THE VIEW FROM NOWHERE: POPULATION-CENTERED AGENT EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-improving AI systems can revise the designs that govern their own behavior. Yet existing approaches center improvement on selected parent agents and their evolutionary experience, leaving the broader population largely underused. We identify a substantial gap between population coverage (*i.e.*, the tasks solved by at least one evolved variant) and the performance of the best individual agent. This gap shows that useful capabilities and diagnostic evidence emerge across the population without being consolidated into its leading agents. These observations motivate the **view from nowhere**, a principle for agent evolution: an agent should be improved by interpreting its behavior through population-wide patterns of success, failure, and prior self-modification. We instantiate this principle as **Population-Centered Evolution (PCE)**. PCE distils task records and modification histories from the population into a population block, which captures recurring failures, complementary successes, and the effects of previously attempted modifications to guide improvements to a selected parent's harness. Under matched models, initial harnesses, and evolution schedules, PCE achieves 72.1% success on SWE-bench Verified and 96.7% on Polyglot, improving over a strong group-evolution baseline by 3.2 and 5.9 percentage points. In our evaluation, PCE reduces the gap between population coverage and best-agent performance by one-third on SWE-bench and closes it on Polyglot. These results suggest that treating the population as evidence—not merely as candidates for selection—provides a complementary principle for agent evolution.

1 INTRODUCTION

Self-improving AI aims to develop systems that can revise the designs governing their own behavior (Good, 1966; Schmidhuber, 2007). Evolutionary coding agents pursue this goal by modifying their software harnesses while keeping the parameters of their foundation models fixed (Zhang et al., 2026b; Weng et al., 2026; Robeyns et al., 2025; Wang et al., 2026). These harnesses determine how an agent uses the model, tools, feedback, and its environment to solve a task. Through repeated cycles of self-modification and evaluation, these systems generate a growing population of agent variants. The goal is not merely to produce diverse variants, but to turn useful discoveries made during evolution into progressively stronger agents. How to consolidate these discoveries into improvements that persist in leading agents is a central challenge for agent evolution.

Existing approaches address this challenge through parent selection and experience reuse. The Darwin Gödel Machine (DGM) preserves an archive of agent variants from which future evolution can branch (Zhang et al., 2026b). The Huxley-Gödel Machine (HGM) uses the performance of an agent and its descendants to estimate the potential of continuing that lineage (Wang et al., 2026). Group-Evolving Agents (GEA) selects a small group of parents and allows them to share task records and past self-modifications when proposing improvements (Weng et al., 2026). Despite these differences, the evidence used to propose a modification remains largely *parent-local*: it comes from the selected parent or parent group rather than from a systematic comparison across the full population. Other variants may remain available for future selection, but their task-level successes, failures, and modification histories do not directly inform the current diagnosis.

Yet prior work has not systematically measured the gap between what an evolutionary population collectively solves and what its best individual agent achieves, or examined how this gap can guide

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.

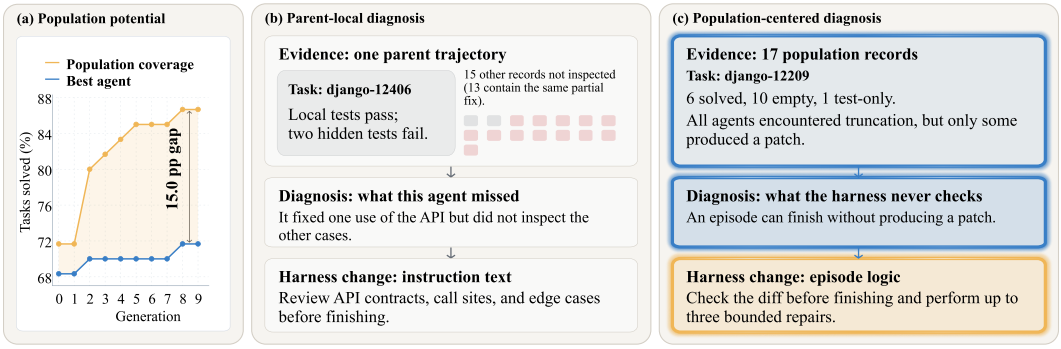


Figure 1: Population potential and task-level evidence. (a) The union of tasks solved by the GEA population exceeds the performance of its best individual agent by 15.0 percentage points (9 tasks in Held-in 60). (b) Parent-local diagnosis uses one parent trajectory to identify missed API cases and add verification instructions, while the other 15 population records are not inspected. (c) Population-centered diagnosis uses patterns across 17 records to identify a missing patch check and add bounded repair logic.

improvement. We define *population coverage* as the proportion of tasks solved by at least one evolved variant. As shown in Figure 1a, in one of our GEA runs the evolved population covers 9 more of the 60 held-in tasks than its best individual agent, corresponding to a 15.0 percentage-point gap. This gap shows that successful outcomes remain distributed across the population rather than being fully consolidated into its leading agent. The associated task records allow us to compare variants that succeed and fail on the same task. The population should therefore be treated not only as a pool for parent selection, but also as comparative evidence for deciding how a selected parent should improve.

These observations motivate the *view from nowhere* (Nagel & Nagell, 1986) as a principle for agent evolution. It holds that decisions about how to improve an agent should be grounded in patterns of success, failure, and prior self-modification across the population. For a given task, population outcomes show whether a failure is isolated or recurring, execution records reveal how successful and unsuccessful variants differ, and modification histories indicate which remedies have already been attempted. Together, these comparisons provide a broader basis for diagnosing a selected parent and guiding its improvement.

We instantiate this principle through **Population-Centered Evolution (PCE)**. For a selected task, a distillation agent compares task records and modification histories across all available variants and organizes its findings into a population block. The block contains three types of information. Outcome evidence measures how frequently different results occur across the population. An observation identifies a cross-agent pattern that may explain why variants behave differently. Candidate files indicate which parts of the harness may be related to that pattern. Together, these fields establish what recurs across the population, suggest a diagnostic hypothesis, and indicate where to investigate it. Diagnosis then checks this hypothesis against the selected parent's execution records and source code before proposing a harness modification, as shown in Figure 2b.

Across SWE-bench Verified and Polyglot, PCE improves agent performance under matched models, initial harnesses, and evolution schedules. Agents evolved by PCE achieve 72.1% success on 200 SWE-bench tasks and 96.7% on 225 Polyglot tasks, exceeding GEA by 3.2 and 5.9 percentage points, respectively. PCE reduces the gap between population coverage and best-agent performance by one-third on SWE-bench and closes it on Polyglot. Its advantage also persists on held-out tasks. Across the full evolutionary process, PCE uses 36% fewer tokens on SWE-bench and 59% fewer tokens on Polyglot than GEA. PCE therefore achieves stronger agents with lower inference expenditure, showing that population-wide evidence enables more effective use of the experience accumulated during evolution.

In summary, we make three contributions:

1. We identify and quantify a previously overlooked gap between population coverage and best-agent performance, and we formulate the *view from nowhere* as a principle for capturing useful information across the population to guide agent evolution.

2. We introduce PCE, which distils population-wide experience into a population block that guides improvements to a selected parent's harness.
3. PCE improves performance and generalization, narrows the population-coverage gap, and achieves these gains with lower total token consumption than baselines.

2 Related Work

Learning from execution experience. Agents use execution experience at several levels. Reflection revises outputs or memories using execution or self-generated feedback (Shinn et al., 2023; Madaan et al., 2023), while memory systems, skill libraries, and workflow memories retain useful experience for future tasks (Park et al., 2023; Packer et al., 2023; Xu et al., 2026; Wang et al., 2023; 2024). ExpeL goes beyond storing individual records by extracting natural-language insights from accumulated task experience (Zhao et al., 2024). Experience can also support parameter updates through self-training on generated responses and filtered trajectories (Zelikman et al., 2022; Chen et al., 2024; Gulcehre et al., 2023; Yuan et al., 2024). Our setting concerns design-level improvement while keeping the foundation model fixed. Self-Harness identifies model-specific failure patterns and validates harness edits with regression tests (Zhang et al., 2026a), while HarnessEvolve aligns failed executions with reference trajectories and clusters error signals to identify systematic failures (Jiang et al., 2026). PCE differs in its unit of analysis. It compares how a population of evolving harness variants handles the same task and relates their outcomes to previous self-modifications. This population-wide comparison provides evidence for diagnosing and modifying the current parent.

Population-based agent design and evolution. Automated agent design methods search over agentic modules, workflows, prompts, and code (Hu et al., 2025; Zhang et al., 2025; Wang et al., 2025; Fernando et al., 2023; Zelikman et al., 2023; Yin et al., 2025; Robeyns et al., 2025). As this search produces more variants, a central problem is how to preserve and reuse discoveries made during evolution. Quality-diversity and open-endedness research retains diverse solutions as stepping stones for continued exploration (Lehman & Stanley, 2011; Mouret & Clune, 2015; Pugh et al., 2016; Chatzilygeroudis et al., 2021; de Vladar, 2016; Stanley et al., 2017; Clune, 2019; Hughes et al., 2024; Faldor et al., 2025; Wang et al., 2019). In coding-agent evolution, DGM maintains an archive from which future evolution can branch (Zhang et al., 2026b), while HGM estimates the potential of continuing a lineage from the performance of an agent and its descendants (Wang et al., 2026). GEA shares task records and self-modification patches among selected parents (Weng et al., 2026), and HarnessBank recombines harnesses drawn from a pool (Luo et al., 2026). These approaches use the population to preserve diversity, select parents, share parent experience, or recombine designs. PCE gives the population a different role. It treats all available variants as comparative evidence for understanding recurring failures, contrasting outcomes, and guiding improvement of a selected parent.

3 Population-Centered Evolution

PCE operates population-centered evolution through two stages, as shown in Figure 2. Population distillation compares task records and modification histories across evaluated agent variants and summarizes the findings in a population block. Diagnosis then checks this population-level account against a selected parent's records and source code before proposing a harness modification.

We note that PCE distinguishes the variants used as evidence from those eligible for parent selection. It maintains a population $\mathcal{V}$ containing all evaluated agent variants whose task records are available, and an archive $\mathcal{A} \subseteq \mathcal{V}$ containing the variants eligible to become parents. Each variant is associated with its harness, task records, and history of self-modification. A variant can therefore inform future improvements even when it is not selected as a parent.

At each evolution step, PCE selects parent agents from $\mathcal{A}$ and chooses a task for diagnosing each parent. Population distillation examines the records available in $\mathcal{V}$ for that task, after which diagnosis proposes and implements a modification to the selected parent's harness. The resulting child is evaluated and added to $\mathcal{V}$. Children satisfying the archive criteria also enter $\mathcal{A}$ for future parent selection.

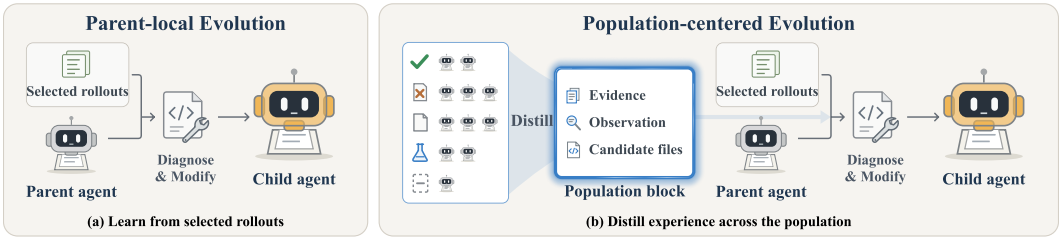


Figure 2: Parent-local and population-centered evolution. (a) Parent-local evolution diagnoses and modifies an agent using execution records associated with the selected parent. (b) Population-centered evolution distils task records and modification histories across the population into a population block. The block summarizes outcome evidence, identifies cross-agent observations, and suggests candidate files for inspection. Diagnosis combines this block with the selected parent's records and source code to propose a harness modification.

3.1 Population Distillation

For the task selected to diagnose a parent, population distillation considers all available task records in $\mathcal{V}$, including the initial agent's record when available. Each record contains an execution log, a submitted task patch, and an evaluation outcome. The corresponding harness can be reconstructed from the initial source and the variant's chain of self-modifications. These materials allow distillation to compare not only which variants succeed or fail, but also how their execution and harness designs differ.

Population distillation combines programmatic measurement with agent-based analysis. A program first counts the outcomes across all available records. The distillation agent then inspects relevant execution logs, task patches, harness source, and earlier self-modifications. It compares successful and unsuccessful variants, identifies recurring outcomes, and examines whether previous harness changes resolved or preserved the observed behavior. Since the full records can be large, the agent chooses which records and source files to inspect within its analysis budget.

The resulting population block contains three fields. *Outcome evidence* reports the program-computed distribution of task outcomes across the population. A *cross-agent observation* summarizes a pattern found by comparing variants and provides a hypothesis about why their outcomes differ. *Candidate files* identify parts of the harness that may implement the relevant behavior and should be examined during diagnosis. This field is left empty when distillation cannot identify a plausible location. Separating measured outcomes from their interpretation preserves a factual population summary while giving diagnosis a focused starting point without prescribing the modification. Appendix A gives the output contract and the distillation budget.

3.2 Diagnosis and Harness Modification

The population block provides a diagnostic hypothesis rather than prescribing a modification. Diagnosis combines this block with the selected parent's source code and task records to determine whether the population-level pattern applies to the current harness. It examines the candidate files when provided, but may inspect other parts of the harness as needed. A mechanism associated with successful variants may be absent from the parent, while an earlier modification may already be present but remain insufficient. Diagnosis uses the parent's execution behavior and implementation to distinguish between these possibilities.

Based on this assessment, diagnosis proposes one harness modification and describes how it should be implemented. The self-modification step applies the proposal to a copy of the parent's harness, producing a child agent. The modification may change the agent's prompts, tools, or execution workflow, while the foundation model and inference limits remain fixed. The child is then passed to the evaluation stage of the evolution loop.

The SWE-bench task django-12209 in Figure 1c illustrates the full process. Across 17 task records, the outcome evidence reported 6 solved patches, 10 empty patches, and 1 test-only patch. Population distillation found that truncation occurred under different outcomes and that increasing the retry

Algorithm 1 PCE evolution loop

Require: Initial agent h_0 with evaluation records; number of generations T
Initialize population and archive $\mathcal{V} = \mathcal{A} = \{h_0\}$
for generation $t = 0, \ldots, T-1$ **do**
  Select parents $P_t = \{h_1, \ldots, h_k\}$ from $\mathcal{A}$; use $P_t = \{h_0\}$ if $t = 0$
  Each parent $h_i \in P_t$ samples a task τ_i from its eligible tasks
  Form inputs L_i: the task record of h_i, followed by the records of the other parents in P_t
  for each $h_i \in P_t$ **in parallel do**
    Collect records R_i on task τ_i from members of $\mathcal{V}$
    Set population block $B_i = \varnothing$
    if $|R_i| \geq 3$ **then**
      $B_i \leftarrow \textsc{Distill}(R_i, \text{patch histories}, h_0)$
      Set $B_i = \varnothing$ if no usable observation is returned
    end if
    Diagnose the harness of h_i using L_i and any available B_i
    Implement the proposal in h_i to produce child $h_{c,i}$
    if a self-modification patch is produced **then**
      Evaluate $h_{c,i}$ on the initial task subset, then remaining tasks if the threshold is met
      Retain its execution logs, task patches, and evaluation outcomes
    end if
  end for
  Add evaluated children to $\mathcal{V}$ and those meeting archive retention criteria to $\mathcal{A}$
end for
return Archive $\mathcal{A}$ and population $\mathcal{V}$ with accumulated records

budget had not consistently prevented empty patches. The observation and candidate files directed diagnosis toward the parent's episode-completion logic. Inspection then showed that the episode could finish without checking whether a patch had been produced. The child harness then added this check and performed bounded repairs when the patch remained empty. This example shows how population-wide patterns are converted into a hypothesis, validated against the selected parent, and implemented as a concrete harness change.

3.3 Evolution Loop

Algorithm 1 specifies the complete PCE evolution loop. The first generation produces two children from the initial agent through separate single-parent diagnoses. Later generations select two parents in out setting from $\mathcal{A}$ and sample one eligible diagnostic task for each. For a selected task, population distillation runs only when at least three records are available in $\mathcal{V}$. Otherwise, or when no usable observation is produced, diagnosis falls back to the parent-local procedure.

Children that produce a self-modification patch are evaluated first on an initial subset and then on the remaining evolution tasks if they meet the evaluation threshold. Every evaluated child joins $\mathcal{V}$, while children satisfying the retention criteria also join $\mathcal{A}$. Appendix A provides the task-sampling rules, record ordering, evaluation thresholds, and archive-update criteria.

4 Experiments Setup

4.1 Benchmarks and Data Splits

We evaluate PCE on SWE-bench Verified and Polyglot (Jimenez et al., 2024; OpenAI, 2024; Aider, 2024). SWE-bench Verified evaluates repository-level issue resolution, while Polyglot contains 225 coding tasks across six programming languages. Following DGM, we use 200 SWE-bench Verified tasks and all 225 Polyglot tasks (Zhang et al., 2026b). For each benchmark, 60 tasks are held in for evolution. The remaining 140 SWE-bench tasks and 165 Polyglot tasks are held out for final evaluation, forming Overall 200 and Overall 225. We also evaluate on an additional set of 60 SWE-bench tasks spanning 12 repositories.

4.2 Baselines and Settings

We compare PCE with GEA, the closest group-evolution baseline (Weng et al., 2026). To isolate the effect of population-centered evolution, both methods use the same foundation models, initial harness, parent-selection procedure, diagnosis prompt, modifiable files, and evaluation protocol. Each method runs for ten generations and produces two children per generation. DeepSeek Flash performs task execution and self-modification. DeepSeek Pro performs diagnosis for both methods and population distillation for PCE. We also evaluate the shared initial agent as a reference.

4.3 Evaluation Protocol

During evolution, each child is first evaluated on a screening set of 10 tasks. Children meeting the advancement threshold are then evaluated on the remaining 50 held-in tasks. Performance across all 60 held-in tasks defines the search score, and the highest-scoring evolved agent is selected for final evaluation.

After evolution, we evaluate the selected agent without further self-modification. For each method, the selected agent and the initial agent are independently evaluated 5 times on the 60 held-in tasks. We report the mean success rate and sample standard deviation. The additional SWE-bench set of 60 tasks is also evaluated 5 times. SWE-bench Held-out 140 and Polyglot Held-out 165 are each evaluated once. Overall scores are task-count-weighted averages of the mean held-in score and the corresponding held-out score. Appendices A and C provide the task splits, model versions, advancement thresholds, and the results of all runs.

5 Results and Analysis

5.1 Agent Performance and Generalization

RQ1: Does PCE evolve stronger agents that generalize beyond the search evaluation?

PCE outperforms GEA on both benchmarks under matched experimental settings. Table 1 reports the performance of the best agents evolved within 10 generations. Both GEA and PCE improve substantially over the shared initial agent. PCE reaches an Overall success rate of 72.1% on SWE-bench and 96.7% on Polyglot, exceeding GEA by 3.2 and 5.9 percentage points, respectively. The Polyglot result is particularly notable because PCE continues to improve performance even when GEA already solves more than 90% of the tasks.

The gains are not limited to a single search or evaluation. Across two independent SWE-bench runs for each method, both PCE runs achieve higher best search scores than either GEA run, as reported in Table 2. When the selected agents are independently evaluated 5 times on Held-in 60, PCE retains a mean advantage of 4.0 percentage points on both benchmarks (Figure 3c). PCE also outperforms GEA on all three held-out task sets (Appendix C, Table 7). Since held-out tasks constitute the majority of Overall 200 and Overall 225, these results show that the gains extend beyond the tasks used during evolution.

For context, Table 1 also reports published results for DGM, GEA, OpenHands, and Aider under their original models and protocols. These values are reference points rather than controlled comparisons. Our main conclusion is based on the matched PCE and GEA runs.

5.2 Population-Wide Improvement and Consolidation

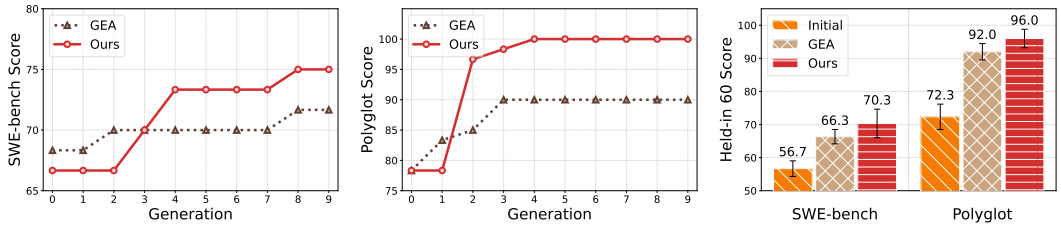
RQ2: Does PCE consolidate population-wide successes into stronger individual agents?

PCE improves not only the best agent, but also the broader set of leading variants. Table 2 reports the highest search score and the minimum score among the top three and top five evolved variants. PCE improves all three measures on both benchmarks. On SWE-bench, even the fifth-ranked PCE variant performs comparably to GEA's best variant. On Polyglot, the fifth-ranked PCE variant outperforms GEA's best. The gains therefore extend beyond a single exceptional agent.

PCE also brings its leading agents closer to what the population collectively solves. Population coverage measures the fraction of tasks solved by at least one evaluated variant, while the coverage

Table 1: Performance (%). Best: highest Held-in 60 search score. †: reported by GEA (Weng et al., 2026). ‡: model transfer. Comparison settings appear in Appendix A.

Benchmark	Method	Model	Gen.	Best↑	Overall↑
SWE-bench	OpenHands†	GPT-5	N/A	N/A	71.8
	DGM	Claude 3.5 Sonnet	80	—	50.0
	DGM‡	Claude 3.7 Sonnet	80	—	59.5
	DGM‡	o3-mini	80	—	33.0
	DGM (GEA)†	Haiku/Sonnet 4.5	60	—	56.7
	GEA†	Haiku/Sonnet 4.5	30	—	71.0
	Initial	DeepSeek-V4-Flash	0	N/A	52.0
	GEA	DeepSeek-V4-Flash	10	71.7	68.9
	PCE	DeepSeek-V4-Flash	10	**75.0**	**72.1**
Polyglot	Aider†	GPT-5 (high)	N/A	N/A	52.0
	DGM	o3-mini	80	—	30.7
	DGM‡	Claude 3.5 Sonnet	80	—	33.3
	DGM‡	Claude 3.7 Sonnet	80	—	36.8
	DGM (GEA)†	Haiku/Sonnet 4.5	40	—	68.3
	GEA†	Haiku/Sonnet 4.5	20	—	88.3
	Initial	DeepSeek-V4.1-Flash	0	N/A	65.1
	GEA	DeepSeek-V4.1-Flash	10	90.0	90.8
	PCE	DeepSeek-V4.1-Flash	10	**100.0**	**96.7**



(a) SWE-bench search (b) Polyglot search (c) Repeated evaluations

Figure 3: Search scores and repeated-evaluation results on Held-in 60. (a, b) Best-so-far search score across the main ten-generation run, with two children produced in each generation. (c) Mean success rate of the shared initial agent and the selected GEA and PCE agents over five independent evaluations after evolution.

gap is the difference between this union and the best-agent score. On SWE-bench, PCE and GEA achieve similar population coverage, but PCE reduces the mean gap from 14.2 to 10.0 percentage points. On Polyglot, both populations cover all 60 held-in tasks. GEA retains a 10.0-point gap, whereas PCE closes the gap with a single agent that solves the full task set.

The reduction does not result from PCE exploring a substantially larger set of solvable tasks. Instead, with similar population coverage, PCE produces leading variants that capture more of the successful outcomes distributed across the population. This result directly supports the central motivation for using population-wide evidence during agent evolution.

5.3 Value of Population-Wide Evidence

RQ3: Does population-wide evidence provide gains beyond additional parent-only analysis?

We conduct an ablation on Polyglot that compares GEA, parent-only distillation, and PCE under the same evolution and evaluation settings. Parent-only distillation uses the same distillation model, tools, step budget, and output format as PCE, but restricts its inputs to the selected parents' task

Table 2: Leading-variant performance and population coverage on Held-in 60 (%). Top-1 is the highest search score, and Top-k min. is the lowest score among the top-k evolved variants. Union is the fraction of tasks solved by at least one evaluated child, and Gap is Union minus Top-1 in percentage points. SWE-bench reports mean ± sample standard deviation over two runs; Polyglot reports one run.

Benchmark	Method	Top-1↑	Top-3 min.↑	Top-5 min.↑	Union	Gap↓
SWE-bench	GEA	70.0 ± 2.4	66.7 ± 2.4	66.7 ± 2.4	84.2 ± 3.5	14.2 ± 1.2
	PCE	**75.0** ± 0.0	**72.5** ± 1.2	**70.8** ± 1.2	85.0 ± 0.0	**10.0** ± 0.0
Polyglot	GEA	90.0	85.0	83.3	100.0	10.0
	PCE	**100.0**	**98.3**	**96.7**	100.0	**0.0**

Table 3: Parent-only distillation ablation on Polyglot Held-in 60 (%). Best is the highest search score during evolution. The selected agent is then evaluated five times. Mean ± sd summarizes these evaluations, Intersection is the fraction of tasks solved in every evaluation, and Union is the fraction solved at least once. Parent-only distillation matches PCE's distillation model, tools, budget, and output format but uses only evidence from the selected parents.

Method	Best	Mean ± sd	Intersection	Union
GEA	90.0	92.0 ± 2.5	85.0	100.0
Parent distillation	91.7	92.0 ± 4.3	78.3	98.3
Population distillation (PCE)	**100.0**	**96.0** ± 2.8	**88.3**	100.0

records, modification histories, and source code. This control separates the value of population-wide evidence from the effect of adding another analysis step.

Population-wide evidence improves performance beyond parent-only distillation. As shown in Table 3, parent-only distillation achieves a best search score of 91.7%, close to GEA's 90.0%. Both methods obtain the same mean success rate of 92.0% across 5 evaluations of their selected agents. PCE instead reaches a perfect search score and retains a mean success rate of 96.0%, 4 percentage points above both alternatives.

The repeated evaluations further show that PCE solves more tasks consistently. All three methods solve nearly the full task set at least once, as reflected by their similar Union scores. PCE nevertheless achieves the highest Intersection, solving 88.3% of the tasks in every evaluation, compared with 85.0% for GEA and 78.3% for parent-only distillation. Within this Polyglot ablation, the results show that the gains come from broadening the evidence beyond the selected parents, rather than from additional analysis alone.

Effect on diagnostic proposals. We also conduct a fixed-context probe to examine how the supplied evidence changes diagnosis. Across 3 SWE-bench contexts, we hold the focal parent, source code, patch history, and task record fixed, vary the auxiliary evidence, and generate five proposals per condition. Adding a population block changes the main target of diagnosis in two of three contexts. On Sphinx-7748 task, proposals addressing truncation recovery increase from two of five to all five. On Sphinx-9229 task, the population block produces two proposals for a post-episode patch check, while GEA and the raw-trajectory controls produce only instruction-level changes. No comparable shift appears on Sphinx-8638. These results suggest that a population block can redirect diagnosis from local instruction refinement toward changes in the agent's execution logic. They also indicate that distilling cross-agent patterns can provide guidance that is not reproduced by simply supplying additional raw trajectories. Since the proposals are not implemented or evaluated, this analysis characterizes how population evidence affects diagnostic behavior rather than its downstream performance. Appendix B provides the full conditions, coding procedure, and results.

5.4 Performance Gains and Token Cost

RQ4: Does PCE achieve better performance at the cost of higher token consumption?

Table 4: Performance and resource use in the main evolution runs. Token and request counts are provider totals for the complete run, covering task execution, diagnosis, self-modification, and population distillation. Distillation calls report the number made by the distillation agent. An evaluation episode is one task attempt, and trajectory length is measured in characters. We report tokens and requests rather than monetary cost because provider prices vary over time.

Metric	SWE-bench		Polyglot	
	GEA	PCE	GEA	PCE
Overall success (%)↑	68.9	**72.1**	90.8	**96.7**
Total tokens (billions)↓	2.98	**1.92**	0.99	**0.41**
Total requests↓	114,629	**69,539**	52,034	**28,823**
Distillation calls	0	758	0	775
Evaluation episodes	1,100	1,200	1,147	1,177
Mean trajectory length, solved (k chars)↓	215	**146**	101	**65**
Mean trajectory length, unsolved (k chars)↓	354	**242**	126	**60**
Episodes above 500k chars↓	158	**45**	4	**1**

No. PCE achieves higher performance while using substantially fewer tokens. Table 4 shows that PCE reduces total token consumption by 36% on SWE-bench and 59% on Polyglot relative to GEA. These totals cover the complete evolution run, including task execution, diagnosis, self-modification, and population distillation. PCE also makes fewer model requests on both benchmarks. Its performance gains therefore do not come from a larger inference budget.

The reduction is not a result of evaluating fewer task episodes. PCE evaluates more episodes than GEA on both benchmarks, but produces substantially shorter trajectories for both solved and unsolved attempts. Extremely long trajectories also become less frequent. On SWE-bench, the number of trajectories exceeding 500,000 characters falls from 158 to 45. PCE therefore performs more evaluations while consuming fewer tokens overall.

These results suggest that population-wide evidence makes the evolutionary process more directed. Population distillation identifies recurring outcomes and informs harness changes that can affect later task attempts. Figure 1c provides one example, where a recurring empty-patch pattern leads to an explicit patch check and bounded repair procedure. This case does not by itself explain the aggregate savings, but it illustrates how population-guided modifications can prevent inefficient behavior from recurring. PCE therefore improves both performance and token efficiency, rather than exchanging a larger inference budget for stronger agents.

6 Conclusion

Agent evolution can produce useful experience that no selected parent captures alone. The gap between population coverage and best-agent performance makes this overlooked potential visible. We formulate the *view from nowhere* as a principle for using the population not only to select parents, but also to understand how an individual agent should improve. PCE realizes this principle by distilling outcomes and modification histories across evolved variants into evidence for parent-specific diagnosis. Across SWE-bench Verified and Polyglot, PCE improves performance, retains its gains on held-out tasks, raises the quality of several leading variants, and brings the best agent closer to what the population collectively solves. The Polyglot ablation further shows that these gains do not arise from additional parent-only analysis. Notably, PCE also moves beyond the usual performance-cost tradeoff. It produces stronger agents while using fewer total tokens, rather than purchasing improvement with a larger inference budget. The broader lesson is that an evolutionary population can serve as a source of knowledge, not merely a pool of candidates. We envision that this perspective motivates further study of how population evidence can be attributed, compressed, and transferred as evolutionary histories grow and self-improving systems move beyond coding agents.

AI USE STATEMENT

We used generative AI tools in three ways. They polished the text and drafted sections of the paper. They searched for related work that we might have missed. They took part in discussions of the experimental design and assisted with implementation code. We did not use generative AI tools to generate synthetic datasets or to prove mathematical claims. All AI-assisted work was reviewed, and all references, code, and results were verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the code as a zip archive in the supplementary material and will open-source all code upon acceptance. Section 3 and Algorithm 1 give the algorithmic details and pseudocode. Section 4 and Appendix A describe the benchmarks, task subsets, foundation models, evolution loop, and distillation budget. Appendix C reports the per-child scores, repeated evaluations, and held-out results behind every table, and Table 4 gives the token and request totals of the main runs. Identifiers of the reported agents and the code versions of each run are provided with the code archive.

REFERENCES

Aider. o1 tops aider's new polyglot leaderboard. `https://aider.chat/2024/12/21/polyglot.html`, 2024. Accessed: 2026-09-05.

Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. Quality-diversity optimization: a novel branch of stochastic optimization. In *Black box optimization, machine learning, and no-free lunch theorems*, pp. 109–135. Springer, 2021.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Jeff Clune. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*, 2019.

Harold P de Vladar. Why greatness cannot be planned: The myth of the objective by kenneth o. stanley and joel lehman. *Leonardo*, 49(1):99–100, 2016.

Maxence Faldor, Jenny Zhang, Antoine Cully, and Jeff Clune. Omni-epic: Open-endedness via models of human notions of interestingness with environments programmed in code. In *International Conference on Learning Representations*, volume 2025, pp. 85260–85385, 2025.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

Irving John Good. Speculations concerning the first ultraintelligent machine. In *Advances in computers*, volume 6, pp. 31–88. Elsevier, 1966.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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

Edward Hughes, Michael Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge Shi, Tom Schaul, and Tim Rocktaschel. Open-endedness is essential for artificial superhuman intelligence. *arXiv preprint arXiv:2406.04268*, 2024.

Wen Jiang, Mingmin Chu, Yimeng Tian, Qianxin Zhang, Haofei Yang, Rui Yang, Yang Liu, Tao Lv, and Fangming Li. Harnessevolve: Learning from reference trajectories for reliable agent self-evolution. *arXiv preprint arXiv:2609.00829*, 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.

Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

Xiaotian Luo, Dizhan Xue, Fengxingyu Wang, Chuanrui Hu, and Yafeng Deng. Harnessbank: Semantic gene-bank search with gated verification for agent-harness self-evolution, 2026. URL https://arxiv.org/abs/2607.13683.

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

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Thomas Nagel and Thomas Nagell. *The view from nowhere*, volume 13. oxford university press Oxford, 1986.

OpenAI. Introducing SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/, 2024. Accessed: 2026-09-02.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent. *arXiv preprint arXiv:2504.15228*, 2025.

Jürgen Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. In *Artificial general intelligence*, pp. 199–226. Springer, 2007.

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

Kenneth O. Stanley, Joel Lehman, and Lisa Soros. Open-endedness: The last grand challenge you've never heard of. O'Reilly Radar, https://www.oreilly.com/radar/open-endedness-the-last-grand-challenge-youve-never-heard-of/, 2017.

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

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

Wenyi Wang, Piotr Piekos, Li Nanbo, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-g\" odel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. In *International Conference on Learning Representations*, volume 2026, pp. 80356–80386, 2026.

Yingxu Wang, Siwei Liu, Jinyuan Fang, and Zaiqiao Meng. Evoagentx: An automated framework for evolving agentic workflows. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 643–655, 2025.

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

Zhaotian Weng, Antonis Antoniades, Deepak Nathani, Zhen Zhang, Xiao Pu, and Xin Eric Wang. Group-evolving agents: Open-ended self-improvement via experience sharing. *arXiv preprint arXiv:2602.04837*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel agent: A self-referential agent framework for recursively self-improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27890–27913, 2025.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (stop): Recursively self-improving code generation. *arXiv preprint arXiv:2310.02304*, 2023.

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. In *International Conference on Learning Representations*, volume 2026, pp. 104223–104294, 2026b.

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.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

A EXPERIMENTAL SETUP

Models and provider. Task execution and self-modification use DeepSeek-V4-Flash on SWE-bench and DeepSeek-V4.1-Flash on Polyglot. Diagnosis and population distillation use DeepSeek-V4-Pro in all runs. Models are accessed through OpenRouter with the provider pinned to CoreWeave.

Table 5: Search scores of the children in all reported runs. Each cell lists the two children of a generation, scored out of 60 tasks. An entry of 0/10 denotes an evaluation stopped at the ten-task gate, and a dash denotes self-improvement that produced no patch. Min, Max, and Median are over fully evaluated children.

	SWE-bench				Polyglot		
Gen	PCE 1	PCE 2	GEA 1	GEA 2	PCE	GEA	Parent dist.
0	40, 34	40, 29	41, 31	25, 15	38, 47	42, 47	45, 48
1	38, 40	40, 36	0/10, 33	27, 19	38, 45	50, 41	45, 46
2	40, 40	42, 45	42, 39	28, 25	42, 58	51, 50	47, 47
3	42, 37	42, 40	40, 37	39, 13	58, 59	47, 54	45, 51
4	44, 38	41, 40	39, 19	37, 27	60, 57	49, 44	0/10, 44
5	43, 42	39, 40	38, 39	40, 21	45, 41	47, 50	45, 50
6	37, 43	39, 43	41, 38	39, 30	57, 36	53, 48	44, 52
7	39, 42	34, 43	39, 39	39, 39	58, 48	44, 51	–, 54
8	45, 45	39, 39	0/10, 43	35, 37	49, 34	47, 47	48, 54
9	42, 42	38, 45	39, 41	37, 41	59, 49	49, 41	53, 55
Fully evaluated	20	20	18	20	20	20	18
Min / Max	34 / 45	29 / 45	19 / 43	13 / 41	34 / 60	41 / 54	44 / 55
Median	41.0	40.0	39.0	32.5	48.5	47.5	47.5

Task sets. Both benchmarks use DGM's 60-task subsets as Held-in 60 (Zhang et al., 2026b). Each subset is partitioned into ten gate tasks and fifty further tasks, and a child passes the gate with at least one solved gate task on SWE-bench and at least four on Polyglot. The SWE-bench Held-in 60 comprises 35 Django and 25 Sphinx tasks. SWE-bench Held-out 140 comprises DGM's additional 118 Django and 22 Astropy tasks, and the separate SWE-bench Held-out 60 spans 12 repositories. Polyglot Held-out 165 is the remainder of the 225 tasks. The best agent of each run is selected by its Held-in 60 search score and evaluated without further self-modification. It is evaluated five times on Held-in 60 and on SWE-bench Held-out 60, and once on Held-out 140 and Held-out 165. Overall weights the mean held-in success rate and the single held-out success rate by their task counts.

Additional SWE-bench Held-out 60. We construct this set by first validating SWE-bench Verified instances in our evaluation environment. We sample 60 tasks using fixed seed, requiring representation from every repository in the candidate pool and matching the held-in difficulty distribution as closely as possible. The resulting set spans 12 repositories. We evaluate the initial agent and the best agents evolved by GEA and PCE five times on this set.

Evolution loop. Each run spans ten generations with two children per generation. Generation 0 uses single-parent diagnosis from the initial harness, with a separately sampled task record for each child. Later generations select two parents ($k = 2$) using GEA's performance-novelty rule and uniformly sample one eligible task per parent. Each diagnosis receives both records, with the inheriting parent's record first. PCE additionally supplies a population block for that parent's task when available. Modifications are restricted to `coding_agent.py` (episode logic), `harness_def.py` (instructions and truncation handling), `tools/`, and `utils/`, excluding evaluation and orchestration helpers. The model, output cap, and tool loop remain fixed. Children producing harness patches are evaluated on ten gate tasks and, if they pass, the remaining fifty. Archive admission requires passing the gate, producing at least one non-empty task patch, and restricting harness changes to the permitted paths. The archive is never pruned. All evaluated children enter $\mathcal{V}_{t+1}$, regardless of archive admission, making their records available for subsequent distillation.

Table 6: Proposal counts from five diagnoses per condition. I: instructions. E: episode logic. T: new tool. Dashes indicate untested trajectory controls due to insufficient eligible trajectories at generation 1.

Prompt condition	Sphinx-7748 I / E / T	Sphinx-8638 I / E / T	Sphinx-9229 I / E / T
GEA	3 / 2 / 0	4 / 0 / 1	5 / 0 / 0
+Success	—	4 / 0 / 1	5 / 0 / 0
+Own failures	—	4 / 0 / 1	5 / 0 / 0
+Population failures	—	3 / 0 / 2	5 / 0 / 0
+Block	0 / **5** / 0	5 / 0 / 0	3 / **2** / 0

Population distillation. The distillation agent runs with a budget of 250 steps and returns an observation of at most 60 words and zero to three candidate files. Program-computed outcome counts are rendered with the observation. Distillation is skipped when fewer than three records exist on the task, so the first opportunity is generation 1, and the diagnosis proceeds without a block when no usable observation is returned.

Published comparison settings. Table 1 lists task-execution models and generation counts. GEA and PCE produce two children per generation, compared with one for DGM. Haiku/Sonnet denotes GEA's Haiku 4.5 to Sonnet 4.5 schedule. DGM's model-transfer entries reuse the evolved harness without additional search.

Table 7: Repeated and held-out evaluations of the selected agents. Rep 1–5 are solved-task counts, and the last column is mean success percentage $\pm$ sample standard deviation. Held-out 140 and Held-out 165 are evaluated once, and their rows list the solved counts and success percentages of the initial agent, GEA, and PCE. All tasks of a set remain in the denominator, including never-graded attempts.

Benchmark	Set	Agent	Rep 1–5	Mean $\pm$ sd (%)
SWE-bench	Held-in 60	Initial	33, 33, 36, 33, 35	56.7 $\pm$ 2.4
		GEA	39, 39, 39, 42, 40	66.3 $\pm$ 2.2
		PCE	39, 43, 45, 44, 40	70.3 $\pm$ 4.3
	Held-out 60	Initial	38, 30, 31, 29, 32	53.3 $\pm$ 5.9
		GEA	41, 39, 39, 40, 37	65.3 $\pm$ 2.5
		PCE	40, 42, 43, 42, 44	70.3 $\pm$ 2.5
	Held-out 140	Initial / GEA / PCE	70 / 98 / 102	50.0 / 70.0 / 72.9
Polyglot	Held-in 60	Initial	46, 43, 45, 40, 43	72.3 $\pm$ 3.8
		GEA	55, 57, 56, 53, 55	92.0 $\pm$ 2.5
		Parent distillation	51, 55, 56, 56, 58	92.0 $\pm$ 4.3
		PCE	56, 56, 60, 58, 58	96.0 $\pm$ 2.8
	Held-out 165	Initial / GEA / PCE	103 / 149 / 160	62.4 / 90.3 / 97.0

B Diagnostic Proposals under Fixed Contexts

We use three diagnostic contexts from a completed SWE-bench GEA run, keeping the focal parent, source code, modification history, and task record fixed. GEA retains the paired parent's original entry; +Block adds a fixed population block. The remaining conditions replace that entry with the paired parent's successful trajectory on another task (+Success), three of its failures on other tasks (+Own failures), or three other agents' failures on the focal task (+Population failures). Sphinx-7748 occurs at generation 1 and lacks sufficient eligible trajectories for these replacements. Table 6 reports five proposals per condition, each classified by its main target as instructions (I), episode logic (E), or a new tool (T).

C Additional Results

For SWE-bench, Table 5 reports per-child search scores from two independent runs of each method, including gate outcomes and summary statistics for fully evaluated children. Run 1 supplies the main performance and cost results, while Table 2 summarizes both runs. Table 7 provides five repeated evaluations on Held-in 60 and Held-out 60, together with the single evaluation on Held-out 140.

For Polyglot, Table 5 reports per-child search scores for GEA, parent distillation, and PCE. Table 7 lists five Held-in 60 evaluations of the initial and best evolved agents, including the parent-distillation control. It also reports Held-out 165 results for the initial agent, GEA, and PCE. These results support the main performance comparison and the population-distillation ablation.