating tools, and coordinating execution, as realized in agent systems such as Codex, Claude Code, and OpenClaw (Lopopolo, 2026; Shihipar, 2026; OpenClaw, 2026). While model optimization has a mature training stack (Brown et al., 2020; Ouyang et al., 2022), Harness optimization remains substantially dependent on engineering judgments about context, tools, and execution (Lee et al., 2026b; Lin et al., 2026). Making the observations and reasoning behind these judgments reusable is a central challenge for automated Harness optimization.

Automated Harness optimization makes the harness an object of the search process (Lou et al., 2026; Luo et al., 2026). Meta-Harness (Lee et al., 2026b) enables end-to-end search for stronger Harnesses through an agentic proposer that autonomously inspects prior code, scores, and execution traces via the filesystem, with minimal prescribed structure in the outer loop. More structured approaches (Lin et al., 2026; Chen et al., 2026b; Zhang et al., 2026a; Chen et al., 2026c) organize execution feedback into diagnostic evidence, explicit change proposals, and validation procedures to make Harness updates more targeted and verifiable. Other approaches (Lee et al., 2026a; Zhang et al., 2026c) organize reuse across rounds through pairwise feedback over revision histories and archives of alternative Harness lineages for selection and recombination. These approaches accumulate increasingly rich histories, candidate archives, and diagnostic records. As these materials grow substantially, the relationships among prior findings and their implications for further optimization can become harder to track (Park et al., 2026), leaving the proposer to reconstruct the evolving improvement state.

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.

This reconstruction burden can reduce proposal efficiency and leave subsequent optimization insufficiently guided by accumulated experience and prior progress. To support further discovery, history should grow into an explicit, revisable improvement state that preserves accumulated experience, maintains an evolving understanding of that experience, and guides subsequent Harness optimization.

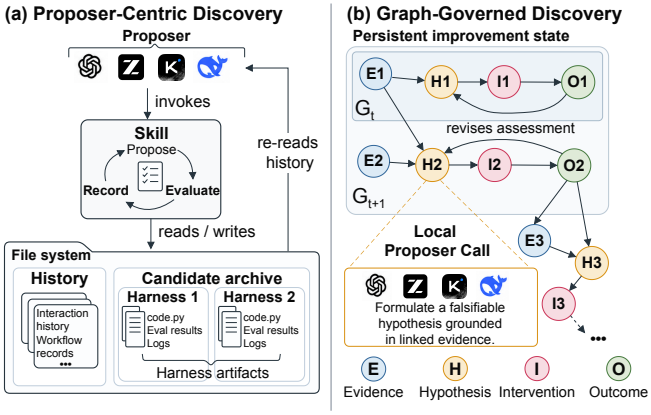


Figure 1: Comparison of proposer-centric discovery and graph-governed discovery. Left: a proposer follows a prescribed procedure to manage an end-to-end process over interaction records and candidate artifacts. Right: CIG makes the growing graph the persistent improvement state, invokes proposers for locally scoped operations, and updates the graph and its assessments using externally evaluated outcomes.

In this paper, we introduce the **Causal Improvement Graph (CIG)**, which shifts Harness optimization from proposer-centric discovery to graph-governed discovery. By linking EVIDENCE, HYPOTHESIS, INTERVENTION, and OUTCOME nodes, CIG organizes raw improvement history into a persistent improvement state that connects observations, explanations, proposed changes, and evaluated outcomes to guide further Harness optimization. As illustrated in Figure 1, CIG maintains the graph as a persistent improvement state and invokes proposers for locally scoped operations. Externally evaluated outcomes revise hypothesis assessments and update the state that guides subsequent Harness optimization.

The main contributions of this work are:

- **Graph-Governed Discovery.** We identify the burden of reconstructing evolving improvement state in proposer-centric discovery and reformulate Harness optimization as graph-governed discovery, guided by a persistent improvement state.
- **Causal Improvement Graph.** We introduce CIG, which grows, links, and updates EVIDENCE, HYPOTHESIS, INTERVENTION, and OUTCOME nodes to represent the improvement state and guide subsequent Harness optimization.
- **Empirical effectiveness and structural attribution.** We demonstrate CIG's effectiveness in discovering stronger Harnesses across tasks and its robustness to the choice of solver and proposer. Ablation studies clarify how CIG's structural design contributes to these gains.

2 Related Work

Harness evolution. Agent design and workflow search provide foundations for Harness evolution: ADAS (Hu et al., 2025) generates programs from archived designs, while GPTSwarm (Zhuge et al., 2024) and AFlow (Zhang et al., 2025b) organize optimization around computational graphs and code-represented workflows, respectively. At the Harness level, Meta-Harness (Lee et al., 2026b) searches executable code using prior candidates, scores, and traces. To make these updates more explicitly grounded in execution evidence, AHE (Lin et al., 2026) structures that evidence and verifies modification predictions against subsequent outcomes, while HarnessFix (Chen et al., 2026b) connects structured trace diagnosis to targeted repairs and retains records of accepted and rejected changes. **CIG** makes the evolving relations among evidence, hypotheses, interventions, and outcomes an operational state: hypotheses retain revisable assessments across experiments, and their current relations and assessments guide subsequent planning and portfolio selection. Recent work (Zhou, 2026; Park et al., 2026; Du et al., 2026; Ren et al., 2026) further explores explicit evolution procedures and persistent state for Harness evolution, procedural adaptation, and autonomous research. Appendix A.2 discusses these methods in greater detail, comparing their state representations and update targets with particular attention to graph-based approaches.

Self-improving agents. Self-improving agents (Zelikman et al., 2024; Zhang et al., 2025a) use feedback to revise task-solving behavior and, increasingly, the procedures that produce these revisions. RHI (Lee et al., 2026a) refines prompt-level Harness specifications using pairwise execution

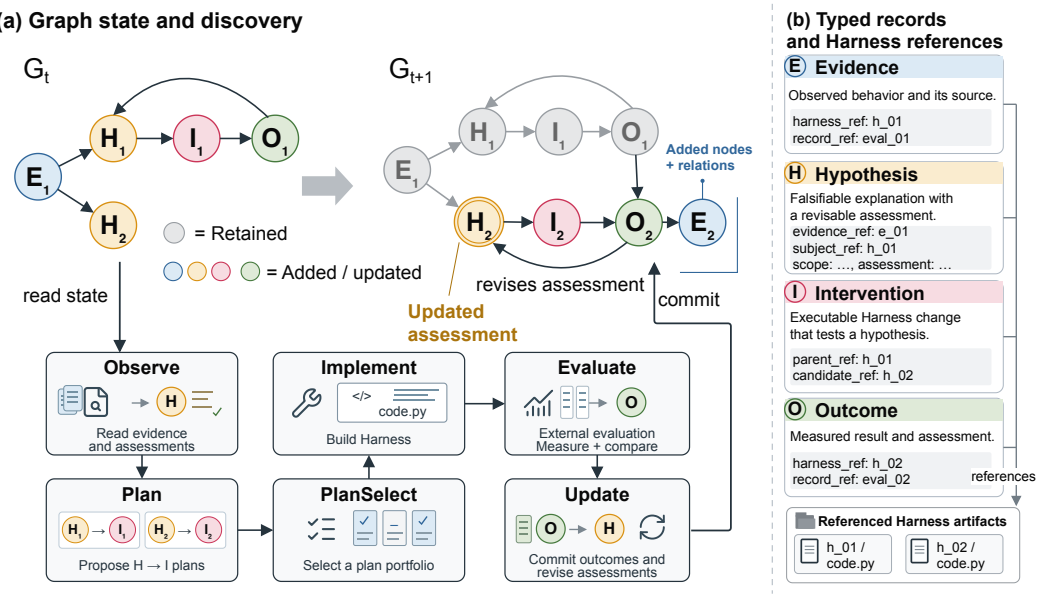


Figure 2: Improvement state transitions in CIG. Left: graph-governed discovery operations transform G_t into G_{t+1} by adding and linking evidence, hypothesis, intervention, and outcome nodes and revising existing hypothesis assessments based on evaluated outcomes. Right: typed records represent the improvement state and reference the corresponding Harness artifacts.

feedback, while Hyperagents (Zhang et al., 2026b) makes the meta-level modification program editable and MetaSkill-Evolve (Wang et al., 2026b) jointly evolves task skills and the strategies used to improve them. **CIG** focuses on the improvement state supporting this process: evaluated Harness changes revise linked hypotheses and inform which interventions to attempt next.

3 Method

In this section, we formalize the Harness optimization problem and introduce the Causal Improvement Graph (CIG), which maintains and updates a persistent improvement state. Figure 2 summarizes the state transition and the information represented by each typed node, which we detail through the state representation, discovery operations, and outcome-grounded update.

3.1 Harness Optimization

A Harness is the executable runtime that constructs task context and controls an agent's execution flow. We study the problem of discovering an effective Harness under a fixed model and external evaluation. A proposer is an LLM-based agent that uses prior candidate artifacts and evaluation feedback to propose and implement Harness changes. The discovery process alternates candidate construction with external evaluation of the resulting Harnesses. Let $\mathcal{H}$ denote the space of possible Harnesses and let $\mathcal{H}_0 \subset \mathcal{H}$ be an initial set. Let B denote the candidate-evaluation budget, measured in complete candidate evaluations. A discovery method constructs a candidate set $\mathcal{C}_B \subset \mathcal{H}$ within this budget. The evaluation function $\mathcal{E}$ maps each candidate Harness $h \in \mathcal{H}$ to its measured behavior on a task distribution $\mathcal{D}$,

$$\mathbf{y}(h) = \mathcal{E}(h, \mathcal{D}) \in \mathcal{Y}. \tag{1}$$

The outcome space $\mathcal{Y}$ may contain a single task score or multiple dimensions such as quality, reliability, cost, and latency. A selection policy Π, fixed before discovery, chooses the final Harness from the evaluated candidates,

$$\hat{h}_B = \Pi(\{(h, \mathbf{y}(h)) \mid h \in \mathcal{C}_B\}). \tag{2}$$

This formulation accommodates different selection policies without defining Harness quality as a single scalar. While Π selects the final Harness, discovery must decide which changes to try within the evaluation budget. This requires using earlier experiments to judge which directions remain promising and which judgments need revision.

3.2 CAUSAL IMPROVEMENT STATE

Proposer-centric methods use accumulated candidate artifacts, evaluations, and diagnostic records to propose further changes. These records preserve accumulated experience, but the proposer must reconstruct how prior findings relate and whether earlier judgments still hold when planning the next change. CIG makes these relationships and current judgments explicit in a persistent improvement state. Linking observations recorded as evidence to hypotheses, interventions, and outcomes allows a new result to revise the judgments it bears on, while preserving their experimental basis. Subsequent proposals can then build on the updated state, connecting what has been observed to what to improve next and which earlier judgments need revision.

We represent this state at discovery round t as a committed causal improvement graph $G_t = (V_t, R_t)$. We use *causal* in the operational sense of intervention-grounded improvement reasoning. Formal causal identification of the task distribution is outside this formulation. The four semantic node types of G_t are EVIDENCE, HYPOTHESIS, INTERVENTION, and OUTCOME. Harness artifacts and plans are associated records linked to these nodes through references. Its relations R_t record provenance, support, contradiction, and Harness lineage. A basic causal path in the graph is

$$\text{EVIDENCE} \longrightarrow \text{HYPOTHESIS} \longrightarrow \text{INTERVENTION} \longrightarrow \text{OUTCOME}. \tag{3}$$

Concretely, an EVIDENCE node stores an observed behavior with its task and Harness provenance. A linked HYPOTHESIS node states a falsifiable explanation. An INTERVENTION node specifies the executable Harness change intended to test it, and an OUTCOME node records the measured behavior and its verdict. These measurements come from external evaluation, which executes the candidate Harness and measures its behavior using $\mathcal{E}$. These results supply evidence for the next discovery round. Across successful and unsuccessful experiments, G_t retains the observations, hypotheses, interventions, and outcomes together with their logical relations, source references, and current hypothesis assessments. This structured record forms the persistent improvement state. Subsequent operations can trace a proposed change to its supporting observations and consult updated assessments of related hypotheses, making accumulated experience directly usable in planning further Harness optimization from linked observations and current hypothesis assessments.

3.3 GRAPH-GOVERNED DISCOVERY LOOP

CIG advances the persistent improvement state represented by G_t through graph-governed discovery operations. Each discovery round updates G_t to G_{t+1} through the following sequence:

$$G_t \xrightarrow{\text{OBSERVE}} A_t \xrightarrow{\text{PLAN}} P_t \xrightarrow{\text{PLANSELECT}} S_t \xrightarrow{\text{IMPLEMENT}} \widetilde{\mathcal{H}}_t \xrightarrow{\text{EVALUATE}} O_t \xrightarrow{\text{UPDATE}} G_{t+1}. \tag{4}$$

The intermediate variables denote linked working products within this transition: an analysis A_t of the current graph state, a plan set P_t, a selected portfolio S_t, realized Harnesses $\widetilde{\mathcal{H}}_t$, and grounded OUTCOME nodes O_t. OBSERVE reads the current graph and permitted raw history to summarize what has been learned and what remains unclear. Each output cites addressable evidence, so a later plan can recover the observations behind it. PLAN proposes falsifiable HYPOTHESIS–INTERVENTION pairs with named parent Harnesses, preserved mechanisms, and expected outcomes. PLANSELECT uses a separate proposer call to assess every plan on seven 0–4 scales: evidence strength, expected gain, information value, novelty, falsifiability, cost, and risk. It selects a portfolio using these assessments and pairwise redundancy, including both an evidence-grounded refinement and a branch or restart when the selected budget permits. The selector records its selection rationale and rejection reasons. In Full CIG, these judgments use the current relations to connect each proposed intervention to its evidence and hypothesis, including unresolved or revised conclusions. Independent IMPLEMENT operations realize the selected plans as executable Harnesses. EVALUATE measures their behavior, and UPDATE incorporates the grounded OUTCOME nodes into the graph and revises hypothesis assessments. Through these operations, CIG connects each evaluated change to the state used for subsequent planning, allowing later proposals to build on prior results and revised judgments across discovery rounds.

Proposers are invoked for locally scoped operations. They interpret EVIDENCE nodes, formulate HYPOTHESIS nodes, or implement a selected INTERVENTION node, but they do not own the persistent state or the complete discovery process. The graph supplies each invocation with a role, a local objective, and explicit dependencies on prior work. Algorithm 1 summarizes the resulting loop.

3.4 OUTCOME-GROUNDED UPDATE

The final UPDATE step in each discovery round (Equation 4) incorporates externally evaluated candidate behavior into the persistent improvement state through a structured graph transition. For candidate $h_{t,p}$ and its parent Harness, CIG records

$$\tau_{t,p} = \left(\mathbf{y}_{t,p}, \mathcal{T}^{+}_{t,p}, \mathcal{T}^{-}_{t,p}, \mathcal{T}^{0}_{t,p}, e_{t,p}, a_{t,p}, v_{t,p}\right). \tag{5}$$

The sets $\mathcal{T}^{+}_{t,p}$, $\mathcal{T}^{-}_{t,p}$, and $\mathcal{T}^{0}_{t,p}$ record corrected, regressed, and unchanged outcomes relative to the parent. Intervention fidelity $e_{t,p}$ records whether the intended change was executed, association $a_{t,p}$ compares its effects with the predicted direction, and verdict $v_{t,p}$ assesses the linked hypothesis at the scope supported by the observations. Unavailable execution or task-transition evidence remains unspecified. Appendix A.5 distinguishes score-based judgments from evidence about the underlying mechanism. The update operator appends these transitions and their dependencies to the persistent state,

$$G_{t+1} = \mathrm{Update}(G_t, \{\tau_{t,p} \mid p \in S_t\}). \tag{6}$$

An OUTCOME node serves two roles: measured performance informs Harness selection, while the linked prediction and observed effects update discovery knowledge. Revision separates measured results from their interpretation. The evaluator supplies scores and parent-relative changes, while reasoning over linked evidence and hypotheses compares these observations with the linked prediction and revises its current assessment. Conflicting results can narrow the supported scope or leave an explanation unresolved. The graph retains the original experiment and its provenance alongside the assessment used for subsequent observation and planning.

Subsequent observation and planning read these assessments together with their evidence. An unsuccessful candidate can therefore expose an informative trade-off or challenge an earlier explanation. Performance selection follows the fixed eligibility policy, independently of whether every proposed explanation is supported. Appendix A.6 describes the retained records and current hypothesis assessments.

Algorithm 1: CIG discovery loop

Require: Initial Harnesses $\mathcal{H}_0$, evaluation $\mathcal{E}$, policy Π, budget B
Ensure: Selected Harness $\hat{h}_B$, final graph G_T

$G_0 \leftarrow \textsc{Initialize}(\mathcal{H}_0)$
for $t = 0, 1, \ldots$ while budget remains **do**
  $A_t \leftarrow \textsc{Observe}(G_t)$
  $P_t \leftarrow \textsc{Plan}(G_t, A_t)$
  $S_t \leftarrow \textsc{PlanSelect}(G_t, P_t, B)$
  for all $p \in S_t$ **in parallel do**
    $h_{t,p} \leftarrow \textsc{Implement}(p, G_t)$
    $\mathbf{y}_{t,p} \leftarrow \textsc{Evaluate}(h_{t,p}, \mathcal{E})$
    $\tau_{t,p} \leftarrow \textsc{Compare}(h_{t,p}, p, \mathbf{y}_{t,p})$
  end for
  $G_{t+1} \leftarrow \textsc{Update}(G_t, P_t, S_t, \{\tau_{t,p}\})$
end for
$\hat{h}_B \leftarrow \textsc{HarnessSelect}(G_T, \Pi)$
return $\hat{h}_B, G_T$

In summary, CIG externalizes the evolving improvement state in a persistent graph. Explicit relations connect evidence, hypotheses, interventions, and outcomes, while externally evaluated outcomes revise hypothesis assessments across iterations. This allows local proposers to build directly on prior experimental dependencies and updated assessments rather than recover them from raw history, enabling prior findings to directly govern subsequent Harness optimization.

4 EXPERIMENTS

In this section, we investigate five questions to assess CIG's effectiveness across environments and models and understand how its search allocation, structural components, and outcome-grounded revisions contribute to automated Harness optimization.

- ***RQ1:*** *How effective is CIG compared with existing approaches?*
- ***RQ2:*** *How do CIG's search schedules perform across solvers?*
- ***RQ3:*** *Does CIG remain effective across proposer models?*
- ***RQ4:*** *Which components contribute to CIG's improvements?*
- ***RQ5:*** *How does CIG turn outcomes into revisable knowledge?*

Table 1: Held-out test performance of optimized Harnesses under the HarnessFix evaluation protocol. Task and optimization agents use the GPT-5 mini configuration (OpenAI, 2025). Entries are test task-completion rates (%), averaged over three runs. Published rows reproduce HarnessFix's reported results, and our runs use the aligned evaluation setup. Full History is our baseline, matched to CIG in initialization and evaluation allowance. Parentheses in the CIG row show percentage-point gains over the reproduced initial Harness. Reporting conventions are given in Appendix A.8.

Method	SWE-V	AppWorld	TB2.0-V	Macro avg.
Published HarnessFix results (3-run means)				
Initial Harness (Chen et al., 2026b)	45.3	36.7	17.6	33.2
GEPA (Agrawal et al., 2026)	46.7	37.4	19.6	34.6
SCOPE (Pei et al., 2025)	48.3	38.9	20.6	35.9
ReCreate (Hao et al., 2026)	51.7	39.3	21.6	37.5
Meta-Harness (Lee et al., 2026b)	54.7	40.4	23.5	39.5
HarnessFix (Chen et al., 2026b)	**57.3**	**43.0**	**26.5**	**42.3**
Our runs (3-run means)				
Reproduced Initial Harness	45.7	35.6	17.6	33.0
Full History (our implementation)	56.7	40.0	23.5	40.1
CIG (Ours)	**60.0** (+14.3)	**44.4** (+8.8)	**29.4** (+11.8)	**44.6** (+11.6)

4.1 Evaluation Protocol

Benchmarks. Our classification studies use three tasks from Meta-Harness (Lee et al., 2026b): USPTO-50k reaction classification (Schneider et al., 2016), diagnosis from symptom descriptions using Gretel's adaptation of Symptom2Disease (Gretel AI, 2023), and LawBench criminal-charge classification (Fei et al., 2024). Our agent-Harness evaluation covers SWE-bench Verified repository repair (Jimenez et al., 2024), stateful application interaction in AppWorld (Trivedi et al., 2024), and terminal-based tasks in Terminal-Bench (Merrill et al., 2026). Table 1 compares CIG with published results under the HarnessFix evaluation protocol. Our runs reuse the benchmark environments and evaluation setup, with aligned initial Harnesses, split sizes, task and optimization models, and three-run reporting. Sampling, data splits, search schedules, and budgets follow the HarnessFix protocol.

Protocol. Within each controlled comparison, methods share initialization, access to raw observations and evaluation history, and the evaluation allowance. The agent-benchmark comparison uses the HarnessFix protocol described above, while the subsequent matched-control studies use eight candidate evaluations. Candidate selection depends only on validation performance. Held-out test outcomes are revealed only after search and selection are complete. Our primary measure is the held-out test performance of the validation-selected Harness. Selection regret is the difference between the test oracle and the test score of the validation-selected Harness. Search curves summarize progress during discovery.

Comparison methods. We compare CIG with (1) prompt and scaffold optimization methods and (2) Harness code optimization methods. In the first group, GEPA (Agrawal et al., 2026) and SCOPE (Pei et al., 2025) evolve prompts through reflective or memory-guided feedback, while ReCreate (Hao et al., 2026) converts experience into reusable agent scaffolds. In the second group, Meta-Harness (Lee et al., 2026b) searches complete Harness code from prior artifacts, whereas HarnessFix (Chen et al., 2026b) diagnoses failed trajectories and applies scoped repairs. To evaluate CIG as an integrated discovery method, we additionally implement **Proposer-Centric Full-History Search (Full History)** as a graph-free control. It gives one proposer the complete permitted history and a skill governing history inspection and candidate implementation while matching CIG in initialization, observable evidence, and candidate-evaluation budget. This comparison tests the complete CIG workflow against proposer-centric search.

4.2 **RQ1**: How effective is CIG compared with existing approaches?

Table 1 reports higher CIG scores than both our Full History control and the published baselines across repository repair, application interaction, and terminal tasks. The gains over Full History across these environments support the complete method beyond a single task interface. The following controlled studies examine model dependence, search allocation, and component contributions.

Table 2: Robustness across task solvers on the three classification tasks. Entries are mean test scores (%) and selection regret under eight candidate evaluations. Δ reports test gains over Full History in percentage points. Wide uses two waves of four candidates, and Deep uses four waves of two.

	Val-selected Test ↑			Δ Test ↑		Selection regret ↓		
Solver	Full History	Wide	Deep	Wide	Deep	Full History	Wide	Deep
Qwen3.5-Plus	51.5	**60.6**	59.7	+9.1	+8.2	7.1	0.0	0.0
Qwen3.7-Flash	41.3	**53.8**	49.4	+12.5	+8.2	0.0	0.0	2.0
DeepSeek-v4-Flash	51.3	**55.2**	51.7	+3.9	+0.4	0.0	0.0	3.9
GLM-5.2-Fast-Preview	50.4	**54.7**	50.3	+4.3	-0.1	0.0	0.0	2.5
MiniMax-M2.5	45.4	**48.9**	47.8	+3.5	+2.4	0.0	0.2	0.1

4.3 RQ2: How do CIG's search schedules perform across solvers?

We fix GPT-5.4 as the proposer for all three classification tasks: USPTO-50k, Symptom2Disease, and LawBench. To assess whether the gains depend on the task model, we vary the solver used to execute each Harness. We use five solvers, including two models from the Qwen family: Qwen3.5-Plus (Alibaba Group, 2026) and Qwen3.7-Flash (Qwen Team, 2026). The other three are DeepSeek-V4-Flash (DeepSeek-AI, 2026), GLM-5.2-Fast-Preview (Alibaba Cloud, 2026), and MiniMax-M2.5 (MiniMax, 2026). We compare Full History with two CIG search settings. CIG (Wide) uses two waves of four validated candidates, while CIG (Deep) uses four waves of two validated candidates. Full History shares Deep's four-wave, two-candidate schedule. Table 2 first compares Deep with Full History under this matched schedule. Deep improves on Full History for four of the five solvers, with a small decrease for GLM, supporting the integrated CIG workflow beyond search width. Within CIG, Wide outperforms Deep for every solver and has lower mean selection regret. This favors broader candidate portfolios with fewer feedback rounds in the tested settings. RQ4 examines component contributions under matched scheduling and staged calls.

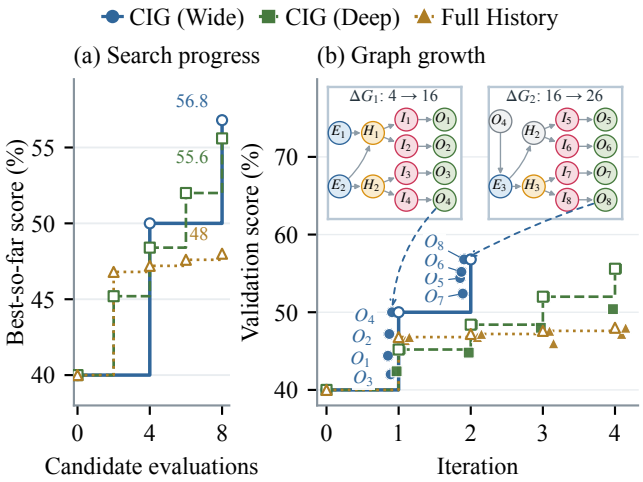


Figure 3: Search progress and graph growth on LawBench using post-search validation re-evaluation means. (a) Best-so-far scores at completed wave boundaries. (b) Filled markers show individual candidate scores, linked to Outcome nodes; hollow markers and step curves show the best-so-far score, as in (a). Insets show node counts before and after updates, with gray marking retained nodes and colors marking additions. Iteration zero denotes the initial graph state before the first round of Harness optimization.

Figure 3 compares candidates discovered under the same eight-candidate allowance using post-search re-evaluation means. Full History shows modest later gains, whereas both CIG schedules discover stronger candidates as search progresses. Deep exceeds Full History's final re-evaluated best score after two waves and improves further, while Wide reaches the highest final score. The insets show how later experiments build on shared evidence and retained hypotheses within this evaluated search horizon, connecting results across discovery rounds.

Table 3: Robustness across proposers on SWE-bench Verified. Scores are mean test pass@1 ± sample standard deviation of the validation-selected Harness. Δ is the difference in means.

Proposer	Full History	CIG (Wide)	Δ
GPT-5.5	58.6 ± 0.5	**61.0** ± 0.0	+2.4
GPT-5.6 Sol	59.0 ± 1.0	**64.0** ± 1.0	+5.0
GLM-5.3	58.2 ± 1.1	**65.6** ± 0.5	+7.4
Kimi K3	56.6 ± 0.5	**65.4** ± 0.5	+8.8
DeepSeek-V4-Pro	59.0 ± 0.7	**63.0** ± 1.0	+4.0

Figure 4 shows that the aggregate advantage extends across the three classification tasks: gains averaged across solvers are positive on each task, and the task-level comparisons are positive across all five solvers. Their magnitudes vary, but the consistent direction supports robustness to solver choice across all three classification tasks and five evaluated solver models.

4.4 RQ3: Does CIG remain effective across proposer models?

We vary the proposer on SWE-bench Verified, where Harness design coordinates repository navigation, context construction, tool use, and repair strategy. We fix Qwen3.5-Plus (Alibaba Group, 2026) as the task solver and compare Full History with CIG (Wide) using GPT-5.5 (OpenAI, 2026a), GPT-5.6 Sol (OpenAI, 2026b), GLM-5.3 (Z.ai, 2026), Kimi K3 (Kimi Team et al., 2026), and DeepSeek-V4-Pro (DeepSeek-AI, 2026) as proposers. The Harness initialization, task split, selection rule, and eight-candidate evaluation budget remain fixed. Table 3 shows a positive gain with every proposer, indicating that the benefit is not confined to a single proposal model. The variation in gain motivates the matched structural comparisons under a common discovery workflow.

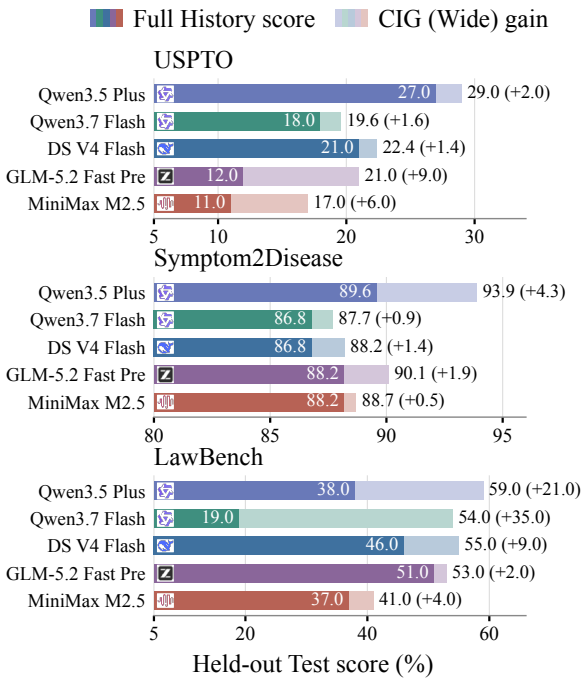


Figure 4: Mean task-level test performance. Labels show CIG (Wide) scores and gains over Full History in percentage points. USPTO denotes USPTO-50k, DS denotes DeepSeek, and Pre denotes Preview. Each panel uses its own linear axis with a nonzero origin. Bar lengths are comparable within each panel, but not across panels with different axis ranges.

4.5 RQ4: Which components contribute to CIG's improvements?

We next examine which parts of CIG account for its gains. Table 4 combines cumulative construction with targeted alternatives to distinguish the benefits of organizing experimental history, linking records, revising assessments, and using that state to select plans. Holding the discovery workflow fixed allows these contrasts to assess how the components contribute to final Harness quality and search progress under the same evaluation allowance.

State and relation semantics. M0 exposes raw history through staged calls. M1 organizes it into typed E–H–I–O records, and M2 adds their logical relations. Both improve test and AUC for both proposers. B1 retains M2's structure but shuffles its relations and loses performance, supporting the value of meaningful dependencies beyond typed records alone.

Persistence and revision. M3 replaces M2's round-local state with persistent, revisable assessments. B2 preserves persistence but permits only append-only records. M3 outperforms both M2 and B2, indicating that carrying knowledge forward and updating its interpretation make complementary contributions along this construction path.

Using relational state for selection. M4 adds graph-guided plan selection to M3 and achieves the highest test and AUC with both proposers. B3 retains persistent, revisable records and local provenance but removes explicit cross-record relations, so selection uses the records directly. Its performance lies between M0 and M4: revisable notes retain substantial value, while cross-record relations provide a further benefit. Together, these comparisons suggest a practical design principle: maintain both accumulated experience and its evolving interpretation as inputs to Harness optimization. Revisable assessments track what prior results currently support, while explicit relations connect those judgments to subsequent planning and plan selection. Appendices A.3 and A.4 provide full control definitions and run-to-run standard deviations.

Table 4: Structural contributions to CIG. Starting from Full History, cumulative variants introduce typed state, relations, and revision before full CIG, while alternative designs (blue) test how these components are used. The results support their complementary value for final Harness quality and search progress under a matched workflow. Appendix A.3 provides exact variant definitions and reporting conventions.

	Cumulative construction				Proposer A		Proposer B	
Variant	State	Relations	Update	Selection	Test ↑	AUC ↑	Test ↑	AUC ↑
M0 (Full History)	Raw	None	Reset	Indep.	58.0	58.0	56.4	56.8
M1 (+ Typed state)	Typed	None	Reset	Indep.	59.6 (+1.6)	59.1	57.4 (+1.0)	57.9
M2 (+ Relations)	Typed	Correct	Reset	Indep.	60.8 (+1.2)	60.5	58.8 (+1.4)	59.2
↪ *B1 (Shuffled)*	Typed	Shuffled	Reset	Indep.	59.0 (-1.8)	59.2	57.4 (-1.4)	58.1
M3 (+ Revision)	Typed	Correct	Revise	Indep.	62.2 (+1.4)	61.8	59.6 (+0.8)	60.4
↪ *B2 (Append-only)*	Typed	Correct	Append-only	Indep.	61.0 (-1.2)	60.6	58.8 (-0.8)	59.6
M4 (Full CIG)	Typed	Correct	Revise	Graph	**63.6** (+1.4)	**63.0**	**61.4** (+1.8)	**61.5**
↪ *B3 (Local only)*	Typed	Local only	Revise	Records	62.0 (-1.6)	61.0	59.2 (-2.2)	59.4

4.6 RQ5: How does CIG turn outcomes into revisable knowledge?

The following trace shows how CIG connects a diagnosed weakness to a testable intervention and a revised assessment. The node excerpts preserve the reasoning chain across two discovery waves.

Evidence-linked intervention and reassessment

$$E_2 \xrightarrow{\text{grounds}} H_2 \xrightarrow{\text{tested by}} I_2 \xrightarrow{\text{yields}} O_2$$

EVIDENCE E_2 **Identify a trade-off.**
Similarity retrieval improves USPTO and Symptom2Disease but lowers LawBench from 60 to 46.

HYPOTHESIS H_2 **Register a testable explanation.**
Restoring demonstration coverage should recover LawBench while preserving the other gains and exceeding the parent's aggregate score of 58.00.

INTERVENTION I_2 **Test a scoped change.**
Keep query-conditioned retrieval, allocate 60% of the prompt budget to nearest neighbors, and backfill the remaining slots using answer-space and token-coverage novelty.

OUTCOME O_2 **Separate effects from the joint verdict.**
LawBench recovers to 52 and Symptom2Disease stays at 88, but USPTO falls to 33.33. Aggregate performance falls to 57.78. The joint target fails, while the predicted LawBench direction remains consistent with the observation.

$$O_2 \xrightarrow{\text{updates assessment}} H_2$$

Local prediction remains consistent. Joint target fails. Coverage explanation remains open.

The linked state preserves both the local recovery and the failed joint target, leaving a scoped assessment that guides subsequent planning. Appendix A.9 provides the detailed trace and a state-intervention comparison that contrasts the plans and Harnesses produced when hypothesis assessments are updated using evaluated outcomes or retained unchanged.

5 Conclusion

CIG shifts Harness optimization from proposer-centric to graph-governed discovery through persistent improvement state, allowing proposers to reuse accumulated experience and updated assessments rather than reconstruct them from raw history. Experiments show stronger Harnesses across tasks and robustness to solver and proposer choice. Matched ablations support explicit relations beyond revisable records, while a state-intervention study shows how updated assessments guide subsequent plans. Together, these findings support maintaining accumulated experience and its evolving interpretation as persistent improvement state to guide further Harness optimization. Our evaluation focuses on Harness optimization within individual searches, leaving the transfer and adaptation of persistent improvement state across tasks and models as a promising direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing and to assist with the implementation of experimental code. All AI-assisted text and code were reviewed by the authors, and the experimental code was tested before use. The authors take responsibility for the final content of this work, including material produced with the aid of generative AI.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. *International Conference on Learning Representations*, 2026.

Alibaba Cloud. GLM-5.2-Fast-Preview, 2026. URL https://www.alibabacloud.com/help/en/model-studio/glm-5-2-fast.

Alibaba Group. Alibaba open-sources Qwen3.5, a natively multimodal model built for high-efficiency inference, 2026. URL https://www.alibabagroup.com/en-US/document-1960233590314762240.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Conference on Neural Information Processing Systems*, 2020.

Jiashun Chen, Yirong Mao, and Wenhui Que. EviGraph: Towards verifiable evidence construction for information-seeking agents. *arXiv preprint arXiv:2608.24667*, 2026a.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, Haiming Zheng, and Qing Wang. From failed trajectories to reliable LLM agents: Diagnosing and repairing harness flaws. *arXiv preprint arXiv:2606.06324*, 2026b.

Tingyang Chen, Shuo Lu, Kang Zhao, Weicheng Meng, Hanlin Teng, Tianhao Li, Chao Li, Xule Liu, Jian Liang, Zhizhong Zhang, Yuan Xie, Heng Qu, Kun Shao, and Jian Luan. HarnessX: A composable, adaptive, and evolvable agent harness foundry. *arXiv preprint arXiv:2606.14249*, 2026c.

DeepSeek-AI. DeepSeek V4 model card, 2026. URL https://fe-static.deepseek.com/chat/transparency/deepseek-V4-model-card-EN.pdf.

Yuetian Du, Yucheng Wang, He Xu, Jiexu Xu, Shanwen Tan, Bing Zhao, Boyu Yang, Zhijie Xu, Ming Kong, Hu Wei, et al. Living-harness is an interactive-agent evolver. *arXiv preprint arXiv:2607.26598*, 2026.

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Alan Huang, Songyang Zhang, Kai Chen, Zhixin Yin, Zongwen Shen, Jidong Ge, and Vincent Ng. LawBench: Benchmarking legal knowledge of large language models. *Conference on Empirical Methods in Natural Language Processing*, 2024.

Gretel AI. Symptom to diagnosis dataset, 2023. URL https://huggingface.co/datasets/gretelai/symptom_to_diagnosis.

Zhezheng Hao, Hong Wang, Jian Luo, Jianqing Zhang, Yuyan Zhou, Qiang Lin, Can Wang, Hande Dong, and Jiawei Chen. ReCreate: Reasoning and creating domain agents driven by experience. *Annual Meeting of the Association for Computational Linguistics*, 2026.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *International Conference on Learning Representations*, 2025.

Yoichi Ishibashi, Taro Yano, and Masafumi Oyamada. Effective harness engineering for algorithm discovery with coding agents. *arXiv preprint arXiv:2605.15221*, 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? *International Conference on Learning Representations*, 2024.

Kimi Team et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Hyunin Lee, Jinglue Xu, Jeffrey Seely, Donghyun Lee, Matei Zaharia, and Yujin Tang. Recursive harness self-improvement. *arXiv preprint arXiv:2607.15524*, 2026a.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026b.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026.

Ryan Lopopolo. Harness engineering: Leveraging Codex in an agent-first world, 2026. URL https://openai.com/index/harness-engineering/.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. AutoHarness: Improving LLM agents by automatically synthesizing a code harness. *arXiv preprint arXiv:2603.03329*, 2026.

Yaxin Luo, Haobin Jiang, Jialv Zou, Xu Huang, Wenhao Yan, Haodong Li, Zhengrong Yue, Jing Li, Xiaofu Chen, Xiaohan Zhao, Jiacheng Liu, Jiacheng Cui, Zhiqiang Shen, and Xiaotong Li. AutoDesign: Meta-harness optimization for long-horizon agentic design. *arXiv preprint arXiv:2608.13560*, 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

MiniMax. MiniMax M2.5: Built for real-world productivity, 2026. URL https://www.minimax.io/news/minimax-m25.

OpenAI. GPT-5 mini model, 2025. URL https://developers.openai.com/api/docs/models/gpt-5-mini.

OpenAI. Introducing GPT-5.5, 2026a. URL https://openai.com/index/introducing-gpt-5-5/.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, 2026b. URL https://openai.com/index/gpt-5-6/.

OpenClaw. OpenClaw: Your assistant, on your devices, in your chats, 2026. URL https://github.com/openclaw/openclaw.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Conference on Neural Information Processing Systems*, 2022.

Sungho Park, Wonjoong Kim, Rongyuan Tan, Jue Zhang, Wook-Shin Han, Pengfei Gao, Chanyoung Park, Yongqiang Yao, Rao Fu, Elsie Nallipogu, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. AutoSaddler: Automatic harness optimization with durable updates from agent execution traces. *arXiv preprint arXiv:2608.23041*, 2026.

Zehua Pei, Hui-Ling Zhen, Shixiong Kai, Sinno Jialin Pan, Yunhe Wang, Mingxuan Yuan, and Bei Yu. SCOPE: Prompt evolution for enhancing agent effectiveness. *arXiv preprint arXiv:2512.15374*, 2025.

Qwen Team. Qwen3.7-Flash model release, 2026. URL https://docs.qwencloud.com/changelog/models.

Zhenjiang Ren, Ruiji Li, Xujing Zhang, Ziliang Pang, Shuo Ren, and Jiajun Zhang. EviGraph: Evidence-guided autonomous research agents. *arXiv preprint arXiv:2608.04738*, 2026.

Nadine Schneider, Nikolaus Stiefl, and Gregory A. Landrum. What's what: The (nearly) definitive guide to reaction role assignment. *Journal of Chemical Information and Modeling*, 2016.

Thariq Shihipar. A harness for every task: Dynamic workflows in Claude Code, 2026. URL https://claude.com/blog/a-harness-for-every-task-dynamic-workflows-in-claude-code.

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. *Annual Meeting of the Association for Computational Linguistics*, 2024.

Yike Wang, Huaisheng Zhu, Zhengyu Hu, Yige Yuan, Zhengyu Chen, Shakti Senthil, Hannaneh Hajishirzi, Yulia Tsvetkov, Pradeep Dasigi, and Teng Xiao. Rethinking the evaluation of harness evolution for agents. *arXiv preprint arXiv:2607.12227*, 2026a.

Zefeng Wang, Minxi Yan, Jinhe Bi, Sikuan Yan, Volker Tresp, and Yunpu Ma. MetaSkill-Evolve: Recursive self-improvement of LLM agents via two-timescale meta-skill evolution. *arXiv preprint arXiv:2607.05297*, 2026b.

Z.ai. GLM-5.3: Overview, 2026. URL https://docs.z.ai/guides/llm/glm-5.3.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. *Conference on Language Modeling*, 2024.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026a.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025a.

Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. Hyperagents. *arXiv preprint arXiv:2603.19461*, 2026b.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. *International Conference on Learning Representations*, 2025b.

Yifan Zhang, Yutong Dai, Juntao Tan, Luyu Yang, Rishi Mullur, Thai Hoang, Zhiyuan Hu, James Zhu, Phil Mui, Silvio Savarese, Ran Xu, and Zeyuan Chen. DarwinX: Evolving agent harnesses through natural selection. *arXiv preprint arXiv:2608.07545*, 2026c.

Tailin Zhou. Hierarchical self-improvement: A framework for task-specific evolvable agent harnesses. *arXiv preprint arXiv:2608.08466*, 2026.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. *International Conference on Machine Learning*, 2024.

APPENDIX

TABLE OF CONTENTS

A Additional Method and Experimental Details **13**

A.1 Graph Schema and Transition Rules 13

A.2 Related Methods and Representation Choices 13

A.3 Structural Controls and Interpretable Contrasts 14

A.4 Run-to-Run Variability of Structural Ablations 16

A.5 Selection Eligibility and Verdict Semantics 16

A.6 Operation Interfaces and Revision Details 17

A.7 Proposer Token Usage 19

A.8 Experimental Configuration and Reporting 19

A.9 Detailed Case Study 20

A ADDITIONAL METHOD AND EXPERIMENTAL DETAILS

A.1 GRAPH SCHEMA AND TRANSITION RULES

The graph contains four independently addressable node types: Evidence, Hypothesis, Intervention, and Outcome. Evidence retains the source observation and its task and Harness provenance. A Hypothesis has its own identity, explanatory claim, applicable scope, and revisable assessment. Plans reference these nodes and specify how an Intervention will test the linked Hypothesis. Outcomes retain the external measurements and connect them to the executed Intervention. Harness references record the parent and resulting candidate without replacing the four node types.

Relations express which Evidence supports or challenges a Hypothesis, which Intervention tests it, which Outcome evaluates that Intervention, and which earlier assessment the result revises. Multiple plans can test the same Hypothesis, and one Outcome can inform several linked assessments. When an Outcome supplies Evidence for a later round, subsequent reasoning references the same measurement rather than counting it as a new independent observation.

Observe produces evidence-linked claims, uncertainty, unresolved questions, and cross-branch conflicts. Plan proposes interventions with parents, preserved mechanisms, predicted effects, falsification conditions, and resource contracts. PlanSelect scores each plan on seven ordinal criteria and compares portfolio redundancy and complementarity. Appendix A.6 specifies the selection interface. Implement realizes the selected changes, while external evaluation produces the measured outcomes. Update appends provenance and deltas and refreshes the decision view for subsequent calls. Related hypotheses can remain distinct, with shared evidence made explicit through their relations. Search ends when the candidate-evaluation allowance is exhausted, which bounds graph growth in these experiments.

A.2 RELATED METHODS AND REPRESENTATION CHOICES

Table 5 compares how related methods retain experience, represent predictions, and organize optimization state.

Representation and update targets. ADAS and AFlow retain program archives and search trees, while AHE and HarnessFix organize predictions and repair experience. Among graph-based systems, AutoSaddler records patch lineage and lessons, HSI also evolves the optimizer, and LivingHarness updates procedural behavior. The two EviGraph systems organize research claims and information-seeking evidence, respectively. **CIG** makes revisable hypotheses the interface between

Table 5: State representations in related methods. Columns indicate persistent search or experience memory, explicit predictions or hypotheses, and graph-structured state used in search or adaptation. The final column contrasts each method's organizing principle with CIG. ✓ denotes an explicitly described feature, and – denotes a feature not established by the cited description.

Method	Persistent memory	Explicit predictions	Graph state	Compared with CIG
ADAS (Hu et al., 2025)	✓	–	–	Organizes search around archived programs, while CIG organizes it around revisable hypotheses.
AFlow (Zhang et al., 2025b)	✓	–	✓	Searches a workflow tree, while CIG links evidence and hypotheses across interventions.
Meta-Harness (Lee et al., 2026b)	✓	–	–	A proposer interprets stored history, while CIG exposes linked assessments to scoped operations.
AHE (Lin et al., 2026)	✓	✓	–	Checks edit-level predictions, while CIG carries their assessments into a persistent hypothesis graph.
HarnessFix (Chen et al., 2026b)	✓	✓	–	Organizes repair memory, while CIG uses linked hypothesis assessments to select experiments.
Living-Harness (Du et al., 2026)	✓	–	✓	Updates procedural behavior, while CIG updates the optimizer's experimental reasoning state.
HSI (Zhou, 2026)	✓	✓	✓	Evolves the Harness and evolver, while CIG revises hypothesis state through fixed typed operations.
AutoSaddler (Park et al., 2026)	✓	–	✓	Centers its graph on patch lineage and lessons, while CIG centers it on revisable hypotheses.
EviGraph (research) (Ren et al., 2026)	✓	✓	✓	Links evidence to research claims, while CIG links it to evaluated Harness interventions.
EviGraph (information seeking) (Chen et al., 2026a)	✓	–	✓	Grounds information seeking in evidence, while CIG uses evidence to guide Harness experiments.
CIG	✓	✓	✓	Linked, outcome-revised hypotheses govern subsequent Harness experiments.

evidence, executable Harness interventions, and subsequent plan selection. Its distinction is the use of these linked assessments to govern discovery, rather than graph storage alone. Table 4 isolates representation, revision, and selection mechanisms.

Evaluation methodology. Rethinking Harness Evolution compares automatic evolution with simple search under matched feedback and inference budgets and evaluates held-out generalization (Wang et al., 2026a). Table 1 provides a comparison under the HarnessFix protocol. Our subsequent matched controls address method behavior under a fixed candidate-evaluation allowance, and Table 4 also matches the staged reasoning schedule.

A.3 Structural Controls and Interpretable Contrasts

Table 4 reports a cumulative construction and diagnostic branches. The definitions below specify what changes in each comparison.

Reporting conventions. Proposers A and B denote GPT-5.6 Sol and DeepSeek-V4-Pro, respectively, with Qwen3.7-Flash (Qwen Team, 2026) as solver on SWE-bench Verified. Values are five-run means. Test is validation-selected held-out pass@1 (%), and AUC summarizes the best-so-far validation curve. Parentheses report test differences in percentage points: M1–M4 versus the preceding main variant, and B1, B2, and B3 versus M2, M3, and M4, respectively.

Matched protocol. M0 is a staged raw-history control for structural attribution. It uses the same staged calls, round schedule, portfolio size, and final Harness selection rule as M1–M4. Across the complete matrix, we also fix the initialization, solver, benchmarks, candidate-evaluation budget, and access to eligible raw history. The variants differ only in the state interface, relation semantics, cross-round update, or selection access named in the table. Candidate generation and selection use validation only, and held-out test remains unavailable until the selections are frozen.

M0 Full History. Supply the staged operations with the visible raw discovery history. No compiled E–H–I–O state or logic relation is carried between rounds.

M1 Typed State. Compile the same eligible observations into E–H–I–O records without explicit logic relations. The compiled state is reset before the next round, so this step tests the typed representation rather than persistence.

M2 Logic Relations. Add typed relations that connect Evidence, Hypotheses, Interventions, and Outcomes within the current round. The graph is still reset before the next round, and plan selection does not directly traverse the relations.

B1 Shuffled Relations. Replace the correct relations of M2 with a recorded type-compatible shuffle. Node content, source attribution, evaluation identity, and executable lineage remain unchanged. The comparison with M2 tests relation semantics rather than edge count.

M3 Persistent Revision. Carry the relational state across rounds. New Outcomes can revise the status or scope of earlier Hypotheses and can redirect subsequent Interventions. Selection remains independent of direct graph access.

B2 Append-only Persistence. Carry the M2 graph across rounds but only append records. Earlier conclusions are not revised when later Outcomes conflict with them. The comparison with M3 separates durable storage from revision-aware experimental state.

M4 Full CIG. Give plan selection access to the current revision-aware graph so the selected portfolio can target unresolved hypotheses, competing explanations, and supported directions.

B3 Without Cross-record Relations. This control starts from M4 and preserves typed E–H–I–O records, cross-round persistence, hypothesis status and scope revision, and access to the resulting records during selection. It retains the same staged calls, plan and candidate counts, initialization, evaluation protocol, and raw-history access. Each experiment retains its evidence sources, tested hypothesis, intervention, candidate, and outcome references, together with revision audit records. The control does not construct, persist, or expose cross-record semantic relations such as support and contradiction across experiments. These relations are not copied into summaries or alternative adjacency fields. Observe, Plan, and PlanSelect instead read the persistent records and their current assessments. The proposer can infer connections from their contents, and Update can revise earlier assessments from new outcomes without storing a semantic edge network. Thus B3 removes explicit relational organization, not the ability to reason or revise. Unlike M1 it preserves state across rounds, unlike B1 it removes rather than corrupts semantic relations, and unlike B2 it retains revision. The contrast between B3 and M4 measures the contribution of explicit cross-record relations in the otherwise complete workflow. Relative to M4, B3 lowers mean test by 1.6 and 2.2 percentage points and search AUC by 2.0 and 2.1 for GPT-5.6 Sol and DeepSeek-V4-Pro, respectively. All reported differences are differences of run means.

Metrics and Mean Contrasts. Let $b_{p,v,0}$ be the validation score of the shared initial Harness, and let $b_{p,v,w}$ be the best-so-far validation score after all candidates in wave w have been evaluated. For the two-wave structural-ablation protocol, we use the normalized trapezoidal area

$$\mathrm{AUC}_p(v) = \frac{1}{W} \sum_{w=1}^{W} \frac{b_{p,v,w-1} + b_{p,v,w}}{2}, \qquad W = 2. \tag{7}$$

This value remains on the validation-score scale and combines the timing and magnitude of improvements. Wave-level checkpoints prevent the arbitrary completion order of candidates within a parallel wave from changing the metric.

Let $Y_p(v)$ denote either the repeat-mean validation-selected test or search AUC for variant v under proposer p. The main-path effects are

$$\Delta_k^p = Y_p(\mathrm{M}k) - Y_p(\mathrm{M}(k-1)), \qquad k \in \{1,2,3,4\}. \tag{8}$$

The branch contrasts are $Y_p(\mathrm{B1}) - Y_p(\mathrm{M2})$, $Y_p(\mathrm{B2}) - Y_p(\mathrm{M3})$, and $Y_p(\mathrm{B3}) - Y_p(\mathrm{M4})$. The difference between M4 and M0 measures the overall gain of the complete CIG design. Each table cell reports the absolute metric, while the parenthesized value reports the difference in means from the corresponding reference.

Interpretation and proposer dependence. M2 improving over M1 while B1 falls below M2 supports the value of correct relation semantics. M3 improving over M2 while B2 falls below M3 supports revision-aware persistence rather than passive accumulation. M4 improving over M3 supports direct graph use during selection. Search AUC and test are interpreted separately because a component can accelerate discovery without changing the final selected Harness, or improve the endpoint without helping early search. Results are reported separately for GPT-5.6 Sol and DeepSeek-V4-Pro. We compare the signs and magnitudes of the mean contrasts across proposers. Each difference measures a component's contribution relative to its specified reference configuration.

A.4 Run-to-Run Variability of Structural Ablations

Table 6 accompanies Table 4, retaining the two proposer groups and reporting test means and sample standard deviations across five runs.

Table 6: Companion statistics for Table 4. Test scores are mean ± sample standard deviation in percentage points, grouped by proposer.

Variant	GPT-5.6 Sol	DeepSeek-V4-Pro
M0 (Full History)	58.0 ± 0.0	56.4 ± 0.9
M1 (+ Typed State)	59.6 ± 0.5	57.4 ± 0.5
M2 (+ Logic Relations)	60.8 ± 0.8	58.8 ± 0.4
B1 (Shuffled Relations)	59.0 ± 0.0	57.4 ± 0.5
M3 (+ Revision-aware Persistence)	62.2 ± 0.8	59.6 ± 0.9
B2 (Append-only Persistence)	61.0 ± 0.0	58.8 ± 0.4
M4 (Full CIG)	63.6 ± 1.1	61.4 ± 0.9
B3 (w/o Cross-record Relations)	62.0 ± 0.0	59.2 ± 0.8

A.5 Selection Eligibility and Verdict Semantics

Candidate legality, intervention fidelity, and hypothesis support answer different questions. Legality concerns executable artifacts, allowed dependencies, and resource contracts. Fidelity asks whether the proposed modification was realized. Support asks whether the measured effects agree with the prediction. Final selection uses eligible measured performance, independently of whether the candidate's motivating explanation is supported.

For score-based observations, the numerical outcome layer first checks improvement over the prior incumbent and assigns `globally_improving`. Otherwise it compares against the best scored parent and assigns `locally_supported` for an increase, `refuted` for a decrease, or `uncertain` otherwise. These labels summarize measured score changes. The reasoning layer reads these labels together with the measured effects and registered predictions to revise the current hypothesis assessment. Revision changes the interpretation available to later calls, not the recorded scores or their provenance. Task-level scores support task-level assessments. Sample-level transitions and execution-fidelity judgments are included only when the corresponding evidence is available.

At the hypothesis level, support means that the measured result agrees with the stated prediction over its specified scope. Local support applies when only a narrower task or condition agrees. Refutation

requires an observed violation of the stated falsification condition. Inconclusive or unavailable evidence leaves the assessment uncertain. Scope refinement retains the original claim and records the narrower conditions under which it remains consistent with observations. Agreement in score alone does not establish why an intervention worked.

A failed aggregate target therefore refutes that target for the tested implementation and setting, not every constituent mechanism. Task-level scores can resolve directional trade-offs. Sample-level corrected/regressed claims require paired traces and cannot be inferred from aggregate scores. Infrastructure failures remain distinct from measured Harness failures.

A.6 Operation Interfaces and Revision Details

The operational interface separates factual observations, revisable assessments, and selection decisions. The role specifications below describe the inputs, reasoning operations, and outputs of each stage.

Observe and reassess. Inputs are the current graph, permitted evaluation history, and predictions linked to the evaluated interventions. The permitted history includes candidate artifacts, validation scores, reports, and available execution traces. The observer accesses these records through source references as needed, within the configured context and computation limits. The output identifies grounded evidence, current hypothesis assessments, unresolved questions, and conflicts. Reassessment compares the predicted effects with the available observations, distinguishes a failed joint target from a partially supported local prediction, and preserves references to the relevant experiment. Existing observations remain unchanged when their interpretation is revised.

Plan and select. Plans name their parent Harnesses, supporting evidence, proposed changes, predicted effects, and falsification conditions. Each prediction specifies the expected direction and relevant task or behavior, while its falsification condition states what observed result would count against it. In the Wide setting, the planner generates eight plans and the selector chooses four per wave. Planning and candidate evaluation are separate, so only the four implemented plans consume candidate evaluations. Selection reads the plans together with the current relational state. It uses the seven criteria described in Section 3 to compare plans and portfolio redundancy, including the relevance of each plan to unresolved or revised hypotheses. Redundancy concerns overlapping target mechanisms and expected effects, while complementarity concerns distinct failure modes, competing explanations, or alternative mechanisms. The portfolio contains an evidence-grounded refinement and a branch or restart and does not allocate every slot to one mechanism family. Selection is a proposer judgment under these constraints, not a deterministic maximization of a weighted score. It returns selected plans without rewriting them, together with reasons for selection and rejection.

Commit and carry forward. The evaluated candidate, parent-relative effects, and numerical verdict are retained as an experimental record. Their dependencies connect the result to its motivating prediction. Comparison first separates whether the intended change was executed, whether the measured direction agrees with the prediction, and whether the evidence supports the proposed explanation. Missing execution evidence is not inferred from a score increase. The revised assessment forms part of the state read by subsequent observation, planning, and selection. Append-only persistence retains the experimental records without revising earlier assessments, whereas revision-aware persistence allows the interpretation guiding later decisions to change as evidence accumulates.

Assessment example. In Appendix A.9, the prediction is to recover LawBench without losing the other gains and to exceed 58.00 aggregate performance. The recorded outcome is LawBench $+6.00$, USPTO -6.67, unchanged Symptom2Disease, and aggregate -0.22. Its numerical label is `refuted`. At the assessment level, the joint target failed, the local LawBench prediction remains consistent with the observation, and the coverage explanation remains unresolved. This illustrates the distinction between a numerical verdict and the scoped interpretation carried by revision.

A.6.1 Prompt Excerpts

The following prompt excerpts illustrate the method's role boundaries and core reasoning operations. The final box describes the falsifiable-hypothesis instructions for Full History.

Role-Specific Instructions

Shared context and execution boundaries

Use the current graph state and its source evidence to carry out the assigned reasoning operation. Consult the underlying records when a summary is insufficient, and keep measured facts distinct from their interpretations. Perform only the assigned role. Do not collapse observation, planning, selection, and implementation into a single step. The runner alone commits graph transitions and measured outcomes, updates the frontier, and manages held-out testing. Never invent or modify runner-grounded validation facts.

Observer

Separate high-confidence observations from inferred explanations. Preserve contradictory and low-frequency evidence. Do not design, rank, select, or implement candidates.

Selector

Select a portfolio, not merely the highest individual scores.

Implementer

The plan fixes the experimental question and primary intervention, but you retain freedom to choose the strongest faithful implementation.

Hypothesis Revision

Read the current hypothesis records together with their linked evidence, interventions, and newly available outcomes. For each existing hypothesis affected by a new outcome:

1. Reference its existing hypothesis ID. Preserve its original claim and previous assessment rather than replacing the historical record.

2. Compare the stated prediction with the observed outcome. Distinguish intervention fidelity, local effects, and the overall performance objective.

3. Propose a current assessment of supported, contradicted, or unresolved at the scope justified by the evidence. A failed joint objective does not by itself refute every local prediction or the broader explanation.

4. State any justified scope refinement and cite the supporting and conflicting evidence. Preserve uncertainty when the evidence cannot discriminate.

5. Explain the implications for subsequent experiments without generating, ranking, selecting, or implementing candidate plans.

Return a revision proposal containing hypothesis_id, previous_assessment, proposed_assessment, previous_scope, proposed_scope, evidence_ids, rationale, and unresolved_questions. Create a new hypothesis ID only for a substantively new claim and identify its relationship to the earlier claim. Do not alter measured outcomes. The runner validates references and commits revisions while preserving earlier versions and their provenance.

Falsifiable Hypotheses (Full History)

Formulate three falsifiable hypotheses targeting different mechanisms. For each hypothesis:

1. Cite the observed evidence in the available history. Distinguish measured facts from inferred explanations and retain conflicting observations.

2. State the explanatory claim, its applicable scope, and its uncertainty.

3. Identify the parent Harness and one primary intervention. State which mechanisms should remain unchanged.

4. Predict an observable effect and specify what result would contradict the prediction. Distinguish local predictions from the overall objective.

5. Explain why the experiment is informative given previous attempts. Revisit a failed direction only with a stated change or new evidence.

Do not invent measurements or treat an improved score as proof of an explanation. Use the available history to choose and implement the next candidates. Prototype the proposed mechanism before writing the final system.

A.7 Proposer Token Usage

Beyond candidate-evaluation counts, we examine how proposer computation translates into Harness quality. AutoSaddler (Park et al., 2026) relates higher optimizer-side cost per patch to improved search outcomes and more selective evaluation. Vesper (Ishibashi et al., 2026) shows that investing more computation per candidate can improve discovery under a common token budget. Following this budget-based perspective, we examine CIG with GPT-5.4 as proposer and Qwen3.7-Flash as solver on USPTO, Symptom2Disease, and LawBench. Table 7 compares complete Full History and CIG Wide searches, each evaluating eight candidates, and a CIG checkpoint within Full History's proposer token budget. The checkpoint is reconstructed from the search trajectory and includes all proposer operations and retries incurred up to that point. Among candidates with completed evaluations at the checkpoint, validation selects the Harness whose test score is reported. Proposer usage covers input and output tokens across discovery operations, separately from solver execution during candidate evaluation.

Table 7: Proposer token usage and validation-selected test performance in a representative configuration. Token ratios use the complete Full History run as reference.

Method / checkpoint	Tokens (M)	Ratio	Candidates	Test (%)
Full History (complete)	0.530	1.00×	8	41.3
CIG Wide (budget checkpoint)	0.530	1.00×	5	46.8
CIG Wide (complete)	0.820	1.55×	8	52.9

At the common budget of 0.530M proposer tokens, CIG reaches 46.8% test with five evaluated candidates, exceeding Full History's complete eight-candidate search by 5.5 percentage points. Continuing CIG to eight candidates raises test to 52.9% at 1.55 times the proposer token usage, a gain of 11.6 points over Full History. Thus, in this configuration, CIG achieves higher Harness quality both at the common proposer budget and with additional computation. This comparison highlights the importance of how proposer computation is organized, alongside how much is used. CIG connects locally scoped operations through persistent improvement state, so prior experimental dependencies and updated assessments guide subsequent proposals. The checkpoint and complete-run results support this workflow's effective use of a common proposer budget and its ability to obtain further gains with additional computation in this setting.

A.8 Experimental Configuration and Reporting

Code availability. We will release the code publicly upon acceptance.

Unless otherwise stated, our experimental results are means over five independent runs. All entries in Table 1, including our runs and the published baselines, are means over three runs. Within each controlled comparison, methods use the same task solver, data partitions, and initial Harness artifacts. Initialization is fixed before search and does not use held-out test feedback. Figure 3 shows individual searches on LawBench with GPT-5.4 as proposer and DeepSeek-V4-Flash as solver. Search evaluates each candidate once. For the figure, each Harness is re-evaluated five times on the fixed 50-task validation set, and the curve uses these means over candidates available at each completed wave. These post-search measurements do not guide discovery or candidate selection. The case studies also describe individual searches. Tables 3, 4, and 6 use the SWE-bench Verified 100/50/100 train/validation/test partition in Table 8, with a binary success result per task in each run, then report the mean across five runs. Delta columns subtract the corresponding run means.

Model configurations. The agent-benchmark comparison uses GPT-5 mini for both proposal and task execution. The solver study fixes GPT-5.4 as proposer and varies Qwen3.5-Plus, Qwen3.7-Flash, DeepSeek-V4-Flash, GLM-5.2-Fast-Preview, and MiniMax-M2.5. The proposer study fixes Qwen3.5-Plus as solver and varies GPT-5.5, GPT-5.6 Sol, GLM-5.3, Kimi K3, and DeepSeek-V4-Pro. Structural ablations use Qwen3.7-Flash as solver with GPT-5.6 Sol or DeepSeek-V4-Pro as proposer. Each model pairing remains fixed throughout its search and final evaluation.

Data partitions. Table 8 lists the classification partitions and the agent-benchmark partitions used for the HarnessFix-aligned comparison. Training examples are available for Harness construction, validation guides discovery and selection, and test is reserved for final evaluation. The Terminal-

Bench partition is reconstructed following the reference protocol using the public split sampler with seed 20260522.

Published comparison results. The published block of Table 1 uses the results reported by HarnessFix. Our block uses the benchmark environments and evaluation setup described by that protocol, including the benchmark-specific initial Harnesses listed below. The two blocks report the same task-completion metric and three-run mean. Within our block, CIG and Full History share the task partitions and initialization.

Table 8: Dataset partition sizes. Classification rows count examples, and agent-benchmark rows count tasks.

Benchmark	Train	Validation	Test
USPTO-50k	50	30	100
Symptom2Disease	200	50	212
LawBench	200	50	100
SWE-bench Verified (HarnessFix protocol)	100	50	100
AppWorld	90	45	90
Terminal-Bench 2.0 Verified	34	17	34

Initialization and search schedule. Classification search starts from the same no-memory and all-training-example few-shot Harnesses. Their frozen validation results are reused within a matched solver configuration, rather than reevaluated for every search method. Agent-benchmark methods start from the same benchmark-specific initial Harness: mini-swe-agent for SWE-bench Verified, simplified ReAct for AppWorld, and Terminus-2 for Terminal-Bench. The controlled solver, proposer, and structural studies use eight candidate evaluations. Full History and CIG (Deep) use four waves of two candidates, while CIG (Wide) uses two waves of four. The structural Full History control instead shares CIG's Wide schedule and staged calls. In CIG (Wide), each wave proposes eight plans and evaluates four selected candidates.

Results use the validation-selected Harness. Selection regret measures the gap to the test oracle within each method's generated candidate pool. The final Harness selection rule is distinct from plan selection during discovery.

A.9 Detailed Case Study

This case follows a two-wave CIG (Wide) search with GPT-5.4 as proposer and Qwen3.5-Plus as solver on USPTO-50k, Symptom2Disease, and LawBench. It shows how an apparently beneficial candidate becomes the starting point for a more specific experimental question. The excerpts below condense the recorded nodes without changing their predictions, intervention, or outcomes. All scores are discovery-time validation scores from this single run.

$$E_2 \xrightarrow{\text{grounds}} H_2 \xrightarrow{\text{tested by}} I_2 \xrightarrow{\text{yields}} O_2 \xrightarrow{\text{informs reassessment}} H_2$$

E_2 — Evidence: expose a trade-off within an improvement

Source comparison. The first wave replaces `fewshot_all` with `similarity_retrieval_fewshot`, selecting demonstrations by query-conditioned lexical similarity.

Recorded observation. Aggregate performance rises from 56.90 to 58.00. USPTO improves from 36.67 to 40.00 and Symptom2Disease from 74.00 to 88.00, while LawBench falls from 60.00 to 46.00.

Implication for the next decision. Retrieval is promising but its benefit is not uniform. The linked task-level evidence identifies a specific regression to investigate without discarding the gains of the parent Harness.

H_2 — Hypothesis: turn the trade-off into a testable prediction

Proposed explanation. Query-conditioned retrieval removes irrelevant demonstrations, but selecting only by similarity may remove the broader coverage needed by LawBench. The recorded uncertainty is medium.

Prediction registered before evaluation. A coverage-preserving refinement should recover LawBench, retain the gains on the other tasks, and improve aggregate performance beyond the parent's 58.00.

Failure condition. No LawBench recovery, or offsetting losses that leave aggregate performance at or below 58.00, would defeat the proposed joint improvement target. The hypothesis therefore specifies what the next experiment should establish, rather than merely requesting a higher-scoring candidate.

I_2 — Intervention: test coverage while retaining the retrieval backbone

Parent and candidate. Refine `similarity_retrieval_fewshot` into `similarity_coverage_backfill_fewshot`.

Executable change. Allocate 60% of the prompt budget to nearest lexical neighbors, then fill the remaining slots using answer-space and token-coverage novelty.

Preserved constraints. Keep raw exemplar prompting and query-conditioned lexical retrieval, with one prediction-time solver call, no train-time solver calls, and linear storage. This makes the proposed remedy concrete and connects the evaluated code change to the coverage explanation, rather than replacing the entire Harness with an unrelated design.

O_2 — Outcome: retain the local recovery and the failed joint target

Measured effects relative to the parent.

Metric	Parent	Intervention	Change
USPTO	40.00	33.33	−6.67
Symptom2Disease	88.00	88.00	0.00
LawBench	46.00	52.00	+6.00
Aggregate	58.00	57.78	−0.22

Stored verdict. The record labels the outcome `refuted` and commits it to G_2 as a negative but informative result. The aggregate target was not met.

Scope of the evidence. LawBench's recovery supports the local directional prediction, while USPTO's regression violates the joint no-regression goal. The broader coverage explanation remains unresolved. The graph retains the measured effects alongside the verdict and their links to the tested hypothesis, intervention, and parent.

What the graph makes explicit. The chain preserves an experimental argument, not just a ranking of Harnesses. $E_2 \to H_2$ links the proposed explanation to the observation that motivated it. $H_2 \to I_2$ connects a prospective prediction to an executable change. $I_2 \to O_2$ records what happened when that change was tested. The outcome-to-hypothesis link keeps the negative verdict attached to its evidential scope: the specific joint prediction failed even though one intended effect appeared. These relations make it possible to inspect why the experiment was proposed and which parts of its rationale remain unresolved, rather than reconstructing that argument from a score and a code diff.

First-wave evidence informs the second-wave coverage intervention in the trace above. A separate successful search provides the updated-assessment branch for the comparison below.

A.9.1 State Intervention: Updated versus Retained Assessment

The updated branch comes from a four-wave CIG search with GPT-5.4 as proposer and Qwen3.5-Plus as solver. Each wave proposes eight plans and evaluates two candidates. The comparison starts after the first-wave top-k retrieval candidate is evaluated on USPTO, Symptom2Disease, and LawBench. Its validation scores are 40.00, 84.00, and 48.00, respectively, with an equal-weight mean of 57.33.

Assessment contrast. The earlier explanation concerns dilution from including all demonstrations. The updated explanation identifies a more specific trade-off: local retrieval may remove the broader demonstration coverage needed by LawBench, whose score fell from 60 to 48. These are summaries of the recorded hypotheses. The refined explanation is recorded under a new hypothesis identifier. Both branches start from the same post-evaluation checkpoint with identical new outcomes, raw history, models, and next-wave plan and evaluation counts, differing only in the supplied assessment state.

Assessment → plan → Harness → outcome

The two assessments motivate distinct directions for the next intervention.

	Updated assessment	**Retained assessment**
Selected plan and rationale	Add coverage anchors to local retrieval. The selector cites the LawBench decline from 60 to 48 and targets recovery while preserving the benefits of query-conditioned retrieval.	Refine local retrieval. The aggregate gain supports prioritizing query-relevant demonstrations, motivating more precise selection to improve performance further.
Harness modification	Refine the top-k retrieval Harness with a global label/pattern anchor bank alongside local demonstrations, retaining one prediction-time solver call.	Expand the retrieval pool and rerank examples using complementary lexical signals before retaining the top-k. Preserve the demonstration budget, prompt format, and single prediction-time solver call.
Evaluation result	USPTO: 43.33 (+3.33). Symptom2Disease: 86.00 (+2.00). LawBench: 52.00 (+4.00). Macro mean: $\mathbf{60.44}$ ($+\mathbf{3.11}$).	USPTO: 40.00 (0.00). Symptom2Disease: 84.00 (0.00). LawBench: 48.00 (0.00). Macro mean: $\mathbf{57.33}$ ($\mathbf{0.00}$).

Scores are validation percentages. Parenthesized changes are relative to the shared parent. The macro mean averages the three tasks within this run.

The retained assessment directs finer local relevance ranking, whereas the updated assessment directs complementary demonstration coverage. The resulting Harnesses score 57.33 and 60.44, a 3.11-point difference that links a changed interpretation to a different intervention and higher measured performance.