Group-Evolving Agents: Open-Ended Self-Improvement via Experience Sharing

Anonymous Authors[1]

Abstract

Open-ended self-improving agents can autonomously modify their own structural designs to advance their capabilities and overcome the limits of pre-defined architectures, thus reducing reliance on human intervention. We introduce **Group-Evolving Agents (GEA)**, a new paradigm for open-ended self-improvements, which treats a group of agents as the fundamental evolutionary unit, enabling explicit experience sharing and reuse within the group throughout evolution. Unlike existing open-ended self-evolving paradigms that adopt tree-structured evolution, GEA overcomes the limitation of inefficient utilization of exploratory diversity caused by isolated evolutionary branches. We evaluate GEA on challenging coding benchmarks, where it significantly outperforms state-of-the-art self-evolving methods (71.0% vs. 56.7% on SWE-bench Verified, 88.3% vs. 68.3% on Polyglot) and matches or exceeds top human-designed agent frameworks (71.8% and 52.0% on two benchmarks, respectively). Analysis reveals that GEA more effectively converts early-stage exploratory diversity into sustained, long-term progress, achieving stronger performance under the same number of evolved agents. Furthermore, GEA exhibits consistent transferability across different coding models and greater robustness, fixing framework-level bugs in 1.4 iterations on average, versus 5 for self-evolving methods.

1. Introduction

Open-endedness and cumulative progress are key characteristics of scientific breakthroughs (Stanley et al., 2017; Zhang et al., 2025a; Stanley & Lehman, 2015). However, most existing AI systems rely on pre-defined model architectures designed by humans. Although such systems can accumulate experience through training, they often struggle to transcend the capability boundaries imposed by their initial designs, as they lack the ability to modify their own structural configurations (Yin et al., 2025). Thus, progress remains heavily dependent on continuous human intervention.

Existing open-ended self-improving systems are largely inspired by biological evolution and designed around individual-centric evolutionary processes (Zhang et al., 2025a; Yin et al., 2025; Schmidhuber, 2003; Robeyns et al., 2025; Novikov et al., 2025). At each iteration, a single agent is selected as the parent and refined to produce one or more offspring(Figure 1a). The overall structure follows chain- or tree-structured evolution, where different branches remain strictly isolated. Consequently, although such systems often exhibit substantial exploratory diversity, this diversity rarely serves as effective stepping stones (Mouret & Clune, 2015; Pugh et al., 2016). Instead, many agents provide only temporary diversity, producing short-lived variants that fail to contribute to long-term cumulative progress.

It is time to rethink agent evolution. *AI agents are not biological individuals; why should their evolution remain constrained by biological paradigms?* In fact, AI agents can directly share trajectories, tools, and learned artifacts, and they can aggregate complementary skills without the constraints of reproduction or lineage.

Therefore, we introduce **Group-Evolving Agents (GEA)**, a new paradigm for open-ended self-improvement that treats a *group of agents*, rather than an individual agent, as the fundamental unit of evolution (Figure 1b). This shift enables explicit experience sharing and reuse across agents within a group, naturally allowing exploratory discoveries from different agents to be consolidated and accumulated into long-term progress rather than remaining as short-lived variants. At each iteration, GEA first selects a parent group of agents using a Performance-Novelty criterion that balances immediate performance gains with evolutionary diversity. The parent agents then jointly produce a child group through a shared pool of aggregated experience from all members.

We evaluate GEA on challenging coding benchmarks, achieving success rates of 71.0% on SWE-bench Verified and 88.3% on Polyglot, significantly outperforming state-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

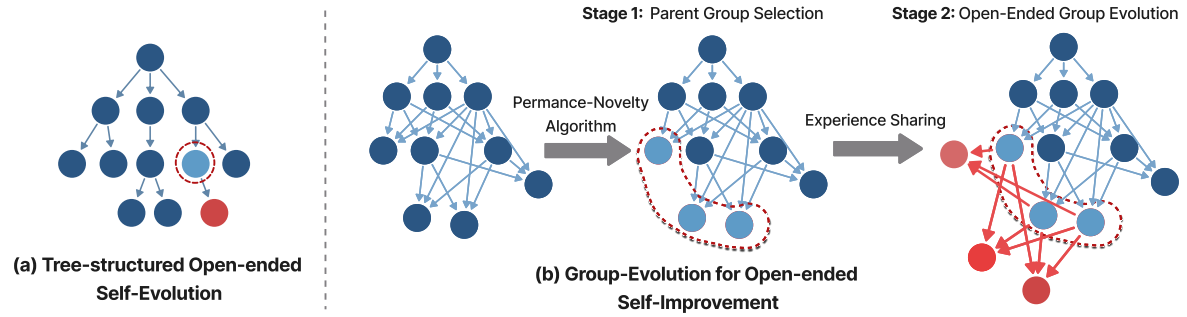


Figure 1. **Overview of Group-Evolving Agents (GEA) vs. tree-structured self-evolution for open-endedness.** GEA treats a *group of agents*, rather than an individual agent, as the fundamental unit of evolution. At each iteration, a parent group jointly gives rise to an offspring group through explicit intra-group Experience sharing and reuse.

of-the-art open-ended self-evolving methods (56.7% and 68.3%, respectively). Analysis reveals that GEA more effectively consolidates the diversity generated during open-ended exploration, yielding sustained progress and stronger performance given the same number of evolved agents. By leveraging experience from better-performing agents, GEA also exhibits stronger robustness to framework-level perturbations. Furthermore, its improvements stem from workflow and tool enhancements rather than model-specific optimizations, thus transferring consistently across GPT- and Claude-series models.

Additionally, by leveraging meta-learning for self-improvement in open-ended exploration, without any human intervention, GEA achieves performance comparable to or even surpassing human-designed state-of-the-art frameworks on both benchmarks (71.0% vs. 71.8% on SWE-bench Verified, 88.3% vs. 52.0% on Polyglot).

In summary, we propose **Group-Evolving Agents**, a new paradigm for open-ended self-improvement that:

1. Overcomes the limitation of inefficient utilization of exploratory diversity caused by branch isolation in existing tree-structured evolution, by enabling explicit experience sharing and reuse within the group during evolution.

2. More effectively consolidates and reuses experience and evolutionary diversity from other agents, achieving significant performance gains and stronger robustness over state-of-the-art open-ended self-evolving methods, with improvements that transfer consistently across different coding models.

3. Matches or surpasses human-designed state-of-the-art frameworks through meta-learning-based self-improvement without human intervention.

2. Related Work

Recent years have witnessed growing interest in how AI systems can continuously improve themselves without human intervention (Fang et al., 2025; Gao et al., 2025; Wang et al., 2025b). Most existing self-improving approaches mainly focus on continuous, iterative refinement of the given agent system (Shinn et al., 2023; Song et al.; Gao et al., 2025; Zelikman et al., 2024; Singh et al.), typically evolving toward a specific optimization objective and following a linear, chain-based evolutionary structure (Zhang et al., 2025b; Wang et al., 2025b; Robeyns et al., 2025) . Such systems achieve self-improvement through mechanisms such as self-play against historical versions or self-generated verification (Setlur et al., 2024; Wang et al., 2024; Huang et al., 2025; Wang et al., 2025a; Wei et al., 2025), supervised fine-tuning (Zelikman et al., 2022; Gou et al., 2023; Ni et al., 2024) or reinforcement learning on selectively filtered feedback (Xin et al., 2024; Zhao et al., 2025; Yuan et al., 2024) , and reflection-based methods (Wang et al., 2025c; Yin et al., 2025; Shinn et al., 2023) or in-context learning (Hua et al., 2024; Sun et al., 2023). While this goal-oriented, chain-based evolutionary paradigm enables autonomous improvement along a particular direction, it inherently limits the ability of self-evolving systems to explore diverse evolutionary directions in open-ended solution spaces.

A line of work has pointed out that one of the key challenges in enabling unbounded improvement and innovation lies in developing open-ended AI systems that can continuously produce both novel and learnable artifacts (Stanley et al., 2017; Zhang et al., 2025a; Stanley & Lehman, 2015; Shapere, 1964). Building on this insight, open-endedness has been characterized as the capability of systems to continuously generate artifacts that are novel, interesting, and learnable from a human perspective (Zhang et al., 2025a; Clune, 2019; Hughes et al., 2024; Jiang et al., 2023; Zhang et al.; Faldor et al.).

Motivated by the potential of enabling unbounded evolution through open-ended exploration in self-evolving agents,

more recent studies adopt lineage-based, tree-structured evolutionary strategies inspired by biological inheritance and mutation (Zhang et al., 2025a; Novikov et al., 2025; Fernando et al., 2024; Huang et al., 2023). In these frameworks, individual parent agents are selected at each iteration to independently produce offspring, enabling various branching exploration across multiple evolutionary directions and helping avoid local optima. However, the strict isolation between evolutionary branches prevents effective information and experience sharing and reuse across lineages. As a result, many promising directions discovered early in evolution persist only as temporary diversity and fail to contribute to long-term cumulative progress. To overcome this limitation, we introduce a group-centric evolutionary paradigm, **Group-Evolving Agents (GEA)**, which explicitly enables intra-group experience sharing and reuse throughout the evolutionary process. By consolidating complementary discoveries across agents, GEA more effectively leverages the diversity generated by open-ended exploration to support sustained cumulative progress.

3. Method

We propose **G**roup-**E**volving **A**gents, a framework for open-ended evolution that treats a *group of agents* as the fundamental unit of evolution. GEA maintains an archive that stores all discovered agents throughout the evolutionary process. As shown in Figure 1, at each iteration, GEA proceeds in two core stages:

(1) Parent Group Selection (§3.1): GEA first selects K parent agents from the archive using a Performance–Novelty selection strategy (Mouret & Clune, 2015; Pugh et al., 2016; Chatzilygeroudis et al., 2021) that balances immediate task-solving competence with long-term evolutionary diversity and potential.

(2) Open-ended Group Evolution (§3.2): The selected agents form a parent group that jointly produces an offspring group of the same size through explicit experience sharing and reuse across parent agents.

We detail the method below.

3.1. Parent Group Selection

Inspired by Mouret & Clune (2015); Pugh et al. (2016); Chatzilygeroudis et al. (2021), parent group selection in GEA balances two key principles: *performance* and *novelty*. We prioritize agents with strong task performance, as performance reflects an agent's immediate competence and its likelihood of producing effective offspring, since evolution in GEA proceeds through iterative modifications of the agent's implementation, which itself constitutes a form of solving coding problems. At the same time, we also encourage exploration beyond currently well-optimized regions of the search space, as agents that exhibit novel evolutionary directions may contribute to long-term cumulative progress even when their current performance is not optimal.

Algorithm 1 Parent Group Selection with KNN Novelty

Input: Archive of agents $\mathcal{A}$; agent-representation vectors $\{z_i \in \{0,1\}^D\}_{i\in\mathcal{A}}$; performance scores $\{\alpha_i\}_{i\in\mathcal{A}}$; parent group size K; KNN size M.
Output: Parent agent group $\mathcal{G}$ with $|\mathcal{G}| = K$.
/* Compute novelty for each agent */
for $i \in \mathcal{A}$ **do**
  Initialize empty list $\mathcal{D}_i$
  for $j \in \mathcal{A},\ j \neq i$ **do**
    $d_{ij} \leftarrow 1 - \dfrac{z_i^\top z_j}{\|z_i\|_2\,\|z_j\|_2 + \varepsilon}$
    Append d_{ij} to $\mathcal{D}_i$
  end for
  Let $\mathcal{N}_M(i)$ be the indices of the M smallest values in $\mathcal{D}_i$
  $\mathrm{nov}(i) \leftarrow \dfrac{1}{M}\sum_{j\in\mathcal{N}_M(i)} d_{ij}$
end for
/* Rank agents by Performance–Novelty score */
for $i \in \mathcal{A}$ **do**
  $\mathrm{score}(i) \leftarrow \alpha_i \cdot \sqrt{\mathrm{nov}(i)}$
end for
$\mathcal{G} \leftarrow$ the top-K agents in $\mathcal{A}$ ranked by $\mathrm{score}(\cdot)$
return $\mathcal{G}$

We represent each agent i using a task-success vector $z_i \in \{0,1\}^D$, where each dimension indicates whether the agent successfully solves a corresponding probe task. Similar binary task–response representations of this form have been widely used to characterize an agent's coding capabilities and to better understand how these capabilities are distributed across various tasks (Polo et al., 2024; Rodriguez et al., 2021). Using this representation, we measure the dissimilarity between two agents via cosine distance:

$$d(i,j) = 1 - \frac{z_i^\top z_j}{\|z_i\|_2\,\|z_j\|_2 + \varepsilon}. \tag{1}$$

We define the novelty of agent i as the average cosine distance to its M most similar neighbors:

$$\mathrm{nov}(i) = \frac{1}{M}\sum_{j\in\mathcal{N}_M(i)} d(i,j), \tag{2}$$

where $\mathcal{N}_M(i)$ denotes the set of M agents with the smallest cosine distance to agent i.

To construct the parent group, we rank agents according to a combined score

$$\mathrm{score}(i) = \alpha_i \cdot \sqrt{\mathrm{nov}(i)}, \tag{3}$$

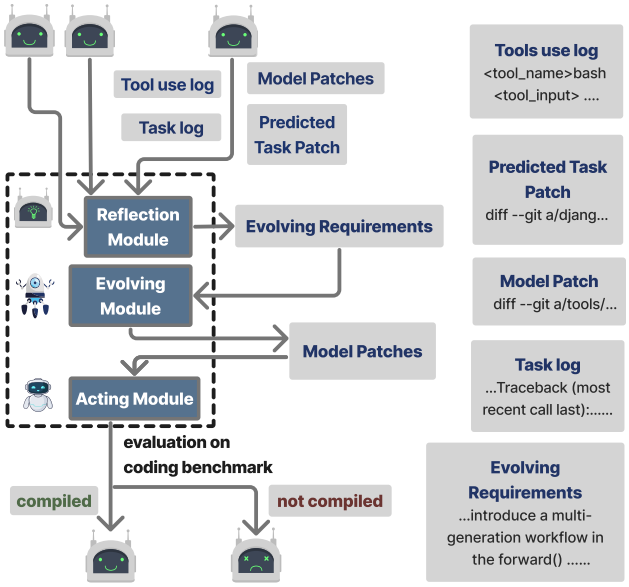


Figure 2. **Detailed illustration of group-level evolution in GEA.** Aggregated evolutionary traces from the parent group are shared across all agents to generate evolution directives and framework-level patches.

where α_i denotes the performance of agent i on downstream coding tasks, and $\sqrt{\mathrm{nov}(i)}$ moderates the influence of novelty. Finally, we select the top-K agents according to this score to form the parent group. Performance serves as the primary selection criterion, while novelty is incorporated as a mild bias without dominating performance, enabling a balanced trade-off between exploitation and exploration. The full procedure is summarized in Algorithm 1.

3.2. Open-Ended Group Evolution

Unlike conventional approaches where parent agents evolve independently without information and experience exchange, GEA explicitly enables experience sharing and reuse among agents during evolution. This group-level experience sharing allows agents to integrate complementary evolutionary directions explored by different agents while maintaining open-ended exploration. Diversity generated during exploration is thus transformed from transient variations into long-term useful experience, effectively contributing to sustained evolutionary progress.

Given a selected parent group $\mathcal{G} = \{a_1, a_2, \ldots, a_K\}$, GEA generates a new group $\mathcal{G}'$ of the same size, where each agent evolves by leveraging both its own evolutionary history and experience aggregated from other members of the parent group, as demonstrated in Figure 2.

For each agent $a_i \in \mathcal{G}$, we collect a set of evolutionary traces consisting of:

1. the code modification patches applied to the agent's framework;
2. a predicted task patch generated by a_i for a randomly sampled unsolved task during evaluation;
3. the corresponding task execution logs, including the complete tool invocation history and execution workflow;
4. the evaluation outcome of the same task, which exposes failure modes and potential directions for framework-level improvement.

The aggregated traces from all agents in the parent group are provided as shared input to every agent. Each agent evolves from this shared pool of group-level experience

Algorithm 2 Open-Ended Group-Evolving

Input: Parent group $\mathcal{G} = \{a_1, \ldots, a_K\}$; archive $\mathcal{A}$; coding benchmark $\mathcal{T}$
Output: Offspring group $\mathcal{G}'$ with $|\mathcal{G}'| = K$; updated archive $\mathcal{A}$
Initialize offspring group $\mathcal{G}' \leftarrow \emptyset$
for $a_i \in \mathcal{G}$ **do**
  /* Collect evolutionary traces */
  $\mathcal{P}_i^{\text{Applied}} \leftarrow \textsc{GetAppliedPatches}(a_i)$
  $t_i^{\text{fail}} \leftarrow \textsc{SampleUnsolvedTask}(a_i)$
  $\mathcal{P}_i^{\text{pred}} \leftarrow \textsc{GetPredictedTaskPatch}(a_i, t_i^{\text{fail}})$
  $\mathcal{L}_i \leftarrow \textsc{GetExecutionLogs}(a_i, t_i^{\text{fail}})$
  $\mathcal{O}_i \leftarrow \textsc{GetOutcomeLog}(a_i, t_i^{\text{fail}})$
  $T_i \leftarrow \{\mathcal{P}_i^{\text{Applied}}, \mathcal{P}_i^{\text{pred}}, \mathcal{L}_i, \mathcal{O}_i\}$
end for
/* Aggregate and share group-level experience */
$S \leftarrow \bigcup_{a_j \in \mathcal{G}} T_j$
for $a_i \in \mathcal{G}$ **do**
  /* Reflection: analyze shared experience */
  $\Delta_i \leftarrow \textsc{Reflect}(a_i;\ S)$ // agent-specific evolution directives
  /* Evolution: generate framework-level patches */
  $\pi_i' \leftarrow \textsc{Evolve}(a_i;\ \Delta_i)$
  /* Acting: evaluate updated agent */
  $a_i' \leftarrow \textsc{ApplyPatch}(a_i, \pi_i')$
  $\textsc{ActAndEvaluate}(a_i';\ \mathcal{T})$
  $\mathcal{G}' \leftarrow \mathcal{G}' \cup \{a_i'\}$
  /* Archive update */
  if $\textsc{Compiles}(a_i')$ **and** $\textsc{BasicCodingFunc}(a_i')$ **then**
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{a_i'\}$
  end if
end for
return $\mathcal{G}', \mathcal{A}$

while diverging through complementary adaptations to its own codebase, enabling the group to explore diverse evolutionary directions while leveraging experiences from one another.

For each agent a_i, the shared group-level experience is fed into its reflection module, which analyzes these traces and produces evolution directives targeting the agent's workflow, tool usage, or prompting strategies. These directives are then passed to the evolution module to generate framework-level patches. Finally, the updated agent is evaluated on downstream programming tasks via the action module. Agents that compile successfully and exhibit basic coding functionality are retained and added to the archive for future evolution.

Applying this process to each agent in the parent group yields an offspring group of size K. This group-level evolution iterates in an open-ended manner, as summarized in Algorithm 2.

4. Experiments

4.1. Benchmarks

Following the evaluation protocol established by the state-of-the-art open-ended self-evolving system, Darwin Gödel Machine (DGM) (Zhang et al., 2025a), we evaluate GEA on two structurally distinct benchmarks to assess its coding capabilities in both repository-level software engineering and multi-language code synthesis settings. To mitigate the substantial cost of evaluating every evolved agent on the full benchmarks, we adopt a staged evaluation strategy where agents must pass smaller subsets before advancing.

SWE-bench. We evaluate on **SWE-bench Verified** (OpenAI, 2024), a curated, human-validated subset of SWE-bench in which every task is confirmed to be solvable. We employ a three-stage evaluation process. First, agents undergo a sanity check on a small set of 10 tasks to discard those with framework-level failures (i.e., cannot compile or solve 0 out of 10 tasks). Agents that demonstrate basic coding functionality are then evaluated on the 50-task *Verified-Mini* set (Hobbhahn, 2025), which is designed to preserve a similar difficulty and pass-rate distribution as the full benchmark with fewer samples. Finally, the top-2 performing agents from this stage are evaluated on the full SWE-bench Verified dataset.

Polyglot. Polyglot (Gauthier, 2024a;b) assesses algorithmic code generation across diverse languages (C++, Rust, Java, etc.) and serves as an out-of-domain generalization test, since it is rarely used for training or fine-tuning coding models (Zhang et al., 2025a). We report pass@1 performance. Unlike SWE-bench Verified (OpenAI, 2024), group-level evolution is conducted exclusively on a 10-task small set. Agents achieving a success rate above 40% are subsequently evaluated on a separate, unseen 50-task medium set, ensuring that the larger set remains a strict out-of-distribution test for generalization.

4.2. Experimental Settings

For both SWE-bench Verified and Polyglot, we set the group size to $K = 2$ and use $M = 4$ nearest neighbors in the KNN-based parent group selection. At each iteration, a parent group of size 2 produces an offspring group of the same size.

For SWE-bench Verified, we run group-evolution for 30 iterations. Due to computational budget constraints, during the first 20 iterations, the evolving and acting modules are powered by Claude Haiku 4.5 (Anthropic, 2025a), while the final 10 iterations use Claude Sonnet 4.5 (Anthropic, 2025b). The reflection module is consistently powered by GPT-o1 across all iterations (Jaech et al., 2024). For Polyglot, we run group-evolution for 20 iterations. The first 10 iterations use Claude Haiku 4.5 (Anthropic, 2025a) for evolving and acting module, and the remaining 10 iterations use Claude Sonnet 4.5 (Anthropic, 2025b). The reflection module is again consistently powered by GPT-o1 (Jaech et al., 2024).

4.3. Baselines

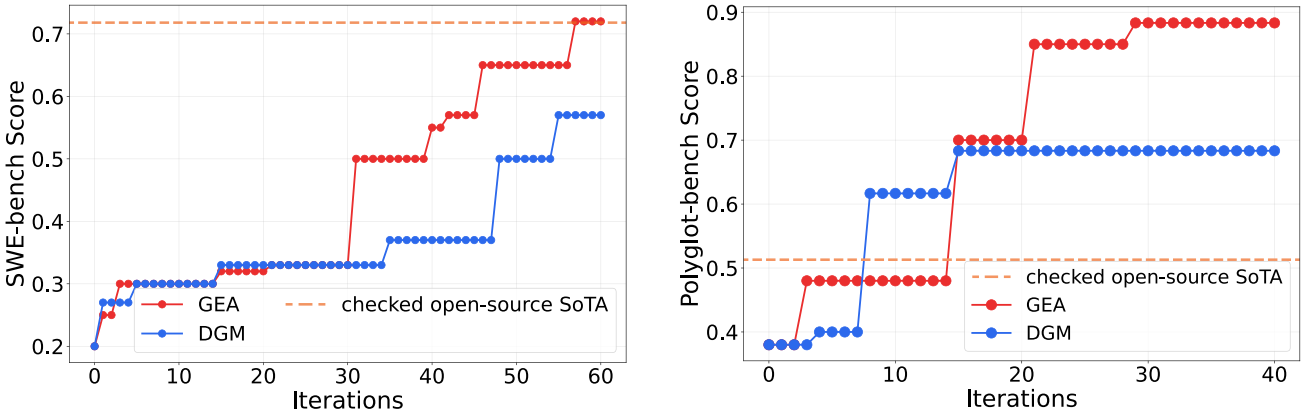
We use state-of-the-art open-ended self-evolving agents as our primary baseline for comprehensive comparison, to systematically examine how experience sharing affects diversity utilization and performance improvement during open-ended evolution, as well as robustness to framework-level perturbations. Additionally, we compare the final performance of GEA, which uses a meta-learning approach without any human intervention, against state-of-the-art human-designed coding agents.

Open-Ended Self-Evolving Baseline. We compare against the current state-of-the-art open-ended self-evolving framework, implemented following DGM (Zhang et al., 2025a). Unlike GEA, this baseline enforces a strict tree-structured evolution, where only a single agent a_i is selected as the parent and independently evolves into one child agent at each iteration. Evolutionary experience is not shared across different evolutionary branches.

Specifically, the reflection module of agent a_i receives only its own evolutionary traces, including:

(i) code modification patches applied to the agent's framework;

(ii) a predicted task patch for a randomly sampled unsolved task;



(a) GEA vs. DGM (Self-evolving Baseline) on SWE-bench

(b) GEA vs. DGM (Self-evolving Baseline) on Polyglot

Figure 3. **Performance comparison between GEA and DGM (self-evolving baseline) on two coding benchmarks.** Under the same number of evolved agents, GEA exhibits substantially larger performance gains than DGM on both SWE-bench and Polyglot, demonstrating the improved efficiency of group-level evolution.

(iii) the corresponding execution log, including tool invocation history and workflow; and

(iv) the evaluation outcome, exposing failure modes and improvement directions.

This design prevents experience reuse across evolutionary branches, resulting in a strictly individual-centric evolutionary process.

For SWE-bench Verified (OpenAI, 2024), we run this baseline for 60 iterations in total: the evolution and coding modules are powered by Claude Haiku 4.5 (Anthropic, 2025a) for the first 40 iterations and Claude Sonnet 4.5 (Anthropic, 2025b) for the final 20 iterations. For Polyglot, we run the baseline for 40 iterations: Claude Haiku 4.5 (Anthropic, 2025a) for the first 20 iterations and Claude Sonnet 4.5 (Anthropic, 2025b) for the remaining 20. In all baseline experiments, the reflection module is consistently powered by GPT-o1 (Jaech et al., 2024). To ensure fair comparison, we intentionally run the baseline for twice as many iterations as GEA so that the total number of evolved agents is comparable across methods, ensuring all comparisons are conducted under matched model schedules.

Human-Designed Frameworks. We additionally compare against state-of-the-art human-designed frameworks on both benchmarks. The top-performing, open-scaffold, checked entry on SWE-bench Verified is OpenHands + GPT-5 (Wang et al.; Singh et al., 2025), achieving 71.8%, where "checked" indicates that the SWE-bench team successfully reproduced the reported patch generations (Jimenez et al., 2024; Zhang et al., 2025a). For Polyglot, which was originally used to evaluate Aider (Gauthier, 2024b;a) by its developers, we compare against Aider, a widely adopted coding agent under continuous development and testing by human developers. The state-of-the-art performance is Aider + GPT-5 (high) (Gauthier, 2024b; Singh et al., 2025), achieving a 52.0% pass@1 success rate.

5. Results and Analysis

5.1. Main Results

GEA vs. State-of-the-Art Open-Ended Self-Evolving Systems. As shown in Figure 3, GEA demonstrates substantial performance improvements over the DGM (self-evolving baseline) on both SWE-bench Verified and Polyglot. On SWE-bench Verified, GEA improves performance from 20.0% to 71.0%, while under the same number of evolved agents, the DGM baseline achieves only 56.7%. On Polyglot, GEA boosts performance from 38.2% to 88.3%, significantly outperforming the DGM baseline (68.3%). Notably, GEA exhibits faster and more pronounced improvement in the mid-to-late stages of evolution compared to DGM (Zhang et al., 2025a), one potential reason is that the archive has accumulated sufficient diverse evolutionary directions by this point, that can be progressively consolidated and reused, leading to more rapid and pronounced performance gain, aligning with Figure 4. We discuss this phenomenon in detail in Section 5.2.

GEA vs. State-of-the-Art Human-Designed Agents. GEA achieves performance comparable to or exceeding state-of-the-art human-designed agents on both benchmarks: 71.0% vs. 71.8% on SWE-bench Verified, and 88.3% vs. 52.0% on Polyglot. Using meta-learning without any human intervention, GEA automatically evolves agent frameworks that match or surpass carefully engineered human designs (Gauthier, 2024a; Singh et al., 2025; Wang et al.), demonstrating the potential and viability of fully

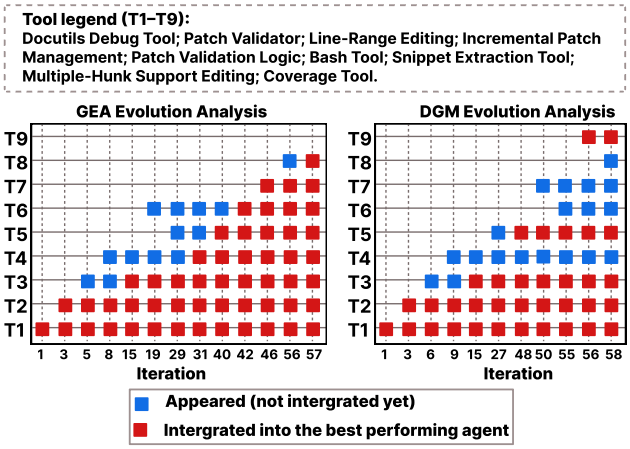


Figure 4. **Evolution analysis of tool discovery and integration over iterations.** Each row (T1–T9) corresponds to a key tool-level functionality. Blue markers indicate tools that have been discovered but not yet integrated into the current best agent, while red markers indicate tools integrated into the best-performing agent.

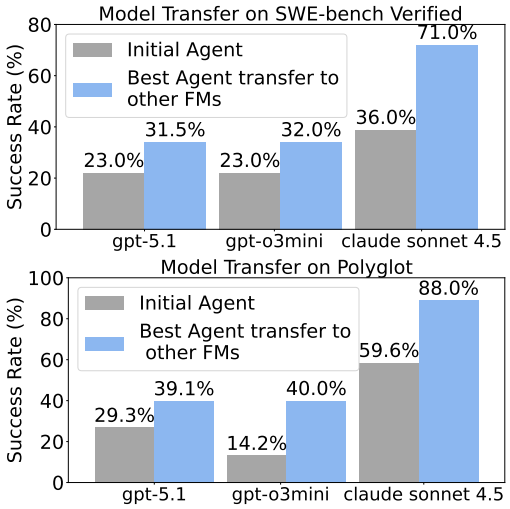


Figure 5. **Model transfer results on both benchmarks.** Across all coding models, the GEA best agent consistently outperforms the corresponding initial (iteration–0) agent, demonstrating that the improvements induced by group-level evolution generalize across different underlying model backbones.

autonomous agent improvement.

Analysis of evolutionary patterns on two benchmarks. GEA achieves state-of-the-art open-ended self-improvement on both benchmarks, with particularly strong performance on Polyglot (Gauthier, 2024a;b). We further analyze and observe that the two benchmarks differ in task complexity: Polyglot (Gauthier, 2024a;b) primarily requires agents to modify a single file from scratch to resolve tasks, involving lower editing complexity without multi-file coordination. In contrast, SWE-bench Verified (OpenAI, 2024; Jimenez et al., 2024) typically requires coordinated modifications across multiple files, demanding that agents understand inter-file dependencies and locate relevant files for coordinated edits.

This difference in complexity leads to distinct evolutionary patterns. On Polyglot (Gauthier, 2024a;b), meta-learning produces larger, more concentrated patches: each iteration yields substantial performance gains, reaching 88.3% in just 4 iterations with a total of 8,677 lines of code added. On SWE-bench Verified (OpenAI, 2024; Jimenez et al., 2024), the evolved patches are smaller and more distributed, requiring 8 iterations to reach 71.0% with 9,663 lines of code added. These observations suggest that GEA adapts its evolutionary behavior to varying task complexity, demonstrating the flexibility and generality of group-level meta-learning across different problem settings.

5.2. Evolution Analysis

Overall, our analysis shows that GEA can efficiently consolidate tool-level innovations discovered across the agents, rather than letting them remain isolated in separate evolutionary branches. Figure 4 summarizes nine key tool-level modifications on agents' framework that drove improvements. GEA integrated eight of these functionalities into its best agent, whereas the best DGM agent integrated only five. Crucially, the four tools missing from the DGM agent *were* explored in isolated branches (e.g., **T4** at iteration 9) but failed to propagate due to lineage isolation. In contrast, GEA systematically consolidated these dispersed capabilities; five of its integrated tools originated from different parent agents, confirming that explicit experience sharing prevents beneficial innovations from dying out.

To quantify this consolidation, we track the number of unique *ancestor agents* contributing to the final solutions (Table 1). The best GEA agent integrates experiences from 17 unique ancestors (28.3% of the population)—nearly double that of the best DGM agent (9 ancestors).

This broader integration correlates directly with population-wide quality. As shown in Table 1, we report the *worst-case* performance among the top-k agents. Notably, the **worst** of GEA's top-5 agents achieves 58.3%, which strictly outperforms the **single best** agent produced by DGM (56.7%). This confirms that GEA does not merely produce outliers, but systematically elevates the entire population by effectively consolidating complementary and diverse evolutionary paths. More broadly, this efficient experience consolidation suggests that GEA may exhibit stronger evolutionary capabilities in environments with greater diversity. Since open-ended evolution naturally leads to a monotonically growing diversity in the archive, GEA not only achieves stronger performance as evolution progresses but also gains enhanced self-improvement capabilities, consistent with the results shown in Figure 3.

	Worst-Case Success ↑		Ancestor Count ↑	
Rank	**DGM**	**GEA**	**DGM**	**GEA**
Top-1	56.7%	**71.0%**	9 (15.0%)	**17** (28.3%)
Top-3	48.3%	**63.3%**	13 (21.7%)	**17** (28.3%)
Top-5	45.0%	**58.3%**	16 (26.7%)	**18** (30.0%)

Table 1. **Comparison of performance (Success Rate) and ancestor integration across the Top-k agents on SWE-bench Verified**. Performance is reported as the *worst-case (minimum) success rate* among the top-k agents. *Ancestor Count* denotes the count of unique historical agents integrated into the solution. Notably, the worst-case performance of GEA's top-5 agents (58.3%) exceeds the single best agent from DGM (56.7%).

Method	E1	E2	E3	E4	E5	**Avg.** ↓
DGM	5	4	5	6	5	5.0
GEA (Ours)	1	1	2	1	2	**1.4**

Table 2. **Robustness to framework-level bugs.** We report the number of evolution iterations required to repair injected bugs across five independent trials (E1–E5). Each entry denotes the number of iterations needed for the agent to successfully repair the injected framework-level bug in a given trial. **Lower means** that GEA repairs bugs significantly **faster** than DGM.

5.3. Transferability

We evaluate the generalization and model transferability of the best-discovered agent produced by GEA. Specifically, we replace the coding model used in the acting module with different GPT-series and Claude-series models during benchmark evaluation. We then compare the performance of the initial(iteration–0) agent and the GEA best-discovered agent under each coding model.

As illustrated in Figures 5, GEA's best-discovered agent driven by GPT-series and Claude-series models consistently achieve higher performance than their corresponding initial agents on both SWE-bench and Polyglot benchmarks. This indicates that the improvements induced by group-evolving persist across different backbone models.

Further analysis reveals that all performance-improving patches discovered during GEA evolution, including those from the best agent and the top-3 performing agents, primarily target the agent's workflow and tool usage rather than model-specific prompting, details can be found in Table 3 in Appendix.

These findings together with Figures 5 demonstrate that although GEA leverages a specific backbone model to drive evolution, it discovers agent-level improvements that are largely model-agnostic and the evolved agents could generalize across different coding models.

5.4. Robustness

To evaluate the robustness of GEA, we introduce framework-level bugs by manually injecting errors into agent implementations. Specifically, we randomly select an agent from the GEA archive and manually inject framework-level bugs into its implementation. We then form a group consisting of this faulty agent and another bug-free agent from its original parent group, and perform group evolution to assess whether GEA can leverage experience from the better-performing agent (i.e., the one without framework-level bugs) to repair the faulty one. For comparison, under the self-evolution setting, the bugged agent evolves independently without access to external experience sharing. In both settings, we measure the number of iterations required to successfully repair the bug.

As shown in Table 2, across five independent trials, GEA requires only 1.4 iterations on average to repair the injected bugs, whereas the self-evolving baseline (DGM) requires 5. This substantial gap demonstrates that group-evolving agents benefit from intra-group experience sharing, enabling successful framework-level experiences from better-performing agents to guide the repair of faulty ones, confirming the robustness of the group-evolving paradigm.

6. Conclusion

We introduce **G**roup-**E**volving **A**gents (GEA), a new paradigm for open-ended self-improvement that treats a *group of agents*, rather than an individual agent, as the fundamental unit of evolution. By enabling explicit experience sharing and reuse within the group, agents can learn from each other's evolutionary experiences and adaptively integrate complementary improvements throughout evolution.

Compared to individual-centric self-evolving approaches, GEA more effectively consolidates valuable exploratory outcomes from early stages into the best-performing agents, efficiently transforming transient diversity into long-term useful experience. As a result, group-level evolution achieves substantially stronger performance given the same number of evolved agents.

Further analysis shows that GEA's improvements primarily stem from enhancements to agent workflows and tool usage, rather than overfitting to a specific coding model. Therefore, its gains transfer consistently across different models, including both GPT-series and Claude-series.

In addition, GEA exhibits stronger robustness than individual-centric self-evolving approaches: through group-level experience reuse, better-performing agents can guide the repair of faulty ones, enabling GEA to recover from framework-level bugs with fewer evolution iterations.

Impact Statement

GEA demonstrates the potential and viability of group-evolving open-ended systems to autonomously modify their own implementation for continuous improvement. While this potential aligns with the goal of building AI that benefits humanity, open-ended exploration also carries inherent considerations worth noting. For instance, the evolutionary process may inadvertently introduce directions misaligned with human intent while consuming substantial computational resources, or produce patches that lack structural clarity, leading to increasingly complex systems that are difficult to fully understand. Therefore, it is essential to establish appropriate boundaries and guide the system to preserve exploratory diversity while ensuring alignment with human intent. Following Zhang et al. (2025a), all experiments in this work are conducted in isolated sandbox environments, thereby limiting potential impacts on host systems.

On the other hand, although we focus on evolving agents' coding capabilities in this work, this paradigm has broader potential applications, for example, enabling systems to mitigate biases through self-improvement, thereby becoming more trustworthy and beneficial for social good.

References

Anthropic. Claude haiku 4.5. https://www.anthropic.com/news/claude-haiku-4-5, 2025a. Accessed: 2025-11-10.

Anthropic. Claude sonnet 4.5. https://www.anthropic.com/news/claude-sonnet-4-5, 2025b. Accessed: 2025-11-10.

Chatzilygeroudis, K., Cully, A., Vassiliades, V., and Mouret, J.-B. 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.

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

Faldor, M., Zhang, J., Cully, A., and Clune, J. Omni-epic: Open-endedness via models of human notions of interestingness with environments programmed in code. In *The Thirteenth International Conference on Learning Representations*.

Fang, J., Peng, Y., Zhang, X., Wang, Y., Yi, X., Zhang, G., Xu, Y., Wu, B., Liu, S., Li, Z., et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.

Fernando, C., Banarse, D., Michalewski, H., Osindero, S., and Rocktäschel, T. Promptbreeder: self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 13481–13544, 2024.

Gao, H.-a., Geng, J., Hua, W., Hu, M., Juan, X., Liu, H., Liu, S., Qiu, J., Qi, X., Wu, Y., et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Gauthier, P. Aider: Ai pair programming in your terminal. https://github.com/Aider-AI/aider, 2024a. Accessed: 2025-11-15.

Gauthier, P. o1 tops aider's new polyglot leaderboard. https://aider.chat/2024/12/21/polyglot.html, December 2024b. Accessed: 2025-11-10.

Gou, Z., Shao, Z., Gong, Y., Shen, Y., Yang, Y., Huang, M., Duan, N., and Chen, W. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*, 2023.

Hobbhahn, M. Swe-bench verified mini. https://github.com/mariushobbhahn/SWEBench-verified-mini, April 2025. Accessed: 2025-11-16.

Hua, W., Yang, X., Jin, M., Li, Z., Cheng, W., Tang, R., and Zhang, Y. Trustagent: Towards safe and trustworthy llm-based agents through agent constitution. In *Trustworthy Multi-modal Foundation Models and AI Agents (TiFA)*, 2024.

Huang, C., Yu, W., Wang, X., Zhang, H., Li, Z., Li, R., Huang, J., Mi, H., and Yu, D. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

Huang, J., Gu, S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 1051–1068, 2023.

Hughes, E., Dennis, M., Parker-Holder, J., Behbahani, F., Mavalankar, A., Shi, Y., Schaul, T., and Rocktaschel, T. Open-endedness is essential for artificial superhuman intelligence. *arXiv preprint arXiv:2406.04268*, 2024.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Jiang, M., Rocktäschel, T., and Grefenstette, E. General intelligence requires rethinking exploration. *Royal Society Open Science*, 10(6):230539, 2023.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. Swe-bench: Can language models resolve real-world github issues? In *12th International Conference on Learning Representations, ICLR 2024*, 2024.

Mouret, J.-B. and Clune, J. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Ni, A., Allamanis, M., Cohan, A., Deng, Y., Shi, K., Sutton, C., and Yin, P. Next: Teaching large language models to reason about code execution. *arXiv preprint arXiv:2404.14662*, 2024.

Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J., Mehrabian, A., et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

OpenAI. Introducing swe-bench verified. https://openai.com/index/introducing-swe-bench-verified/, August 2024. Accessed: 2025-11-10.

Polo, F. M., Weber, L., Choshen, L., Sun, Y., Xu, G., and Yurochkin, M. tinybenchmarks: evaluating llms with fewer examples. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 34303–34326, 2024.

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

Robeyns, M., Szummer, M., and Aitchison, L. A self-improving coding agent. *arXiv preprint arXiv:2504.15228*, 2025.

Rodriguez, P., Barrow, J., Hoyle, A. M., Lalor, J. P., Jia, R., and Boyd-Graber, J. Evaluation examples are not equally informative: How should that change nlp leaderboards? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4486–4503, 2021.

Schmidhuber, J. Gödel machines: self-referential universal problem solvers making provably optimal self-improvements. *arXiv preprint cs/0309048*, 2003.

Setlur, A., Nagpal, C., Fisch, A., Geng, X., Eisenstein, J., Agarwal, R., Agarwal, A., Berant, J., and Kumar, A. Rewarding progress: Scaling automated process verifiers for llm reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Shapere, D. The structure of scientific revolutions. *The Philosophical Review*, 73(3):383–394, 1964.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Garcia, X., Liu, P. J., Harrison, J., Lee, J., Xu, K., et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*.

Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Song, Y., Zhang, H., Eisenach, C., Kakade, S. M., Foster, D., and Ghai, U. Mind the gap: Examining the self-improvement capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*.

Stanley, K. O. and Lehman, J. Why greatness cannot be planned: The myth of the objective. *(No Title)*, 2015.

Stanley, K. O., Lehman, J., and Soros, L. Open-endedness: The last grand challenge you've never heard of. *While open-endedness could be a force for discovering intelligence, it could also be a component of AI itself*, 2017.

Sun, H., Zhuang, Y., Kong, L., Dai, B., and Zhang, C. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36:58202–58245, 2023.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

Wang, S., Jiao, Z., Zhang, Z., Peng, Y., Ze, X., Yang, B., Wang, W., Wei, H., and Zhang, L. Socratic-zero: Bootstrapping reasoning via data-free agent co-evolution. *arXiv preprint arXiv:2509.24726*, 2025a.

Wang, X., Li, B., Song, Y., Xu, F. F., Tang, X., Zhuge, M., Pan, J., Song, Y., Li, B., Singh, J., et al. Openhands: An open platform for ai software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*.

Wang, Y., Liu, S., Fang, J., and Meng, Z. 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, 2025b.

Wang, Z., Xu, H., Wang, J., Zhang, X., Yan, M., Zhang, J., Huang, F., and Ji, H. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. *arXiv preprint arXiv:2501.11733*, 2025c.

Wei, Y., Sun, Z., McMilin, E., Gehring, J., Zhang, D., Synnaeve, G., Fried, D., Zhang, L., and Wang, S. Toward training superintelligent software agents through self-play swe-rl. *arXiv preprint arXiv:2512.18552*, 2025.

Xin, H., Ren, Z., Song, J., Shao, Z., Zhao, W., Wang, H., Liu, B., Zhang, L., Lu, X., Du, Q., et al. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*, 2024.

Yin, X., Wang, X., Pan, L., Lin, L., Wan, X., and Wang, W. Y. 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.

Yuan, W., Pang, R. Y., Cho, K., Li, X., Sukhbaatar, S., Xu, J., and Weston, J. E. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*, 2024.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zelikman, E., Lorch, E., Mackey, L., and Kalai, A. T. Self-taught optimizer (stop): Recursively self-improving code generation. In *First Conference on Language Modeling*, 2024.

Zhang, J., Lehman, J., Stanley, K., and Clune, J. Omni: Open-endedness via models of human notions of interestingness. In *The Twelfth International Conference on Learning Representations*.

Zhang, J., Hu, S., Lu, C., Lange, R., and Clune, J. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025a.

Zhang, Y., Duan, Y., Zhang, Z., He, J., and Zheng, S. Population-evolve: a parallel sampling and evolutionary method for llm math reasoning. *arXiv preprint arXiv:2512.19081*, 2025b.

Zhao, A., Wu, Y., Yue, Y., Wu, T., Xu, Q., Lin, M., Wang, S., Wu, Q., Zheng, Z., and Huang, G. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

A. Appendix

A.1. Cost Estimate

The primary cost for both GEA and DGM arises from benchmark evaluation. Since we generate the same number of agents for both methods, their overall costs are very similar. Following the settings described in Sections 4.2 and 4.3, the estimated cost of completing a full run is approximately USD 13,000 per method on SWE-bench and USD 1,500 on Polyglot. A more detailed estimated cost breakdown is provided below:

Coding Model	Benchmark	Number of Tasks	Cost Estimate (USD)
Claude Sonnet 4.5	SWE-bench	60	$370
Claude Sonnet 4.5	Polyglot	60	$60
Claude Haiku 4.5	SWE-bench	60	$120
Claude Haiku 4.5	Polyglot	60	$20

A.2. Case Study

Agent	Patch Description	Δ Score
Top-1	docutils_debug	+0.10
	Modified core agent logic	+0.10
	Updated tests logic	+0.22
	Add snippet_extract tool	+0.10
	Updating edit tools (v1)	+0.01
	Updating edit tools (v2)	+0.05
	Updating edit tools (v3)	+0.02
Top-2	docutils_debug	+0.10
	Modified core agent logic	+0.10
	Updated tests logic	+0.22
	Add snippet_extract tool	+0.10
	Add bash tools	+0.01
	Updating edit tools	+0.07
Top-3	docutils_debug	+0.10
	Modified core agent logic	+0.10
	Updated tests logic	+0.22
	Add multi-file tools (v1)	+0.01
	Add multi-file tools (v2)	+0.07

Table 3. **Evolutionary trajectories of the top-3 performing agents discovered by GEA on SWE-bench Verified.** Across all three agents, the performance-improving patches primarily focus on enhancing the agents' workflows and tool usage, rather than relying on model-specific prompting strategies.