AGENTIC ESOPT: FINE-TUNING LONG-HORIZON LLM AGENTS WITH MINIMAL GPU REQUIREMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Adapting long-horizon LLM agents to specific tasks requires efficient and flexible optimization under limited GPU resources. Although Reinforcement Learning (RL) has shown promise in single-turn LLM fine-tuning, this setting exposes two challenges: backpropagation makes full-parameter adaptation increasingly expensive for larger LLMs and longer contexts, while sparse rewards and branching interactions complicate long-horizon credit assignment. We argue that evolution strategies (ES) can often be a better choice for such **task-specific adaptation**, offering **1) Model Scalability** through forward-only full-parameter optimization, **2) Flexibility** to integrate parameter updates with prompt-space optimization, and **3) Long-Horizon Scalability** by avoiding explicit turn-level credit decomposition. Based on these insights, we propose Agentic ESOpt, a full-parameter ES framework supporting both train-time fine-tuning and on-the-fly adaptation during agentic test-time compute, together with a cosine perturbation schedule for balancing exploration and adaptation. Agentic ESOpt outperforms the strongest Agentic RL baseline by 13.19 percentage points in our longest-horizon Sudoku experiment, improves over Agentic GRPO on multi-turn tool-usage Math and DocVQA, enables full-parameter fine-tuning on a MoE LLM Qwen3-30B-A3B with 4×80GB GPUs, and improves existing automatic heuristic design pipelines in 28 of 36 comparisons. Code is available at `https://anonymous.4open.science/status/Agentic-ESOpt-anonymous-197C`.

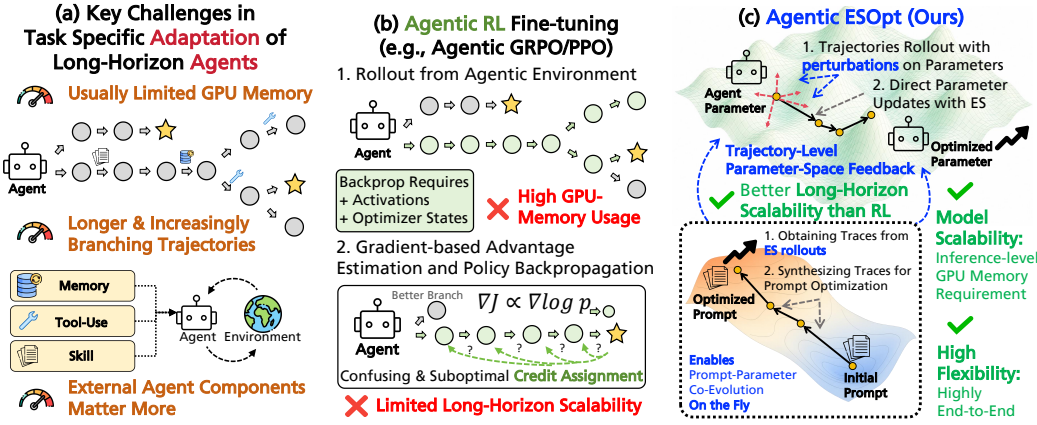


Figure 1: (a) Long-horizon agentic reasoning introduces new challenges for Agentic RL (b), including high GPU memory requirements in training and difficult credit assignment across horizons. Agentic ESOpt (c) addresses these issues through inference-level GPU memory, high flexibility, and better long-horizon scalability compared to Agentic RL.

1 INTRODUCTION

Advanced Large Language Models (LLMs) such as Qwen3, DeepSeek-R1, and Gemini 2.5 have demonstrated strong capabilities as general-purpose agents (Yang et al., 2025; Guo et al., 2025; Gemini Team, 2025). They can navigate websites (Zhou et al., 2024), edit software repositories

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.

(Yang et al., 2024), and interact with external tools. However, general-purpose agents can still struggle with uncommon APIs (Ma et al., 2024) and specialized scientific or algorithmic tasks (Liu et al., 2024). Therefore, **efficiently adapting LLM agents to task-specific environments under limited GPU resources remains important** (Du et al., 2026).

Reinforcement learning (RL) has become a strong approach for LLM fine-tuning from scalar feedback, particularly in single-turn reasoning (Shao et al., 2024; Liu et al., 2025c). Recent work has also explored evolution strategies (ES) (Salimans et al., 2017) for single-turn LLM fine-tuning (Qiu et al., 2026; Sun et al., 2026; Sarkar et al., 2025), demonstrating that forward-only optimization can provide a memory-efficient alternative to gradient-based training. However, **task-specific parameter adaptation of long-horizon agents** presents a different regime, exposing fundamental scalability and optimization challenges for Agentic RL. **1)** Long interaction histories increase the activation and systems burden of backward-based training, while full-parameter RL additionally requires gradients and optimizer states. **2)** Meanwhile, sparse trajectory-level rewards must guide increasingly long and branching sequences of decisions, making credit assignment substantially harder (Kim et al., 2026). These observations motivate a central question:

What fine-tuning method is well suited to task-specific adaptation of long-horizon LLM agents?

This paper first argues that ES can be a particularly attractive choice in this setting. Compared with Agentic RL, ES offers three key advantages:

- **1) Model Scalability:** Forward-only ES avoids activation storage and optimizer-state overhead, substantially reducing the GPU-memory barrier to full-parameter fine-tuning of larger LLMs.
- **2) Flexibility:** ES provides a lightweight end-to-end fine-tuning pipeline requiring only forward execution and scalar feedback, making parameter adaptation of larger LLM agents easy to compose with prompt-space evolution methods, such as test-time compute (Liu et al., 2024) and skill optimization (Yang et al., 2026).
- **3) Long-Horizon Scalability:** ES directly attributes trajectory-level feedback to parameter perturbations, avoiding the need to assign sparse rewards to individual actions or turns, which can become unreliable in long-horizon Agentic RL.

Based on these insights, we propose **Agentic ESOpt**, a full-parameter agentic fine-tuning framework using ES. Agentic ESOpt evaluates a population of G perturbed agents with scalar feedback and applies online reward-weighted parameter updates. Utilizing this flexibility, Agentic ESOpt can interleave parameter updates with test-time search or train-time skill optimization, enabling parameter–context co-optimization on the fly. Moreover, we introduce a cosine decay schedule for the perturbation scale σ: train-time optimization retains a nonzero terminal σ_T for regularization, whereas test-time optimization decays σ_T to zero for finer local adaptation.

We evaluate Agentic ESOpt across controlled long-horizon Sudoku, multi-turn ReAct-style tool use on Math and DocVQA, WebArena, and agentic test-time compute, using LLMs ranging from 4B to 30B. Our results reveal a clear horizon-dependent crossover between the performance of ES and Agentic RL on Sudoku, demonstrate consistent gains over Agentic GRPO on multi-turn tasks, and demonstrates full-parameter adaptation to Qwen3.5-27B and an MoE LLM, Qwen3-30B-A3B, in a limited memory environment. In agentic test-time compute, Agentic ESOpt further improves existing search procedures through online parameter adaptation. Moreover, a preliminary population-sensitivity study suggests that stronger LLMs may require fewer perturbation directions for effective ES adaptation. Our contributions are as follows:

- We demonstrate that ES can often be better suited the **long-horizon adaptation regime** for Agents, revealing a horizon-dependent performance crossover with Agentic RL and preliminary evidence that the population requirement of ES may decrease as model capability increases.
- Agentic ESOpt introduces an **cosine-annealed perturbation schedule**, using cosine-decayed perturbations with setting-dependent terminal scales to balance exploration and local adaptation across train-time and test-time optimization.
- We show that Agentic ESOpt is able to implement **on-the-fly parameter adaptation** for parameter–context co-optimization, skill optimization, and agentic test-time compute in task-specific environments with limited GPU resources.

2 PRELIMINARIES: AGENTIC LLM REASONING

We define a multi-turn LLM agent that repeatedly observes the environment and produces an action (can be a response vector) as follows:

$$\boldsymbol{a_t} \sim \pi_\theta(\boldsymbol{a_t} \mid \boldsymbol{o}_{\le t}, c_t),$$

where θ denotes model parameters, $\boldsymbol{o}_{\le t}$ is the interaction history, and c_t is an external prompt, memory, skill, or tool instruction. An episode induces a multi-turn trajectory $\boldsymbol{\tau} = (\boldsymbol{o_1}, \boldsymbol{a_1}, \ldots, \boldsymbol{o_H}, \boldsymbol{a_H})$, where the **horizon** H is the number of agent–environment turns before termination. Its return is $R(\boldsymbol{\tau}) = \sum_{t=1}^{H} \gamma^{t-1} r_t$. In many agentic reasoning tasks, rewards are sparse: intermediate rewards are typically zero, $r_t = 0$ for $t < H$, and only the completed trajectory receives a task score. In some problems (e.g., MATH & DocVQA ReAct-style Tool Usage), the agent receives the complete multi-turn trajectory $\boldsymbol{\tau}$ as input, but in some problems with (partial) Markov properties (Sudoku, WebArena), the agent only needs to receive partial local input (e.g., $\boldsymbol{o_t}$ only).

The optimization of the Task-specific agent can act on c_t or θ. Prompt-space optimization methods optimize c_t while freezing the LLM. They are lightweight, but can only elicit behaviors already accessible to the frozen policy (Ni et al., 2026; Liu et al., 2024). Policy fine-tuning methods optimize θ for better agentic ability with Supervised Fine-Tuning (SFT), On-Policy Distillation (OPD), Group Relative Policy Optimization (GRPO), or Proximal Policy Optimization (PPO).

Agentic SFT & OPD. Agentic SFT learns from expert actions (Chu et al., 2025), while Agentic OPD learns from token distributions of higher-ability LLMs (Song & Zheng, 2026). Both methods require labels beyond the scalar environment reward, which requires additional cost, so we exclude them from our comparison scope.

Agentic PPO & GRPO. Agentic PPO and Agentic GRPO require only a scalar environment reward, making them applicable to many agentic settings (Qi et al., 2025). Agentic GRPO (Shao et al., 2024) samples G trajectories for the same task and assigns trajectory i the group-relative advantage

$$\widehat{A}_i = \frac{R_i - \mathrm{mean}_j(R_j)}{\mathrm{std}_j(R_j) + 10^{-5}}. \tag{1}$$

Let $q_{i,t}(\theta) = \pi_\theta(\boldsymbol{a_{i,t}} \mid h_{i,t}) / \pi_{\theta_{\mathrm{old}}}(\boldsymbol{a_{i,t}} \mid h_{i,t})$, where $h_{i,t} = (\boldsymbol{o}_{\le t}, c_t)$. Agentic GRPO minimizes

$$\mathcal{L}_{\mathrm{GRPO}} = -\frac{1}{G} \sum_{i=1}^{G} \sum_{t=1}^{H_i} \min\Big(q_{i,t} \widehat{A}_i, \mathrm{clip}(q_{i,t}, 1-\epsilon, 1+\epsilon) \widehat{A}_i \Big) + \beta D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}). \tag{2}$$

As shown in Figure 5, recent studies (Feng et al., 2025; He et al., 2026a) note that trajectory-level group-relative advantages might lead to wrong credit assignment within multi-turn trajectories. So, PPO has also been adopted for multi-turn agentic RL (Li et al., 2026); it uses the same clipped surrogate but replaces $\widehat{A}_i$ with a turn-level advantage estimated by a critic (Schulman et al., 2017):

$$\mathcal{L}_{\mathrm{PPO}} = -\sum_{t=1}^{H} \min\Big(q_t \widehat{A}_t, \mathrm{clip}(q_t, 1-\epsilon, 1+\epsilon) \widehat{A}_t \Big), \quad \widehat{A}_t = \sum_{l=0}^{H-t} (\gamma\lambda)^l \big(r_{t+l} + \gamma V_\phi(h_{t+l+1}) - V_\phi(h_{t+l}) \big). \tag{3}$$

PPO cannot fully eliminate the long-horizon difficulty: its critic requires warm-up, sparse terminal rewards make early value estimates unreliable, and its policy gradient still accumulates H action-level score terms. Appendix C.3 analyzes the resulting horizon-dependent component. These credit-assignment limitations motivate the parameter-space ES formulation of Agentic ESOpt.

3 METHODOLOGY: AGENTIC ESOPT

As illustrated in Figure 2, Agentic ESOpt performs full-parameter ES optimization by sampling parameter perturbations around the current LLM θ, evaluating the perturbed agents with scalar environment rewards, and applying a reward-weighted parameter update. This update is forward-only, which only requires storing the noise seed and using in-place addition and subtraction (Qiu et al., 2026), so the GPU memory requirement of Agentic ESOpt remains the same as inference. Moreover, the same trajectory feedback can be reused by test-time compute or skill-space optimizers. Formally, let $\boldsymbol{\tau} = (\boldsymbol{o_1}, \boldsymbol{a_1}, \ldots, \boldsymbol{o_H}, \boldsymbol{a_H})$ denote an interaction trajectory induced by policy π_θ, and let $R(\boldsymbol{\tau})$ denote its scalar reward. For a fixed external agent prompt c, the objective is as follows:

$$J(\theta; c) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_\theta(\cdot|c)} \left[R(\boldsymbol{\tau}) \right].$$

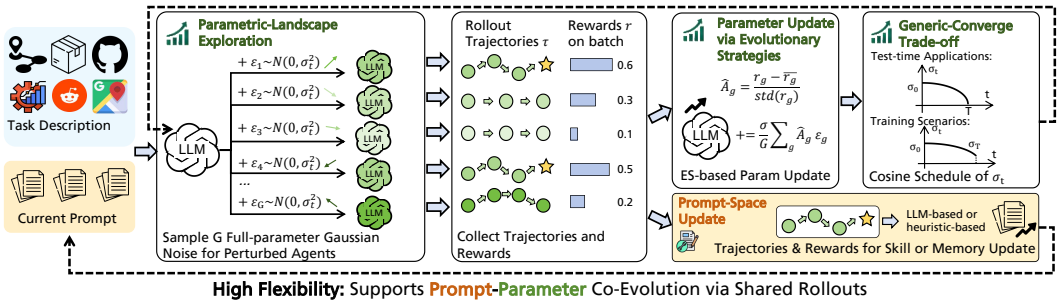


Figure 2: **Detailed workflow of Agentic ESOpt.** Starting from the current LLM, Agentic ESOpt samples parameter perturbations, evaluates the perturbed agents in the environment, normalizes their scalar rewards, and applies a reward-weighted ES update. Compared with Agentic RL, Agentic ESOpt provides **model scalability**, **optimization flexibility**, and **long-horizon scalability**.

Agentic ESOpt optimizes the objective by searching over the parameter space around θ, where $\boldsymbol{\epsilon} \in \mathbb{R}^d$ is a full-parameter perturbation on the d-dimensional parameters, and the Gaussian-smoothed objective is as:

$$J_\sigma(\theta; c) = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}\left[J(\theta + \sigma\boldsymbol{\epsilon}; c)\right]. \tag{4}$$

Then we can derive the ES pseudo-gradient as follows (Salimans et al., 2017; Qiu et al., 2026):

$$\nabla_\theta J_\sigma(\theta; c) = \frac{1}{\sigma}\mathbb{E}_{\boldsymbol{\epsilon}}\left[J(\theta + \sigma\boldsymbol{\epsilon}; c)\boldsymbol{\epsilon}\right]. \tag{5}$$

The derivation is detailed in Appendix C.1. The ES gradients are estimated from scalar scores without differentiating through the agent–environment interaction.

In implementation, to estimate equation 5, Agentic ESOpt samples G perturbations $\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_G$, evaluates the corresponding perturbed agents, and obtains rewards $R_i = R(\boldsymbol{\tau}_i)$. To reduce variance, we normalize the rewards within the population with a z-score as follows:

$$\hat{R}_i = \frac{R_i - \mu_R}{s_R + 10^{-5}}, \quad \text{where} \quad \mu_R = \frac{1}{G}\sum_{j=1}^{G} R_j, \quad s_R^2 = \frac{1}{G}\sum_{j=1}^{G}(R_j - \mu_R)^2. \tag{6}$$

In practice, as analyzed in Appendix C.1, our normalized update omits the explicit $1/\sigma$ factor in equation 5, with α serving as the effective update scale. The implemented update is:

$$\theta_{t+1} = \theta_t + \frac{\alpha}{G}\sum_{i=1}^{G}\hat{R}_i\boldsymbol{\epsilon}_i. \tag{7}$$

We follow Qiu et al. (2026) in storing only the noise seed of each perturbation and using in-place addition and subtraction for each perturbed agent, so Agentic ESOpt requires only the amount of GPU memory in inference.

Prompt-Space Composition and Prompt-Parameter Co-Evolution. Test-time compute and prompt-space optimization methods generally keep the LLM parameters fixed throughout the search process, **which may limit optimization when solving the task requires changing the underlying policy itself**. Existing approaches (He et al., 2026b; Huang et al., 2026) that jointly update the policy and prompt-space components typically require specialized algorithmic designs and tightly coordinated optimization schedules, making them difficult to scale to large LLMs in limited GPU resources. In contrast, the lightweight Agentic ESOpt allow parameter adaptation to proceed directly alongside prompt-space optimization, enabling simple **prompt–parameter co-evolution**.

This composition relies on a mild *trajectory-compatibility assumption*: parameter perturbations are sufficiently local that the perturbed agents remain task-competent and their trajectories retain meaningful behavioral information for the external prompt optimizer. Thus, the same perturbed rollouts used to estimate the ES update can also serve as optimization experience for the external context. Formally, at iteration t, we sample and collect trajectories from perturbed LLMs as follows:

$$\mathcal{D}_t = \{(\boldsymbol{\tau_i}, R_i)\}_{i=1}^{G}, \qquad \boldsymbol{\tau_i} \sim \pi_{\theta_t+\sigma_t\epsilon_i}(\cdot \mid c_t), \qquad \epsilon_i \sim \mathcal{N}(0, I),$$

Let $\mathcal{U}_{\text{ES}}$ denote the Agentic ESOpt parameter update and $\mathcal{U}_c$ an external update rule for context c_t. Since $\mathcal{D}_t$ remains informative for both optimization spaces, a general alternating outer loop can reuse the same rollouts:

$$\theta_{t+1} = \mathcal{U}_{\text{ES}}(\theta_t; c_t, \mathcal{D}_t), \qquad c_{t+1} = \mathcal{U}_c(c_t; \mathcal{D}_t).$$

3.1 Cosine Decay of the Perturbation Radius σ

With perturbations $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, Agentic ESOpt effectively optimizes the Gaussian-smoothed objective in equation 4. However, a nonzero perturbation radius introduces a smoothing bias, whose leading term is characterized as:

Lemma 1 (Gaussian-smoothing bias)**.** *Assume $J(\theta; c)$ is sufficiently smooth in a neighborhood of θ. Then*

$$J_\sigma(\theta; c) = J(\theta; c) + \frac{\sigma^2}{2}\mathrm{Tr}\big(\nabla_\theta^2 J(\theta; c)\big) + O(\sigma^4).$$

The proof is in Appendix C.2. The second-order term $\mathrm{Tr}(\nabla_\theta^2 J)$ can be viewed as a **regularization term**. For a maximization objective, it penalizes sharp local optima while favoring flatter parameter neighborhoods. Therefore, a larger σ introduces stronger regularization but also a larger bias from the original objective.

Existing ES methods for single-turn LLM fine-tuning typically use a fixed perturbation radius throughout optimization (Qiu et al., 2026), without explicitly adapting this trade-off. Agentic ESOpt instead gradually decreases σ over T update steps, using a larger radius early for broader exploration and stronger regularization, and a smaller radius later for reduced objective bias as follows:

$$\sigma_t = \sigma_T + (\sigma_0 - \sigma_T)\frac{1 + \cos(\pi t/T)}{2}, \qquad t = 0, \ldots, T.$$

So, for **train-time Agentic ESOpt**, we retain a nonzero σ_T to balance exploitation with exploration and regularization. Instead, **Test-time compute** focuses on the outcome of the current task instead of the generalization of the agent, so we decay σ_T to zero to minimize the objective bias toward the end of optimization. Since $\sigma_T = 0$, we do not update the LLM in the final step of test-time compute.

3.2 Long-Horizon Scalability of Agentic ESOpt

To study the **long-horizon scalability** of Agentic ESOpt relative to Agentic RL, we compare the horizon-dependent variance contributions of their canonical estimators.

A Variance Perspective on Long-Horizon Adaptation. For a trajectory $\boldsymbol{\tau}$ with terminal return and normalized with equation 6 $\hat{R}(\boldsymbol{\tau})$. Following Salimans et al. (2017, Sec. 3.1), as derived in Appendix C.3, a policy-gradient estimator and the corresponding ES estimator are:

$$\widehat{g}_{\mathrm{RL}} = \hat{R}(\boldsymbol{\tau}) \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(\boldsymbol{a_t}|h_t), \qquad \mathrm{Var}[\widehat{g}_{\mathrm{RL}}] \approx \mathrm{Var}[\hat{R}(\boldsymbol{\tau})] \underbrace{\mathrm{Var}\Big[\sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(\boldsymbol{a_t}|h_t)\Big]}_{\propto H \text{ under the stated assumptions}}. \tag{8}$$

$$\widehat{g}_{\mathrm{ES}} = \frac{1}{G}\sum_{i=1}^{G} \hat{R}_i \boldsymbol{\epsilon}_i, \qquad \mathrm{Var}[\widehat{g}_{\mathrm{ES}}] \approx \mathrm{Var}[\hat{R}(\boldsymbol{\tau})]\frac{1}{G}\mathrm{Var}\left[\boldsymbol{\epsilon}\right].$$

Here $\mathrm{Var}[\cdot]$ denotes total vector variance. Both $\mathrm{Var}[\widehat{g}_{\mathrm{RL}}]$ and $\mathrm{Var}[\widehat{g}_{\mathrm{ES}}]$ have the respective normalized-return variance terms $\mathrm{Var}[\hat{R}(\boldsymbol{\tau})]$. After factoring it out, the remaining variance in the PG score is $\mathrm{Var}[\sum_{t=1}^{H} u_t]$. Under regular on-policy sampling, cross-turn score covariances vanish; if the per-turn score covariance remains comparable, this term grows approximately linearly with H. In contrast, ES uses $\mathrm{Var}[\boldsymbol{\epsilon}]$ for the entire rollout, without explicit horizon dependence.

Thus, the comparison isolates an additional horizon-dependent score-variance component in policy gradients that is absent from ES. Appendix C.3 gives the detailed derivation and conditions. This provides an optimization-level rationale for applying ES to long-horizon agents, while Section 4.1 evaluates their relative adaptation performance across increasing minimum successful horizons.

4 Experiments

This section evaluates the proposed Agentic ESOpt in three settings. Section 4.1 evaluates Agentic ESOpt on **Sudoku with controlled horizons**, providing empirical evidence on the long-horizon scalability shown in Section 3.2. To show the practical task-specific adaptation ability, Section 4.2 evaluates Agentic ESOpt on **ReAct-based Tool Use**. Section 4.3 tests the Agentic ESOpt in cooperating **agentic test-time compute**, evaluating the on-the-fly parameter optimization ability.

Table 1: Agentic Sudoku final evaluation success rate ($\times 100$) and GPU memory requirement, grouped by minimum successful horizon H^*. Values are reported across three evaluations. To show that this observation holds across independent trainings, we train Agentic GRPO‡, Agentic ESOpt, and Vanilla ES with independent seeds three times and further average their mean and std.

Method	GPU Mem Req.	$H^* = 5$	$H^* = 10$	$H^* = 15$
Qwen3.5-27B	51.75GB	86.46 ± 3.90	50.00 ± 2.55	28.13 ± 2.55
Qwen3.5-4B	**8.41GB**	63.54 ± 7.80	31.25 ± 4.42	10.42 ± 1.47
+ Agentic PPO*	109.34GB	$\mathbf{90.63 \pm 0.00}$	56.25 ± 0.00	0.00 ± 0.00
+ Agentic GRPO†	58.88GB	80.21 ± 1.47	44.79 ± 2.95	30.21 ± 2.95
+ Agentic GRPO‡	58.88GB	85.42 ± 1.47	$\mathbf{67.71 \pm 1.47}$	41.67 ± 2.32
+ Agentic GiGPO‡	58.88GB	86.46 ± 1.47	61.46 ± 3.9	31.25 ± 0
+ Agentic ESOpt (G=32)	**8.41GB**	89.58 ± 2.95	62.50 ± 2.55	$\mathbf{54.86 \pm 2.19}$
w/o σ decay (Vanilla ES)	**8.41GB**	85.42 ± 3.90	55.21 ± 5.89	46.88 ± 3.00
w/o σ_T (i.e., $\sigma_T = 0$)	**8.41GB**	85.42 ± 3.90	54.17 ± 3.90	28.13 ± 2.55

†GRPO training uses temperature 0.7, top-$p = 0.8$, and top-$k = 20$. ‡GRPO, ‡ GiGPO, and *PPO training use temperature 1, top-$p = 1$, and top-$k = -1$. All GRPO, GiGPO, and PPO evaluations, as well as Vanilla ES and Agentic ESOpt training and evaluation rollouts, use temperature 0.7, top-$p = 0.8$, and top-$k = 20$.

4.1 Agentic Sudoku: Experiments on Long-Horizon Scalability

Controlled Long-Horizon Sudoku. In multi-turn Sudoku, each valid action fills at most one cell, and the environment provides only a terminal reward. Masking 5, 10, or 15 cells sets the minimum successful horizon to $H^* \in \{5, 10, 15\}$, where $H^*(x) = \min_{\tau: R(\tau)=1} |\tau|$. Increasing the mask count scales both the missing information and the number of correct decisions required for success. At each scale, all methods face the same puzzles, action interface, terminal reward, and interaction budget. We therefore compare how ES and Agentic RL adapt to increasingly demanding decision sequences within a common task family. The real horizon H may exceed H^* due to wrong actions.

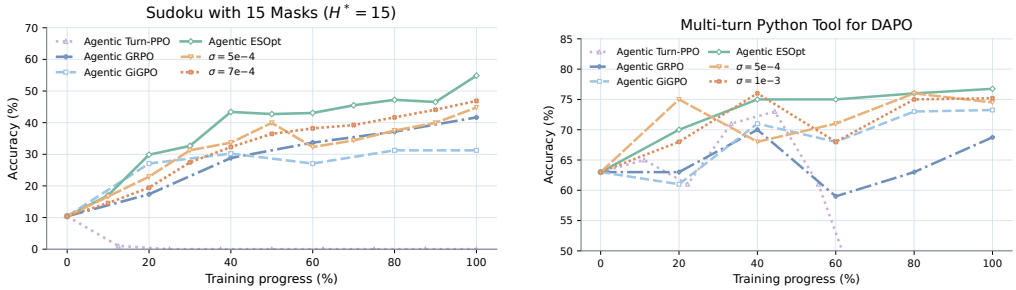
Baselines. We adopt a same set of RL baselines across Section 4.1 and Section 4.2 as follows: **1) Agentic GRPO** with 8 rollouts (Shao et al., 2024), **2) Agentic Turn-PPO** (Li et al., 2026), **3) Agentic Group-in-Group Policy Optimization (GiGPO)** with also 8 rollouts (Feng et al., 2025). For 30B LLMs that are not applicable to full-parameter RL on four 80GB GPUs, we include **4) GRPO+LoRA** (Hu et al., 2022) with rank=32.

Experiment Settings. We implement baselines and Agentic ESOpt on 4×H100 80GB GPUs. Each H^* uses disjoint 32-instance training and evaluation sets. We report task success—solving the full puzzle within the interactive budgets—as the mean and standard deviation over three evaluations. Details are in Appendix D.1, and repeated-run experiments are in Appendix D.1.2. As shown in Appendix C.5, we approximately match the total FLOPs between baselines and Agentic ESOpt and observe the change in performance as the horizon changes. To align the FLOPs between Agentic ESOpt and Agentic GRPO under the same number of training steps, we set $G = 32$ as the default.

Observation 1: Agentic ESOpt Exhibits a Horizon-Dependent Performance Crossover. Table 1 shows that the relative ranking changes across the Sudoku settings. PPO achieves the highest success rate at $H^* = 5$ (90.63%), and GRPO leads at $H^* = 10$ (67.71%). At $H^* = 15$, Agentic ESOpt achieves the strongest result at 54.86%, exceeding the strongest GRPO baseline by 13.19 percentage points; Vanilla ES reaches 46.88%, while PPO fails to solve the evaluation puzzles. These results establish a performance crossover within this task family, with ES-based adaptation becoming particularly competitive in the longest-horizon setting. Appendix D.1.1 further characterizes the associated advantage-sign patterns and realized trajectory lengths.

Observation 2: Agentic ESOpt can Converge Faster. We plot the Sudoku evaluation curves with $H^* = 15$ in Figure 3(a). Agentic GRPO improves slowly at the beginning of training, while PPO quickly collapses. The advantage-sign and trajectory-length diagnostics in Appendix D.1.1 are consistent with long-horizon credit assignment contributing to this behavior.

Observation 3: Agentic ESOpt Demonstrates High GPU Memory Efficiency. As shown in the column GPU Mem Req. of Table 1, when considering the minimal possible GPU requirement (e.g., no KV Cache or CUDA graph, and overlooking activation for RL methods due to short responses in Sudoku), Agentic ESOpt requires only inference-level GPU memory. This efficiency makes full-parameter fine-tuning of larger agents practical on limited GPU resources (e.g., 4×80GB GPUs).



(a) $H^* = 15$ evaluation curve (b) Multi-Turn Python Tool for Agentic MATH

Figure 3: Evaluation curves of Agentic ESOpt and baselines on (a) Sudoku with $H* = 15$. (b) Multi-Turn Python Tool for Agentic MATH.

Table 2: Train-time Math-reasoning and DocVQA results, including full-parameter adaptation of the Qwen3-30B-A3B MoE. Accuracies are reported in %. Mean@4 averages four sampled Pass@1 scores; Pass@4 or Max@4 represents the best of four samples.

		Math Reasoning				DocVQA			
		DAPO		AIME 2026		ANLS		Accuracy	
Model	Method	Mean@4	Pass@4	Mean@4	Pass@4	Mean@4	Max@4	Mean@4	Pass@4
Qwen3.5-4B	Base LLM	63.0	**86.0**	55.8	86.7	0.3875	0.5981	40.3	62.0
Qwen3.5-4B	Agentic GRPO	68.8	83.0	58.3	76.7	0.4627	0.5398	48.0	56.0
Qwen3.5-4B	Agentic GiGPO	73.3	84.0	65.0	83.3	0.4846	0.5654	49.8	62.0
Qwen3.5-4B	Agentic ESOpt	**76.8**	**86.0**	**70.8**	**96.7**	**0.5043**	**0.6507**	**52.5**	**67.0**
	Δ vs base	↑13.8	0.0	↑15.0	↑10.0	↑0.1168	↑0.0526	↑12.2	↑5.0
Qwen3-30B-A3B	Base LLM	40.5	67.0	18.3	30.0	0.2030	0.3987	21.8	43.0
Qwen3-30B-A3B	Agentic GRPO+LoRA	37.8	68.0	21.7	40.0	0.2975	0.5068	31.0	53.0
Qwen3-30B-A3B	Agentic ESOpt	**50.0**	**74.0**	**28.3**	**46.7**	**0.3780**	**0.5190**	**40.8**	**56.0**
	Δ vs base	↑9.5	↑7.0	↑10.0	↑16.7	↑0.1750	↑0.1203	↑19.0	↑13.0

4.2 Adaptation to Long-Horizon ReAct-Style Tool Usage on Math & DocVQA

To demonstrate the ability in task-specific adaptation towards long-horizon scenarios, in this section we evaluate Agentic ESOpt in fine-tuning LLMs for ReAct-style tool usage (Yao et al., 2022). We consider two multi-turn tool-use settings as follows: **(1) Agentic Math with Python**, where the agent iteratively executes Python programs for computation and verification before returning a final answer; and **(2) Agentic DocVQA with OCR**, where the agent invokes OCR and image processing programs to inspect a document image before answering. In both settings, the action space consists of a parameterized Bash tool call and a terminal text answer action, with textual tool outputs appended as observations each turn. These experiments are widely adopted in recent works as challenging long-horizon adaptation settings (Zhang et al., 2026; Ni et al., 2026; Yu et al., 2025b).

Experiment Settings. We follow the setting in Ni et al. (2026), using 400 math problems from the DAPO Yu et al. (2025a) dataset as the training set, while taking 100 held-out DAPO problems and 30 out-of-distribution AIME 2026 problems as the testing set. For DocVQA, we adopt 50 training examples and 100 held-out problems from the DocVQA dataset. We compare baselines mentioned in 4.1 with full-parameter Agentic ESOpt (setting $G = 16$) on **Qwen3.5-4B** and further test Agentic ESOpt on **Qwen3-30B-A3B**, a sparse MoE with 30.5B total parameters and 3.3B activated per token (Yang et al., 2025). We also control the total FLOPs between Agentic ESOpt and baselines. Complete settings are in Appendix D.3. For both tasks, we set the maximum number of turns to 50. **We recognize both tasks as long-horizon scenarios.** The Qwen3.5-4B Base LLM consumes 26.0 turns in DAPO, 20.9 turns in AIME2026, and 19.9 turns in DocVQA.

Observation 1: Agentic ESOpt Consistently Outperforms Matched Agentic RL. As shown in Table 2, on Qwen3.5-4B, Agentic ESOpt improves the base model by an average of 13.7%, demonstrating the best performance across DAPO, AIME 2026, and DocVQA.Agentic GiGPO usually performs the second best, while Agentic ESOpt shows clear improvements on three datasets. On the 30B LLM, Agentic ESOpt improves the DAPO and AIME 2026 Mean@4 scores by +9.5% and +10.0%, DocVQA ANLS by +0.1750, and DocVQA Mean@4 by +19.0%. Agentic GRPO + LoRA can boost the performance on AIME2026 and DocVQA, but it fails to improve on DAPO.

Observation 2: Model Scalability: Agentic ESOpt can Full-Parameter Fine-Tune a 30B MoE LLM. Moreover, ESOpt provides clear performance gains for the **Qwen3-30B-A3B** MoE LLM with full-parameter optimization, improving all eight metrics. As analyzed in Appendix C.7, full-parameter fine-tuning with the RL methods is not possible under our GPU memory constraints, so we only compare with GRPO+LORA. We also evaluate Agentic ESOpt in WebArena Zhou et al. (2024) in Appendix D.5 with Qwen3.5-27B and get similar observations.

Observation 3: Agentic ESOpt might better preserve repeated-sampling coverage: Agentic ESOpt improves every reported Pass@4 metric over 4B GRPO and every Pass@4 or Max@4 metric over the 30B MoE base model. Full 4B Pass@K profiles up to $k = 32$ are in Appendix D.4.4 (Yue et al., 2025), suggesting broader coverage of successful trajectories after parameter optimization.

4.3 Agentic ESOpt on External Components: Test-Time Compute with Automatic Heuristic Design and Skills Adaptation

Beyond parameter-space-only fine-tuning, the flexibility of Agentic ESOpt makes it possible to do on-the-fly parameter updates with optimization over external agent components (e.g., Memory, Skill). RL requires complex engineering; the end-to-end nature of Agentic ESOpt offers advantages, particularly with limited GPU. This section evaluates this capability with a test-time search scenario, automatic heuristic design (AHD) (Liu et al., 2024), where Agentic ESOpt performs on-the-fly parameter updates alongside heuristic search with memory Liu et al. (2024). Besides, in Appendix D.5, we conduct on-the-fly parameter updates in train-time adaptation with SkillOpt Yang et al. (2026).

Experiment Settings. In constructive AHD, the heuristic makes local decisions for NP-hard combinatorial optimization problems (defined in Appendix C.4): TSP, KP, and ASP. We adopt Agentic ESOpt to boost the performance of a representative test-time compute work for AHD, EoH (Liu et al., 2024). EoH improves the performance of these algorithms by maintaining a population (i.e., fixed-size memory) of high-quality heuristic algorithms (evaluated on a fixed training dataset) and applying crossover and mutation operators over them. In generating each new heuristic, it provides LLMs with problem descriptions; the LLM generates heuristic code, and the evaluator returns a scalar objective after executing the code within a solver. Agentic ESOpt can be inserted into existing EoH without changing their search scaffold, achieving both heuristic-space search and parameter-space optimization. We design heuristics for the constructive pipeline and compare EoH with the corresponding +Agentic ESOpt under the same evaluation budgets. Details are in Appendix C.4.

Implementation of Agentic ESOpt. For Agentic ESOpt + EoH, the reward is the selected parent's cost minus the perturbed child's cost. Agentic ESOpt preserves the original EoH operators and attaches parameter updates only to the mutation operators $m1$ and $m2$. For Agentic ESOpt + Sample, the reward is the negative child cost. Rewards are z-score normalized within each parameter-update batch. We report the best candidate from the final generation and average over three repeated runs.

Table 3: Design constructive heuristics at total evaluation $T = 1000$. TSP is minimized; KP and ASP are maximized. The *Optimal* row reports the known optimum $x^\star$ only as a reference. We define the normalized optimality gap as $g(x) = \frac{|x-x^\star|}{|x^\star|}$, and report the gain of an Agentic ESOpt result m over its matched baseline b as $\Delta = \frac{g(b)}{g(m)} - 1$. *LLaMA-3.1-8B-Instruct* is used for all runs.

Method	TSP $N = 20$	TSP $N = 50$	KP $N = 50, W = 12.5$	KP $N = 100, W = 25$	ASP $N = 15, W = 10$	ASP $N = 21, W = 15$
Optimal	3.8199	5.6750	20.0370	40.2710	3,003	43,596
Greedy Construct	4.4797	6.9590	19.9850	40.2250	1,530	15,050
Total Evaluations: $T = 1000$						
EoH	4.2481	6.5450	19.9958	40.2320	2,760	28,465.67
Agentic ESOpt + EoH	4.2170	6.4631	20.0007	40.2351	2,770	30,887.33
Δ vs EoH	↓7.83%	↓10.39%	↑13.59%	↑8.65%	↑4.29%	↑19.06%

Observation 1: The On-the-Fly Parameter-Update of Agentic ESOpt can Improve Performance As shown in Table 3, at $T = 1000$, Agentic ESOpt + EoH improves all six constructive test sets. Experiments under more AHD scenarios are reported in Appendix D.6.2. In total, Agentic ESOpt improves 28 of 36 matched method–budget comparisons across 12 test sets and six scenarios. A 20-run significance analysis on TSP and KP settings is provided in Table 22.

Table 4: WebArena-Lite Dataset Avg. success rates (%). Results are the mean ± standard deviation over three runs. Detailed Settings and Results are in Appendix D.5.

Methods	GPT-5.4	Qwen3.5-27B	Qwen3.5-27B + Trace2Skill	Qwen3.5-27B + Trace2Skill + Agentic ESOpt	Qwen3.5-27B + SkillOpt	Qwen3.5-27B + SkillOpt + Agentic ESOpt
Success Rates	34.14 ± 0.76	29.49 ± 1.14	33.94 ± 2.76	$\mathbf{36.36 \pm 0.86}$	28.89 ± 0.57	$\mathbf{34.75 \pm 0.76}$

Observation 2: On-the-Fly Parameter Updates Are Also Effective for Skill Optimization. In Appendix D.5, we further evaluate Agentic ESOpt together with SkillOpt Yang et al. (2026) and Trace2Skill Ni et al. (2026) on WebArena-Lite, where parameter updates are performed alongside skill optimization. As seen in Table 4, combining SkillOpt with Agentic ESOpt improves the average success rate of the dataset by $+5.90\%$. This result suggests that the benefit of on-the-fly parameter adaptation is not specific to test-time heuristic search, but can also extend to train-time skill-space optimization, supporting the broader applicability of parameter–context co-optimization.

5 Discussion

5.1 Ablation on the Perturbation Schedule

The σ Schedule Plays an Important Role in Performance. As shown in Figure 3, the cosine perturbation schedule consistently optimizes better than fixed-σ ES on both long-horizon Sudoku and Agentic Math. A relatively large fixed σ maintains stronger smoothing but can hinder fine-grained adaptation in later stages, whereas a smaller fixed σ may restrict exploration too early. By gradually decreasing σ, Agentic ESOpt combines broader early exploration with more precise late-stage adaptation. Appendix D.6.2 further validates the schedule in the test-time setting.

5.2 Population Scaling with Model Size

Stronger LLMs may Allow Smaller G. Population size G is a key hyperparameter. As Table 5, for the 4B LLM, increasing the population from $G = 8$ to $G = 16$ on the same 15-turn Sudoku setting increases a lot while the 9B and 27B LLMs are less sensitive. As shown in Figure 4, one plausible interpretation is that a pre-trained stronger LLM will lead to a more competent local region, so perturbations are more likely to be informative for the update. This interpretation is consistent with the finding of Gan & Isola (2026) that useful behavioral diversity becomes denser around stronger pre-trained LLMs. This supports the prospect of using Agentic ESOpt for **fast task-specific adaptation with larger LLMs**; we will consider establishing a universal scaling law as future work.

Table 5: Vanilla-ES population sensitivity on 15-turn Sudoku, $G = 16$ with $G = 8$.

Backbone	G	Best test	Final test	Δ best test	Δ final test
Qwen3.5-4B	8	5.10	2.95	–	–
Qwen3.5-4B	16	35.42	22.92	+594.5%	+677.0%
Qwen3.5-9B	8	30.21	30.21	–	–
Qwen3.5-9B	16	37.50	30.21	+24.1%	0.0%
Qwen3.5-27B	8	51.03	51.03	–	–
Qwen3.5-27B	16	51.03	51.03	+0.0%	0.0%

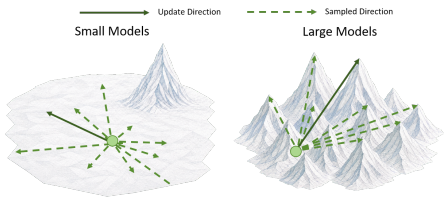


Figure 4: Intuition for population scaling: sampled directions around a stronger backbone are more likely to align with a useful update direction.

6 Conclusion

This paper presents Agentic ESOpt, a full-parameter ES framework for long-horizon LLM agents. Agentic ESOpt demonstrates **model scalability**, **flexibility**, and **long-horizon scalability** that are preferred in task-specific long-horizon agent adaptations. It avoids explicit horizon-wise action-score accumulation while retaining inference-level memory use and compatibility with prompt-space optimization. Moreover, the proposed cosine-annealed perturbation schedule is also effective. Experiments show advantages of Agentic ESOpt in long-horizon agentic adaptation tasks from a 4B dense model to the 30B Qwen3-30B-A3B sparse MoE, and Agentic ESOpt can also be applied for on-the-fly parameter adaptation in test-time compute and skill optimization. As detailed in Appendix A.1 and A.2, limitations include expensive evaluation scenarios, and future work includes combining quantized optimization scenarios and frontier LLMs.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs, implement methods, and create or edit software code. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat datasets, or interpret results, and qualitative and thematic data analysis is not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, create artifacts, and edit the research paper to improve readability.

We have reviewed all AI-assisted work. AI-assisted proofs were checked by the authors for correctness, and AI-assisted implementations, software code, figures, images, artifacts, and manuscript edits were reviewed and tested as appropriate before inclusion. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For reproducibility, we additionally provide an anonymous source-code release at `https://anonymous.4open.science/status/Agentic-ESOpt-anonymous-197C`. Detailed experimental configurations, optimization hyperparameters, rollout budgets, evaluation protocols, computational accounting, and additional results for Sudoku, Math, DocVQA, WebArena-Lite, and automatic heuristic design are provided in Appendix D. Appendix E reports the prompts and task interfaces used in the agentic environments, while Appendix F documents external resources, repositories, and licensing information.

REFERENCES

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Shangheng Du, Jiabao Zhao, Jinxin Shi, Zhentao Xie, Xin Jiang, Yanhong Bai, and Liang He. A survey on the optimization of large language model-based agents. *ACM Computing Surveys*, 58 (9):1–37, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2025.

Zihang Fu, Haonan Wang, Jian Kang, Kenji Kawaguchi, and Jiaying Wu. Reasoning resides in layers: Restoring temporal reasoning in video-language models with layer-selective merging. *arXiv preprint arXiv:2604.11399*, 2026.

Yulu Gan and Phillip Isola. Neural thickets: Diverse task experts are dense around pretrained weights. *arXiv preprint arXiv:2603.12228*, 2026.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Yu Gu, Zhi Zheng, Yunpeng Ba, Xialiang Tong, Mingxuan Yuan, and Zhenkun Wang. Hyper-es: Effective evolution strategies for llm reasoning via descent direction merging. *arXiv preprint arXiv:2608.05541*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

Shuo He, Lang Feng, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations*, volume 2026, pp. 27572–27593, 2026a.

Zelin He, Haotian Lin, Boran Han, Wei Zhu, Haoyang Fang, Bernie Wang, Xuan Zhu, Runze Li, and Matthew Reimherr. Reskill: Reconciling skill creation with policy optimization in agentic rl. *arXiv preprint arXiv:2606.01619*, 2026b.

Zhenyu Hou, Yujiang Li, Jie Tang, and Yuxiao Dong. Single-rollout asynchronous optimization for agentic reinforcement learning. *arXiv preprint arXiv:2607.07508*, 2026.

William Hoy, Binxu Wang, and Xu Pan. Matching accuracy, different geometry: Evolution strategies vs grpo in llm post-training. *arXiv preprint arXiv:2604.01499*, 2026.

EJ Hu, Y Shen, P Wallis, Z Allen-Zhu, Y Li, S Wang, L Wang, and W Chen. Lora: Low-rank adaptation of large language models. iclr. *arXiv preprint arXiv:2106.09685*, 2022.

Ziyao Huang, Weiwei Wu, Kui Wu, Wei-Bin Lee, and Jianping Wang. Calm: Co-evolution of algorithms and language model for automatic heuristic design. In *International Conference on Learning Representations*, volume 2026, pp. 72468–72510, 2026.

Sunghwan Kim, Junhee Cho, Beong-woo Kwak, Taeyoon Kwon, Liang Wang, Nan Yang, Xingxing Zhang, Furu Wei, and Jinyoung Yeo. On training large language models for long-horizon tasks: An empirical study of horizon length. *arXiv preprint arXiv:2605.02572*, 2026.

Daria Korotyshova, Boris Shaposhnikov, Alexey Malakhov, Alexey Khokhulin, Nikita Surnachev, Kirill Ovcharenko, George Bredis, Alexey Gorbatovski, Viacheslav Sinii, and Daniil Gavrilov. Essa: Evolutionary strategies for scalable alignment. *arXiv preprint arXiv:2507.04453*, 2025.

Junbo Li, Peng Zhou, Rui Meng, Meet P Vadera, Lihong Li, and Yang Li. Turn-ppo: Turn-level advantage estimation with ppo for improved multi-turn rl in agentic llms. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6227–6243, 2026.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024.

WenTao Liu, Siyu Song, Hao Hao, and Aimin Zhou. Ea4llm: A gradient-free approach to large language model optimization via evolutionary algorithms. *arXiv preprint arXiv:2510.10603*, 2025a.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Song XiXuan, Yifan Xu, Shudan Zhang, Hanyu Lai, Jiadai Sun, Xinyue Yang, et al. Visualagentbench: Towards large multimodal models as visual foundation agents. In *International Conference on Learning Representations*, volume 2025, pp. 95650–95707, 2025b.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025c.

Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, et al. Gem: A gym for agentic llms. *arXiv preprint arXiv:2510.01051*, 2025d.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. Spreadsheetbench: Towards challenging real world spreadsheet manipulation. *Advances in Neural Information Processing Systems*, 37:94871–94908, 2024.

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

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 79791–79821, 2025.

Xin Qiu, Yulu Gan, Conor F. Hayes, Qiyao Liang, Yinggan Xu, Roberto Dailey, Elliot Meyerson, Babak Hodjat, and Risto Miikkulainen. Evolution strategies at scale: LLM fine-tuning beyond reinforcement learning. In *International Conference on Machine Learning*, 2026.

Günter Rudolph. Evolution strategies. *Evolutionary computation*, 1:81–88, 2000.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Bidipta Sarkar, Mattie Fellows, Juan Agustin Duque, Alistair Letcher, Antonio León Villares, Anya Sims, Clarisse Wibault, Dmitry Samsonov, Dylan Cope, Jarek Liesen, et al. Evolution strategies at the hyperscale. *arXiv preprint arXiv:2511.16652*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

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

Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

Zhishen Sun, Sizhe Dang, Guang Dai, and Haishan Ye. Essam: A novel competitive evolution strategies approach to reinforcement learning for memory efficient llms fine-tuning. *arXiv preprint arXiv:2602.01003*, 2026.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning. *arXiv preprint arXiv:2602.02710*, 2026.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

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.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents. *arXiv preprint arXiv:2605.06078*, 2026.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *arXiv preprint arXiv:2504.15965*, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

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

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent–computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

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

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025a.

Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. Demystifying reinforcement learning in agentic reasoning. *arXiv preprint arXiv:2510.11701*, 2025b.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Xuqin Zhang, Quan He, Zhenrui Zheng, Zongzhang Zhang, Xu He, and Dong Li. Aster: Agentic scaling with tool-integrated extended reasoning. *arXiv preprint arXiv:2602.01204*, 2026.

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor Tsang. Second-order fine-tuning without pain for llms: A hessian informed zeroth-order optimizer. In *International Conference on Learning Representations*, volume 2025, pp. 43496–43520, 2025.

Zhi Zheng, Yu Gu, Wei Liu, Yee Whye Teh, and Wee Sun Lee. Soft-grpo: Surpassing discrete-token llm reinforcement learning via gumbel-reparameterized soft-thinking policy optimization. *arXiv preprint arXiv:2511.06411*, 2025a.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte carlo tree search for comprehensive exploration in llm-based automatic heuristic design. *arXiv preprint arXiv:2501.08603*, 2025b.

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

APPENDIX CONTENTS

A. Limitations & Future Work 15
A.1. Limitations 15
A.2. Future Work 15
B. Additional Related Work 16
B.1. Agentic Reinforcement Learning 16
B.2. Evolution Strategies 16
B.3. Traditional Evolutionary Strategies Algorithms 16
B.4. Skill and Memory-Based Self-Improvement 16
C. Theoretical Analysis and Computational Accounting 17
C.1. Derivation of the Scalar-Score ES Gradient 17
C.2. Proof of the Gaussian-Smoothing Bias Expansion 18
C.3. Long-Horizon Credit Assignment 18
C.4. AHD as Heuristic-Space Optimization 20
C.5. Training FLOPs & Clocktime Calculation 21
C.6. Systems Advantages and Scope of Forward-Only Adaptation 22
C.7. Details of GPU Memory Accounting 23
D. Experimental Settings and Additional Results 24
D.1. Sudoku 24
D.2. Math Reasoning and Skill-Conditioned Results 27
D.2.1. Train-Time Fine-Tuning Combined with Trace2Skill 30
D.3. DocVQA 29
D.4. WebArena-Lite Results and Settings 30
D.4.1. Model-Scalability and Skill-Composition Results 30
D.5. Automatic Heuristic Design 34
D.6. Cross-Setting Hyperparameter Summary 36
E. Prompts and Skills 37
E.1. Sudoku 37
E.2. Math Reasoning 37
E.3. DocVQA 38
E.4. WebArena-Lite 39
E.5. Automatic Heuristic Design 39
F. Licensing and External Components 40
F.1. Licensing Scope 40
F.2. Upstream Repositories Used by the Released Workflow 40

A LIMITATIONS & FUTURE WORK

A.1 LIMITATIONS

Introducing new hyperparameters. Compared to Agentic RL, Agentic ESOpt introduces hyperparameters for the perturbation radius σ. Their optimal values can depend on LLMs, reward distribution, and environment, which may introduce additional tuning cost compared with using a fixed training recipe. However, as summarized in Appendix D.7, we use relatively consistent configurations across tasks and LLMs, taking $\sigma_0 \approx 1e-3$ and $\alpha \approx 5e-4$ across 5 experiments. This shows that this set of parameters is highly generic. We will take the automatic schedule for σ as future work.

Concerns in Expensive Evaluation Scenarios. Agentic ESOpt trades backpropagation cost for a larger number of independent environment evaluations: under matched model FLOPs, it can afford more trajectories because each trajectory requires only a forward pass. This trade-off may become less favorable when environment evaluation itself is *extremely* expensive, in which case rollout cost rather than model computation can dominate the total budget.

Continual learning remains unclear. The current experiments establish in-setting adaptation. As mentioned in Hoy et al. (2026), unlike GRPO, which usually does sparse updates, ES will lead to a random walk in directions irrelevant to the optimization objective, raising concerns about continual learning. However, although the ES update is dense in the strict sense, its magnitude is still highly concentrated. As shown in Table 6, Agentic ESOpt updates can still maintain sparsity within a certain threshold: 96.26% of the final parameter updates remain within the perturbation scale σ, and 99.42% are smaller than 2.0×10^{-3}.

Table 6: Distribution of parameter-update magnitudes after Agentic ESOpt for WebArena on Qwen3.5-27B with perturbation scale $\sigma_t = 1.5 \times 10^{-3}$.

Update-magnitude threshold	Fraction of parameters
$\lvert\Delta\theta\rvert \le 1.0 \times 10^{-3}$	83.68%
$\lvert\Delta\theta\rvert \le 1.5 \times 10^{-3}$ $(= \sigma)$	96.26%
$\lvert\Delta\theta\rvert \le 2.0 \times 10^{-3}$	99.42%

A.2 FUTURE WORK

Scaling to advanced LLMs and population scaling laws. The inference-level memory requirement of Agentic ESOpt provides a direct path toward full-parameter adaptation of substantially larger LLM agents. Our 4B/9B results further suggest that stronger backbones may require fewer perturbation directions, motivating a systematic study of population scaling with model capability. Moreover, as future work, scaling to advanced LLMs (Team et al., 2026; Guo et al., 2025; Gemini Team, 2025) may also enable online adaptation in industrial scenarios involving proprietary tools, APIs, and complex workflows.

Quantized ES optimization. Quantization could further extend the scalability of Agentic ESOpt, but applying dense parameter perturbations directly to quantized weights requires scale-aware noise generation and numerically stable updates (Sarkar et al., 2025). Developing ES-specific infra for quantization-compatible perturbation and seed-replay mechanisms is therefore an important systems direction for large-scale ES fine-tuning.

B Additional Related Work

B.1 Agentic Reinforcement Learning

Reinforcement learning provides the standard route for optimizing model parameters from verifiable task feedback (Shao et al., 2024; Liu et al., 2025c; Yu et al., 2025a; Zheng et al., 2025a; Tajwar et al., 2026). These methods have demonstrated strong performance on single-turn LLM reasoning.

The recent agentic scenarios bring new challenges about long-context and long-horizon credit assignment (Kim et al., 2026; Liu et al., 2025d), and there are studies that try to apply RL techniques to agentic scenarios (Qi et al., 2025; Feng et al., 2025; Wang et al., 2026). Some work highlights better rollout in a GRPO-style baseline (Feng et al., 2025; He et al., 2026a), and there are also works training critic network and use PPO for better credit assignment (Hou et al., 2026). These methods can improve the credit assignment ability of RL, but we believe that considering Agentic ESOpt should be seen as a more direct solution. Gradient-based agentic RL usually requires a training stack that stores rollouts, estimates token-level policy gradients, manages reference models or KL constraints, and consumes substantial memory. Agentic ESOpt maintains the same high-level goal of parameter adaptation, but uses scalar black-box fitness so that the environment trajectory does not need to be differentiable or retained for backpropagation.

B.2 Evolution Strategies for single-turn LLM Fine-Tuning.

As advanced LLMs are equipped with larger and larger parameters, fine-tuning them on devices with moderate scales becomes unaffordable, and ES have emerged as efficient gradient-free optimizers for fine-tuning single-turn LLMs. Qiu et al. (2026) demonstrated that ES can scale to full-parameter LLM fine-tuning. This momentum has driven ES-based methods into pre-training (Liu et al., 2025a), few-shot adaptation (Gan & Isola, 2026; Korotyshova et al., 2025; Gu et al., 2026; Fu et al., 2026), and memory-efficient tuning via sharpness-aware mechanisms (Sun et al., 2026). In these methods, ES achieves higher GPU memory efficiency but lower performance than RL methods. However, we argue that the structural advantages of ES are actually pronounced in fine-tuning long-horizon agents, where Agentic ESOpt can become preferable to Agentic RL, instead of merely serving as a cheaper alternative.

As a similar category of gradient-free LLM fine-tuning, zeroth-order methods (Malladi et al., 2023; Zhao et al., 2025) focus on SFT; because there are no labels in reasoning scenarios, zeroth-order methods are hard to apply to single-turn and multi-turn reasoning scenarios.

B.3 Traditional Evolutionary Strategies Algorithms

Before being applied to LLMs, traditional ES algorithms optimize numerical parameters by sampling parameter perturbations and updating the search distribution according to their fitness (Rudolph, 2000). Classical variants include CMA-ES (Hansen & Ostermeier, 2001) and Natural Evolution Strategies (Wierstra et al., 2014). ES has long been used for neural-network optimization, with Salimans et al. (2017) scaling it to deep networks and demonstrating competitive performance with RL in control and gaming environments.

B.4 Skill and Memory-Based Self-Improvement and Test-Time Compute

A broad class of self-improving agents improves behavior by optimizing external, non-parametric components while keeping the underlying LLM fixed. Reflexion converts failed trajectories into verbal feedback that conditions later trials (Shinn et al., 2023); Voyager accumulates executable skills for open-ended embodied exploration (Wang et al., 2023); and recent memory systems organize and retrieve past experience over long-horizon interactions (Xu et al., 2025; Wu et al., 2025). More recent methods such as SkillOpt and Trace2Skill make this optimization explicit by treating the skill document itself as an optimizable object (Yang et al., 2026; Ni et al., 2026).

Test-time compute (Snell et al., 2024) methods follow a related principle: they improve task performance through additional search (Yao et al., 2023), sampling, reflection (Madaan et al., 2023), or evolution (Liu et al., 2024) at inference time, while typically leaving the model parameters unchanged. These approaches are lightweight and flexible, but their optimization remains constrained by the behaviors accessible to the frozen policy.

Agentic ESOpt complements these methods by introducing parameter adaptation as an additional optimization dimension. Its black-box update can reuse the same trajectory-level feedback already

collected for skill, memory, or test-time search, allowing external-space optimization and parameter-space optimization to proceed together. This enables Agentic ESOpt to strengthen existing self-improvement and test-time compute pipelines without replacing their original search or skill-update mechanisms.

C Theoretical Analysis and Computational Accounting

C.1 Derivation of the Scalar-Score ES Gradient

Let d be the number of LLM parameters and let

$$q_\sigma(\vartheta \mid \theta) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{\|\vartheta - \theta\|_2^2}{2\sigma^2}\right)$$

denotes the density of a Gaussian perturbation centered at θ. Equivalently, $\vartheta = \theta + \sigma\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. The smoothed objective can then be written as

$$J_\sigma(\theta; c) = \int J(\vartheta; c) q_\sigma(\vartheta \mid \theta)\, d\vartheta.$$

Assuming differentiation and integration can be interchanged, the log-derivative identity gives

$$\nabla_\theta J_\sigma(\theta; c) = \int J(\vartheta; c) \nabla_\theta q_\sigma(\vartheta \mid \theta)\, d\vartheta \tag{9}$$

$$= \mathbb{E}_{\vartheta \sim q_\sigma(\cdot \mid \theta)} \left[J(\vartheta; c) \nabla_\theta \log q_\sigma(\vartheta \mid \theta) \right]. \tag{10}$$

For the Gaussian density,

$$\nabla_\theta \log q_\sigma(\vartheta \mid \theta) = \frac{\vartheta - \theta}{\sigma^2} = \frac{\boldsymbol{\epsilon}}{\sigma},$$

and hence

$$\nabla_\theta J_\sigma(\theta; c) = \frac{1}{\sigma} \mathbb{E}_{\boldsymbol{\epsilon}} \left[J(\theta + \sigma\boldsymbol{\epsilon}; c) \boldsymbol{\epsilon} \right].$$

Now expand the inner objective using its trajectory definition:

$$J(\theta + \sigma\boldsymbol{\epsilon}; c) = \mathbb{E}_{\boldsymbol{\tau} \sim \pi_{\theta + \sigma\boldsymbol{\epsilon}}(\cdot \mid c)} \left[R(\boldsymbol{\tau}) \right].$$

The law of total expectation therefore yields

$$\nabla_\theta J_\sigma(\theta; c) = \frac{1}{\sigma} \mathbb{E}_{\substack{\boldsymbol{\epsilon} \sim \mathcal{N}(0, I) \\ \boldsymbol{\tau} \sim \pi_{\theta + \sigma\boldsymbol{\epsilon}}(\cdot \mid c)}} \left[R(\boldsymbol{\tau}) \boldsymbol{\epsilon} \right].$$

Thus, for independent perturbations $\boldsymbol{\epsilon}_i$ and conditional rollouts $\boldsymbol{\tau}_i \sim \pi_{\theta + \sigma\boldsymbol{\epsilon}_i}(\cdot \mid c)$, the canonical Monte Carlo estimator is

$$\widehat{g}_{\mathrm{ES}} = \frac{1}{G} \sum_{i=1}^{G} R(\boldsymbol{\tau}_i) \boldsymbol{\epsilon}_i, \qquad \mathbb{E}[\widehat{g}_{\mathrm{ES}}] = \frac{1}{\sigma} \nabla_\theta J_\sigma(\theta; c).$$

Because $\mathbb{E}[\boldsymbol{\epsilon}] = 0$, any perturbation-independent scalar baseline b may replace $R(\boldsymbol{\tau}_i)$ by $R(\boldsymbol{\tau}_i) - b$ without changing this expectation. Crucially, the estimator uses the environment only to produce the scalar $R(\boldsymbol{\tau}_i)$; no derivative of the sampled actions, environment transitions, or reward function is required. The within-population reward standardization used by Agentic ESOpt is the practical finite-sample variant described in the main text.

Effect of omitting the $1/\sigma_t$ factor. At each iteration with $\sigma_t > 0$, the factor $1/\sigma_t$ is a positive scalar shared by all perturbation directions. For the same sampled perturbations and reward weights, omitting this factor only rescales the estimator and does not change its update direction. This also holds for the population-standardized rewards used in our implementation.

C.2 Proof of the Gaussian-Smoothing Bias Expansion

This subsection proves Lemma 1. We suppress the fixed external context c for readability and write $J(\theta)$ for the corresponding task objective.

Proof of Lemma 1. Define the Gaussian-smoothed objective

$$J_\sigma(\theta) = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}\left[J(\theta+\sigma\boldsymbol{\epsilon})\right].$$

By the Gaussian score-function identity, or equivalently integration by parts,

$$\nabla_\theta J_\sigma(\theta) = \frac{1}{\sigma}\mathbb{E}_{\boldsymbol{\epsilon}}\left[J(\theta+\sigma\boldsymbol{\epsilon})\boldsymbol{\epsilon}\right].$$

Thus, the population ES estimator is unbiased for $\nabla_\theta J_\sigma(\theta)$.
To compare the smoothed and original objectives, expand $J(\theta+\sigma\boldsymbol{\epsilon})$ around θ. Odd moments of a centered Gaussian vanish, and $\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top] = I$, giving

$$J_\sigma(\theta) = J(\theta) + \frac{\sigma^2}{2}\operatorname{Tr}\big(\nabla_\theta^2 J(\theta)\big) + O(\sigma^4) = J(\theta) + \frac{\sigma^2}{2}\Delta_\theta J(\theta) + O(\sigma^4).$$

Differentiating the expansion yields

$$\nabla_\theta J_\sigma(\theta) = \nabla_\theta J(\theta) + \frac{\sigma^2}{2}\nabla_\theta\Delta_\theta J(\theta) + O(\sigma^4).$$

Therefore, the leading smoothing bias relative to $\nabla_\theta J(\theta)$ is proportional to σ^2, as stated in Lemma 1. □

C.3 Long-Horizon Credit Assignment

We derive Eq. equation 8 for normalized feedback and the ES search direction, retaining the effects of finite-population normalization. For a random vector Z, define $\mathrm{Var}[Z] = \mathrm{Tr}(\mathrm{Cov}[Z])$.

Estimators and assumptions. Fix the checkpoint, external context, task batch, perturbation scale $\sigma > 0$, and ES population size $G \geq 2$, and initially consider a fixed finite horizon H. The reference PG signal is $\widehat{g}_{\mathrm{RL}} = \hat{R}(\boldsymbol{\tau})S_H$ before clipping and minibatch aggregation. For ES, let

$$Y_i = \hat{R}_i\boldsymbol{\epsilon}_i, \qquad \widehat{g}_{\mathrm{ES}} = \frac{1}{G}\sum_{i=1}^{G} Y_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(0, I_d).$$

The weights $\hat{R}_i$ are defined in Eq. equation 6. Assume exchangeable samples within each method's normalization group and finite required moments. Because the normalized weights sum to zero in every group, exchangeability gives $\mathbb{E}[\hat{R}(\boldsymbol{\tau})] = \mathbb{E}[\hat{R}_i] = 0$. Write $v_{\mathrm{RL}} = \mathrm{Var}[\hat{R}(\boldsymbol{\tau})] > 0$ and $v_{\mathrm{ES}} = \mathrm{Var}[\hat{R}_i] > 0$; they need not be equal. Here $\widehat{g}_{\mathrm{ES}}$ is the search direction before multiplication by α, distinct from the canonical gradient estimator in Appendix C.1.

Cross-turn score covariances vanish exactly. Let $u_t = \nabla_\theta \log \pi_\theta(a_t \mid h_t)$ and $S_H = \sum_{t=1}^{H} u_t$, where h_t is the full pre-action history. Assume a regular on-policy likelihood with locally fixed support, valid interchange of differentiation and summation, and finite score second moments. The likelihood must describe the action distribution actually sampled. Conditional normalization gives

$$\mathbb{E}[u_t \mid h_t] = \sum_a \pi_\theta(a \mid h_t)\nabla_\theta \log \pi_\theta(a \mid h_t) = \nabla_\theta \sum_a \pi_\theta(a \mid h_t) = 0.$$

For $s < t$, u_s is determined by h_t, so the tower property yields

$$\mathbb{E}[u_s u_t^\top] = \mathbb{E}\big[u_s \mathbb{E}[u_t^\top \mid h_t]\big] = 0, \qquad \mathrm{Var}[S_H] = \sum_{t=1}^{H}\mathrm{Var}[u_t].$$

This cancellation does not require independent actions or an additional decorrelation assumption. The approximation $\mathrm{Var}[S_H] \approx H\,\mathrm{Tr}(\Sigma_u)$ further requires $\mathrm{Cov}(u_t) \approx \Sigma_u$ to remain comparable across turns and horizons. For ES, $\mathbb{E}[\boldsymbol{\epsilon}_i] = 0$ and $\mathrm{Var}[\boldsymbol{\epsilon}_i] = d$.

Exact variance decomposition. Expanding the variance of the exchangeable population sum gives

$$\mathrm{Var}[\widehat{g}_{\mathrm{ES}}] = \frac{1}{G}\left(\mathrm{Var}[Y_1] + (G-1)\,\mathrm{Tr}\,\mathrm{Cov}(Y_1, Y_2)\right).$$

The weighted terms Y_i can be dependent even when perturbations are independent, because their weights share the population mean and standard deviation. For any zero-mean scalar X and zero-mean vector Z, direct expansion gives

$$\mathrm{Var}[XZ] = \mathrm{Var}[X]\,\mathrm{Var}[Z] + \mathrm{Cov}(X^2, \|Z\|_2^2) - \|\mathbb{E}[XZ]\|_2^2.$$

Applying this identity to $\hat{R}(\boldsymbol{\tau})S_H$ and Y_1 gives

$$\frac{\mathrm{Var}[\widehat{g}_{\mathrm{RL}}]}{v_{\mathrm{RL}}} = \mathrm{Var}[S_H] + \Delta_{\mathrm{RL}},$$
$$\frac{G\,\mathrm{Var}[\widehat{g}_{\mathrm{ES}}]}{v_{\mathrm{ES}}} = d + \Delta_{\mathrm{ES}},$$

where

$$\Delta_{\mathrm{RL}} = \frac{\mathrm{Cov}(\hat{R}(\boldsymbol{\tau})^2, \|S_H\|_2^2) - \|\mathbb{E}[\widehat{g}_{\mathrm{RL}}]\|_2^2}{v_{\mathrm{RL}}},$$
$$\Delta_{\mathrm{ES}} = \frac{\mathrm{Cov}(\hat{R}_1^2, \|\boldsymbol{\epsilon}_1\|_2^2) - \|\mathbb{E}[Y_1]\|_2^2 + (G-1)\,\mathrm{Tr}\,\mathrm{Cov}(Y_1, Y_2)}{v_{\mathrm{ES}}}.$$

Eq. equation 8 uses the approximation $|\Delta_{\mathrm{RL}}| \ll \mathrm{Var}[S_H]$ and $|\Delta_{\mathrm{ES}}| \ll d$. The displayed $1/G$ leading factor therefore retains an explicit finite-population correction; independent perturbations alone do not establish inverse-population variance scaling as G varies.

Dividing out feedback variance and population averaging. Under these approximations,

$$\frac{\mathrm{Var}[\widehat{g}_{\mathrm{RL}}]/v_{\mathrm{RL}}}{G\,\mathrm{Var}[\widehat{g}_{\mathrm{ES}}]/v_{\mathrm{ES}}} \approx \frac{\mathrm{Var}[S_H]}{d} \approx \frac{H\,\mathrm{Tr}(\Sigma_u)}{d}.$$

After these factors are removed, PG retains accumulated action-score variation, whereas ES retains one perturbation contribution. The exact difference is $\sum_{t=1}^{H}\mathrm{Var}[u_t] - d + \Delta_{\mathrm{RL}} - \Delta_{\mathrm{ES}}$. Thus, $\sum_{t=1}^{H}\mathrm{Var}[u_t] > d + |\Delta_{\mathrm{RL}} - \Delta_{\mathrm{ES}}|$ is sufficient for the adjusted PG variance to be larger. This is a comparison at the displayed normalizations and fixed estimator scaling, not an unconditional ordering of complete optimizer updates. Although $\mathrm{Var}[\boldsymbol{\epsilon}] = d$, normalized feedback and corrections can depend on both H and σ.

Variable horizons and experimental scope. For bounded random termination, reaching a turn is determined before its action is sampled. Zero-padding after termination therefore gives $\mathrm{Var}[S_H] = \mathbb{E}[\sum_{t=1}^{H}\|u_t\|_2^2]$. Section 4.1 compares adaptation across increasing minimum successful horizons H^*, and Figure 7 records realized horizons. Clipping, value-based advantages, learning-rate scaling, and RL minibatch aggregation belong to the implemented optimizers evaluated experimentally; the analysis isolates their underlying score-accumulation structure.

From realized horizon H to minimum successful horizon H^*. The theoretical quantity above is the realized trajectory length H. In the Sudoku experiments, we instead control

$$H^*(x) = \min_{\boldsymbol{\tau}: R(\boldsymbol{\tau})=1} |\boldsymbol{\tau}|,$$

which gives a task-dependent lower bound on the horizon of every successful trajectory. Because each valid Sudoku action fills at most one masked cell, masking 5, 10, and 15 masks yields $H^* = 5$, 10, and 15, respectively. Failed, invalid, or unproductive actions can make the realized horizon substantially larger than H^*.

The Sudoku task family scales the required decision sequence through the number of masked cells. A larger H^* entails more missing information and more correct cell-filling decisions before a successful terminal reward. This task-scale increase is shared by all methods, enabling direct comparison of their adaptation performance at each H^*. The experiments report the resulting performance crossover, while Figure 7 additionally records the realized trajectory length H during optimization.

C.4 AHD AS HEURISTIC-SPACE OPTIMIZATION

Automatic heuristic design is not a language-generation task in the usual sense: the generated text matters only through the algorithmic behavior it induces. We therefore view AHD as optimization over a heuristic space (Zheng et al., 2025b). Let $\mathcal{X}$ be a distribution over problem instances and let $h \in \mathcal{H}$ denote a heuristic sampled from the LLM-induced proposal distribution. A solver A_h uses h to construct a solution $y = A_h(x)$ for instance x. The idealized objective is

$$\max_{h \in \mathcal{H}} \mathbb{E}_{x \sim \mathcal{X}} \left[S(y, x) \right], \qquad y = A_h(x),$$

where S is a signed score chosen so that larger is better. This paper covers 5 NP-hard combinatorial optimization problems, including the Traveling Salesman Problem (TSP), which seeks the shortest tour visiting all nodes exactly once; the Capacitated Vehicle Routing Problem (CVRP), which minimizes the total routing cost for capacity-constrained vehicles serving customer demands; the 0-1 Knapsack Problem (KP), which maximizes the total value of selected items under a capacity constraint; the Bin-Packing Problem (BPP), which minimizes the number of fixed-capacity bins required to pack all items; and the Admissible Set Problem (ASP), which seeks the largest subset satisfying a predefined set of feasibility constraints.

For minimization problems such as TSP, CVRP, and BPP, S is the negative objective value; for maximization problems such as KP and ASP, S is the original objective value. The relevant object is therefore not the textual form of h, but the induced mapping from problem states to algorithmic decisions.

In constructive AHD, A_h is a sequential decision rule. At a partial solution state s_t, the heuristic induces a preference over feasible actions,

$$a_t \in \arg\max_{a \in \mathcal{A}(s_t)} h(s_t, a), \qquad s_{t+1} = T(s_t, a_t),$$

until a complete solution is obtained. TSP evaluates the length of the constructed tour, KP evaluates the value of the selected item set under a capacity constraint, and ASP evaluates the cardinality of the constructed admissible set. These tasks test whether Agentic ESOpt can alter the LLM proposal distribution toward heuristics whose local decisions accumulate into better global solutions.

In ACO-style AHD, A_h is stochastic. The heuristic defines an energy or desirability function over solution components, which biases a sampling distribution of the form

$$p_h(a_t \mid s_t) \propto \exp\left(\beta h(s_t, a_t)\right) \cdot \Phi_t(s_t, a_t),$$

where Φ_t denotes the non-learned search state, such as pheromone or feasibility terms, and β controls the strength of the heuristic bias. The objective is the expected quality of solutions sampled by this stochastic solver. Agentic ESOpt + EoH preserves this outer optimization problem and changes only the model-induced distribution over h.

C.5 Training FLOPs & Clocktime Calculation

We follow the FLOPs accounting of Gan & Isola (2026, Appendix E.1). For a model with P parameters and a trajectory containing $\bar{L}$ model-processed tokens, a forward pass requires approximately $2P\bar{L}$ FLOPs, while a backward pass requires approximately $4P\bar{L}$ FLOPs.

Let T, B, and G denote the number of training iterations, prompt batch size, and number of rollouts or perturbation directions per prompt, respectively. The method-specific training costs are

$$\begin{aligned}
\mathrm{FLOPs}_{\mathrm{ES}} &= T_{\mathrm{ES}} B_{\mathrm{ES}} G_{\mathrm{ES}} \underbrace{(2)}_{\substack{\text{policy forward}}} P\bar{L}_{\mathrm{ES}} \\
&= 2T_{\mathrm{ES}} B_{\mathrm{ES}} G_{\mathrm{ES}} P\bar{L}_{\mathrm{ES}}, \qquad (11) \\
\mathrm{FLOPs}_{\mathrm{GRPO}} &= T_{\mathrm{GRPO}} B_{\mathrm{GRPO}} G_{\mathrm{GRPO}} \underbrace{(2+2+4)}_{\substack{\text{policy forward} \\ +\text{ reference forward} \\ +\text{ policy backward}}} P\bar{L}_{\mathrm{GRPO}} \\
&= 8T_{\mathrm{GRPO}} B_{\mathrm{GRPO}} G_{\mathrm{GRPO}} P\bar{L}_{\mathrm{GRPO}}, \qquad (12) \\
\mathrm{FLOPs}_{\mathrm{PPO}} &= T_{\mathrm{PPO}} B_{\mathrm{PPO}} G_{\mathrm{PPO}} \underbrace{(2+2+2+4+4)}_{\substack{\text{policy forward+reference forward} \\ +\text{ critic forward+policy backward+critic backward}}} P\bar{L}_{\mathrm{PPO}} \\
&= 14T_{\mathrm{PPO}} B_{\mathrm{PPO}} G_{\mathrm{PPO}} P\bar{L}_{\mathrm{PPO}}. \qquad (13)
\end{aligned}$$

Thus, for trajectories of equal length, Agentic ESOpt costs one policy forward pass per sampled trajectory, **compared with approximately four forward-pass equivalents for Agentic GRPO and seven for Agentic PPO**. Environment execution is excluded throughout, since the comparison measures model-side training FLOPs.

Sudoku. In the reported Sudoku comparison, Agentic ESOpt and GRPO both use 100 update rounds with a prompt batch size of 32, while Agentic PPO does 500 update rounds. Agentic ESOpt evaluates $G_{\mathrm{ES}} = 32$ perturbation directions per prompt, whereas Agentic GRPO uses $G_{\mathrm{GRPO}} = 8$ rollouts. Therefore, assuming equal trajectory lengths,

$$\frac{\mathrm{FLOPs}_{\mathrm{ES}}}{\mathrm{FLOPs}_{\mathrm{GRPO}}} \approx \frac{2 \times 32}{8 \times 8} = 1.$$

Hence, the fourfold larger ES population is exactly compensated by its fourfold lower model-side cost per trajectory. Agentic PPO is capped at 500 training steps to remain within a comparable compute budget. For the measured values in Table 7, we further replace the nominal trajectory length by the actual number of model-processed tokens, yielding 3.1/6.3/9.4 EFLOPs for Agentic ESOpt and 3.2/7.6/10.9 EFLOPs for Agentic GRPO at $H^* \in \{5, 10, 15\}$. The higher measured GRPO FLOPs coincide with its longer realized trajectories; the diagnostics below are consistent with credit-assignment difficulty contributing to this difference but do not establish causality.

Table 7: Training compute and wall-clock time on agentic Sudoku. Both Agentic GRPO and Agentic ESOpt are executed on the same four NVIDIA H100 GPUs and fully utilize the available hardware.

	$H^* = 5$		$H^* = 10$		$H^* = 15$	
Method	FLOPs	Time	FLOPs	Time	FLOPs	Time
Qwen3.5-4B + Agentic GRPO	3.2 EFLOPs	5.4 h	7.6 EFLOPs	13.1 h	10.9 EFLOPs	19.0 h
Qwen3.5-4B + Agentic ESOpt (G=32)	3.1 EFLOPs	3.1 h	6.3 EFLOPs	5.8 h	9.4 EFLOPs	9.4 h

Compute and wall-clock efficiency. The memory and compute results characterize two complementary efficiency properties of Agentic ESOpt. The GPU-memory results in Table 1 measure its minimal training-side requirement: Agentic ESOpt requires only 8.41GB, equal to the inference memory of the Qwen3.5-4B backbone and 85.7% below GRPO's 58.88GB requirement.

ES requires a larger population ($G = 32$ versus 8-rollout in GRPO); however, it does not introduce a corresponding model-compute disadvantage. As analyzed in Appendix C.5, Agentic ESOpt with

$G = 32$ has a comparable model-FLOPs budget to eight-rollout GRPO. As shown in Table 7, in the actual four-H100 implementation, the measured wall-clock results indicate that the larger ES population does not translate into additional end-to-end overhead in this environment.

Math Reasoning and DocVQA. For Math Reasoning, Agentic ESOpt ($G = 16$) and Agentic GRPO ($G = 8$) (or Agentic GiGPO ($G = 8$), Agentic GRPO-LoRA ($G = 8$)) each cover the 400-example training set once. On DocVQA, all these methods cover at least 640 training examples. So Agentic ESOpt will not consume more FLOPs than Agentic GRPO.

For Turn PPO, we also keep its total FLOPs the same as Agentic GRPO.

C.6 Systems Advantages and Scope of Forward-Only Adaptation

Forward-only memory advantage. The main systems advantage of Agentic ESOpt is that parameter adaptation does not require a differentiable policy-training pass. Perturbation directions can be reconstructed from random seeds, and the update uses only scalar rewards without storing token-level gradients, backward activations, or persistent optimizer moments. For a context of length n, conventional gradient-based training additionally incurs sequence-dependent activation memory, approximately

$$M_{\mathrm{act}}(n) \propto b\, n\, d_{\mathrm{hid}}\, L_{\mathrm{saved}}, \tag{14}$$

whereas Agentic ESOpt introduces no corresponding backward-specific term. It still retains ordinary inference costs, including model weights, KV cache, inference workspaces, and parameter-update buffers.

Context length versus interaction horizon. This memory advantage should be distinguished from the horizon-dependent credit-assignment advantage analyzed in Appendix C.3. A large interaction horizon H does not necessarily imply a long model context n: for example, Sudoku exposes a compact local state at every turn. In contrast, ReAct-style Math and DocVQA append interaction history and tool outputs to the model context, making sequence-dependent training memory increasingly relevant.

Scope of the forward-only systems advantage. Beyond reducing model-state memory, forward-only adaptation removes several backward-specific systems requirements. Agentic ESOpt does not retain activations for backward computation, perform activation recomputation, maintain optimizer moments, or require backward-specific communication and scheduling. It also avoids a separate differentiable training backend for recomputing policy probabilities, eliminating the numerical-alignment issue between rollout and training engines. These advantages concern the additional cost of parameter adaptation rather than total inference cost: KV cache, inference workspaces, and model execution remain necessary. We also distinguish interaction horizon H from model context length n: Sudoku primarily stresses long-horizon credit assignment with short local contexts, whereas Math and DocVQA additionally accumulate long ReAct histories and therefore expose sequence-dependent training-memory costs.

Why forward-only adaptation is particularly suitable for agentic optimization. Beyond memory efficiency, Agentic ESOpt provides a highly end-to-end optimization interface. It treats the complete mapping from model parameters to environment outcome as a black box: tool execution, environment transitions, external memory or skills, and search procedures may all lie inside this mapping, while the optimizer requires only the resulting scalar feedback. In contrast, gradient-based Agentic RL must construct a differentiable policy-training objective through likelihood ratios, advantage estimation, and backward propagation. This distinction is particularly useful for on-the-fly and test-time adaptation, where trajectories and task evaluations are already produced by an external search loop and can be reused directly for parameter updates without introducing a separate differentiable training pipeline.

C.7 Details of GPU Memory Accounting

Task-specific memory accounting: Sudoku. For Sudoku, each turn contains only a compact board state and a short action, so the measured activation and transient-memory footprint is sufficiently small to not affect the reported values at our precision; the numbers therefore effectively reflect persistent training-side memory. Qwen3.5-4B occupies 8.41GB in BF16 inference. Agentic ESOpt introduces no additional persistent GPU state: perturbation seeds and rewards are stored on CPU, while noise is regenerated from seeds and applied, reverted, and accumulated into the update in place, avoiding full perturbation tensors, gradients, optimizer states, or a second model copy. It therefore remains at 8.41GB. In contrast, Agentic GRPO stores policy weights (8.41GB), reference weights (8.41GB), gradients (8.41GB), and two FP32 Adam moments (2×16.82GB), totaling 58.88GB. Agentic PPO additionally maintains a critic and its training states, reaching $58.88 + 50.46 = 109.34$ GB,GB. For Math and DocVQA, substantially longer accumulated ReAct contexts make sequence-dependent activation memory non-negligible, which we account for separately below.

Sequence-dependent memory in Math and DocVQA. Unlike Sudoku, Math reasoning and DocVQA accumulate reasoning traces, tool calls, and observations in the model context, introducing sequence-dependent backward memory for gradient-based training. To instantiate equation 14 with a concrete and reproducible quantity, we report the BF16 decoder-block checkpoint-input footprint, $M_{\text{ckpt}} = 2bnd_{\text{hid}}L$, for one training sequence ($b = 1$) at reference context lengths of 16K and 32K tokens. For Qwen3.5-4B ($d_{\text{hid}} = 2560$, $L = 32$), this gives 2.684/5.369 GB; for Qwen3-30B-A3B ($d_{\text{hid}} = 2048$, $L = 48$), it gives 3.221/6.442 GB. These values capture one explicit sequence-dependent component rather than the complete backward peak; within-block intermediate activations, runtime workspaces, and common inference-side KV cache are excluded.

Table 8: Training-side memory accounting in decimal GB at reference context lengths of 16K and 32K tokens. Checkpoint activation counts one BF16 decoder-block input per layer for one training sequence ($b = 1$). Accounted memory is the static state plus this explicit sequence-dependent component and should not be interpreted as the complete runtime peak. Common inference-side KV cache and runtime workspaces are excluded.

Backbone	Method	Static State	16K Ckpt. Act.	16K Accounted	32K Ckpt. Act.	32K Accounted
Qwen3.5-4B	Base Evaluation	8.41 GB	–	8.41 GB	–	8.41 GB
	Agentic ESOpt	8.41 GB	–	8.41 GB	–	8.41 GB
	Full-Parameter GRPO	58.88 GB	+2.684 GB	61.564 GB	+5.369 GB	64.249 GB
	Agentic PPO	109.34 GB	+2.684 GB	112.02 GB	+5.369 GB	114.709 GB
Qwen3-30B-A3B	Base Evaluation	61.064 GB	–	61.064 GB	–	61.064 GB
	Agentic ESOpt	61.064 GB	–	61.064 GB	–	61.064 GB
	GRPO + LoRA	61.492 GB	+3.221 GB	64.713 GB	+6.442 GB	67.934 GB
	Full-Parameter GRPO	427.450 GB	+3.221 GB	430.671 GB	+6.442 GB	433.892 GB

Why full-parameter GRPO is not evaluated at 30B. The 30B comparison uses the same four 80GB GPUs as Agentic ESOpt. Under the GPU-resident training-state configuration considered here, full-parameter GRPO requires approximately 427.45 GB before sequence-dependent activations, already exceeding the available device-memory budget; offloading-based implementations could trade GPU memory for additional host memory and communication, but are outside our matched setup. We therefore use GRPO+LoRA as the practical gradient-based baseline. Its 61.492 GB static footprint consists of 61.064 GB of shared frozen base-model weights, 0.107 GB of FP32 LoRA parameters, 0.107 GB of LoRA gradients, and 0.214 GB for two FP32 Adam moments. LoRA-GRPO still backpropagates through the backbone, introducing the sequence-dependent activation component above. In contrast, Agentic ESOpt stores perturbation metadata on CPU and applies seed-replayed perturbations and updates in place, introducing no additional persistent GPU training state or backward activation beyond the inference-resident model.

D EXPERIMENTAL SETTINGS AND ADDITIONAL RESULTS

This appendix reports implementation details and additional results in the order used in the main text: Sudoku, Math, DocVQA, WebArena-Lite, and automatic heuristic design. It ends with a consolidated Agentic ESOpt hyperparameter table.

D.1 SUDOKU

D.1.1 EMPIRICAL DIAGNOSTICS FOR LONG-HORIZON SCALING

We complement the long-horizon analysis in Section 4.1 with a qualitative visualization of local ES search, realized trajectory lengths during training, and advantage-sign diagnostics.

For each visualization, we evaluate a two-dimensional plane spanned by orthogonal parameter-space directions around the same Qwen3.5-4B checkpoint. Each heatmap value is the Gaussian-smoothed mean terminal reward on the Sudoku batch. The plots provide an intuitive low-dimensional view of what ES explores: nearby parameter perturbations produce different trajectory-level scores, and their relative scores guide the update. They are illustrative slices only and are not used to estimate reward density, sparsity, or the geometry of the full parameter space.

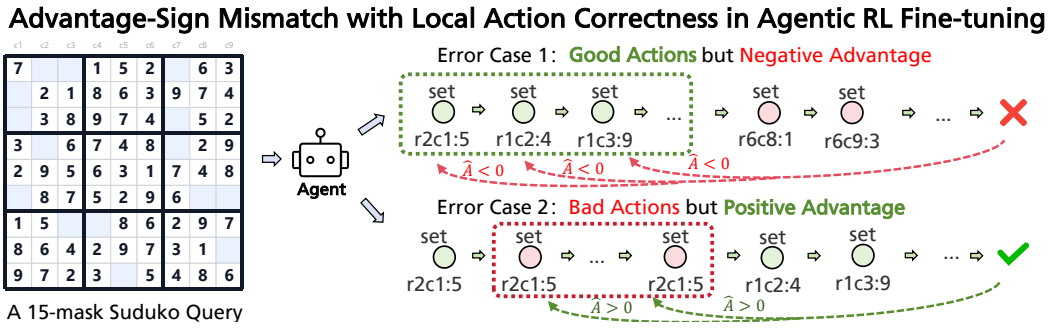


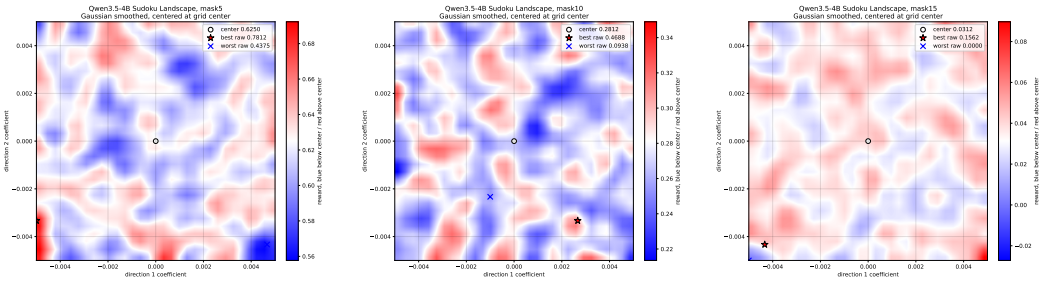
Figure 5: Two examples of advantage-sign misalignment with local action correctness in Agentic GRPO.

For an action sampled during training, let A denote the advantage assigned by the RL method. Table 9 reports two complementary sign-misalignment rates: the probability that an incorrect action receives positive advantage, $P(A > 0 \mid \text{incorrect})$, and the probability that a correct action receives negative advantage, $P(A < 0 \mid \text{correct})$. An action is considered correct when it successfully fills a currently empty cell with its unique target value. We compute descriptive conditional proportions over the logged training actions for each method and horizon; the table does not report a hypothesis test or confidence interval.

Table 9: Advantage-sign misalignment relative to local action correctness on Sudoku (%). We report the frequency of positive advantages on incorrect actions and negative advantages on correct actions.

Method	Conditional probability	$H^* = 5$	$H^* = 10$	$H^* = 15$
Agentic GRPO	$P(A > 0 \mid \text{incorrect action})$	1.60	2.60	4.50
Agentic GRPO	$P(A < 0 \mid \text{correct action})$	47.60	79.00	82.40
Agentic GiGPO	$P(A > 0 \mid \text{incorrect action})$	28.80	62.70	77.10
Agentic GiGPO	$P(A < 0 \mid \text{correct action})$	42.70	27.50	1.60

The two methods exhibit complementary failure modes. As H^* increases, GRPO rarely assigns positive advantage to incorrect actions, with $P(A > 0 \mid \text{incorrect})$ remaining below 5%, but increasingly assigns negative advantage to correct actions, rising from 47.60% to 82.40%. GiGPO substantially reduces this latter error, with $P(A < 0 \mid \text{correct})$ falling to 1.60% at $H^* = 15$, but at the cost of increasingly assigning positive advantage to incorrect actions, which rises from 28.80% to 77.10%. Thus, GiGPO mitigates one side of the sign-misalignment problem while exacerbating the other. These descriptive results are consistent with credit-assignment difficulty contributing to

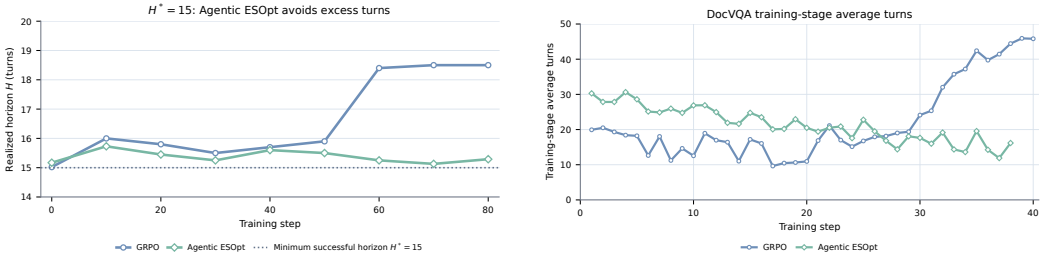


(a) Minimum successful horizon $H^* = 5$

(b) Minimum successful horizon $H^* = 10$

(c) Minimum successful horizon $H^* = 15$

Figure 6: Qualitative visualization of local ES search around the Qwen3.5-4B checkpoint for Sudoku tasks with $H^* \in \{5, 10, 15\}$. Each panel shows a two-dimensional parameter-space slice and marks the unperturbed checkpoint and the best and worst raw perturbations in that slice. Colors indicate Gaussian-smoothed mean terminal rewards. These slices illustrate the score variation available to parameter-space search; they are not estimates of reward density or the full optimization landscape.



(a) Sudoku ($H^* = 15$).

(b) DocVQA.

Figure 7: Realized trajectory length during training. (a) On Sudoku, Agentic GRPO trajectories lengthen while Agentic ESOpt remains near the 15-turn minimum. GRPO trains with temperature 1, top-$p = 1$, and top-$k = -1$; ESOpt uses temperature 0.7, top-$p = 0.8$, and top-$k = 20$. Both the curves of GRPO and ESOpt are evaluated under temperature 0.7, top-$p = 0.8$, and top-$k = 20$. (b) On DocVQA, both methods use the same rollout decoder and are aligned through update 40; GRPO again produces longer trajectories later in training.

the observed optimization behavior, while not establishing its causal importance relative to other factors.

Figure 6 provides intuition for an ES update: the optimizer samples nearby parameter directions, evaluates complete trajectories, and uses their relative scores to move the checkpoint. The figure serves only as a visual companion to the algorithm and does not support quantitative claims about reward density or how the full landscape changes with horizon.

Figure 7 shows the same qualitative separation on two tasks: GRPO trajectories lengthen during training, whereas Agentic ESOpt remains shorter and more stable. This is diagnostic rather than causal evidence, but it matches the credit-assignment analysis in Appendix C.3.

D.1.2 Detailed Ablation of the Sigma Schedule

As discussed in the main-text ablation, the cosine schedule separates two roles of the perturbation radius. A larger early σ supports broader exploration and stronger neighborhood smoothing, whereas a smaller late σ enables finer adaptation around the updated checkpoint. A fixed large radius can therefore obstruct late-stage refinement, while a fixed small radius can narrow the search too early. The repeated-run curves in Figure 8 show that the cosine schedule retains this advantage across independent training runs rather than relying on a single favorable seed. Each training run is evaluated three times. For the corresponding repeated-training entries in Table 1, we first compute the mean and standard deviation over the three evaluations within each run, and then report the average of

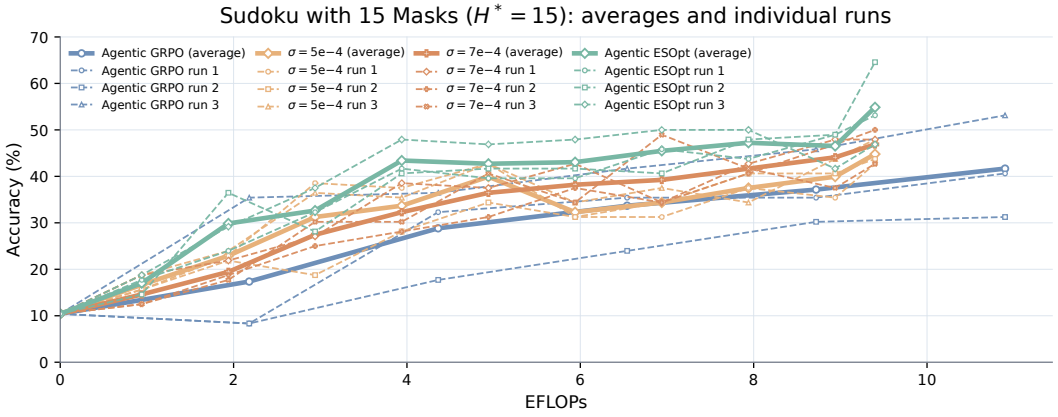


Figure 8: EFLOPs–performance curves across repeated runs.

the three run-level means together with the average of the three run-level standard deviations. Thus, the reported $\pm$ value reflects the average evaluation variability within training runs, rather than the standard deviation across independent training seeds.

Train-time adaptation additionally benefits from retaining a nonzero terminal radius. In Table 1, decaying to $\sigma_T = 0$ reduces held-out success from 89.58%, 62.50%, and 54.86% to 85.42%, 54.17%, and 28.13% for $H^* \in \{5, 10, 15\}$, respectively. One plausible explanation is overfitting: as the radius approaches zero, the update loses the local smoothing effect and may specialize more strongly to the finite training puzzles or sampled update directions. This interpretation is consistent with the larger degradation at $H^* = 15$, but we treat it as a hypothesis rather than a direct measurement of overfitting. Unlike test-time search, which optimizes the current task and can reasonably decay σ_T to zero, train-time adaptation must preserve held-out generalization; our results therefore support a nonzero σ_T in the training setting.

D.1.3 Training Configuration

Agentic ESOpt and Vanilla ES. All fixed ES comparisons use Qwen3.5-4B for 100 generations. At each generation, $G = 32$ full-parameter directions are evaluated on the same batch of 32 puzzles, and the resulting trajectory rewards are z-score normalized before an update with $\alpha = 5 \times 10^{-4}$. The interaction budget is three times the mask count—15, 30, and 45 turns for $H^* = 5$, 10, and 15—with at most 64 generated tokens per turn. Training and evaluation use temperature 0.7, top-$p = 0.8$, top-$k = 20$, min-$p = 0$, presence penalty 1.5, and repetition penalty 1.0. Evaluation is performed every 10 generations with three repeated runs.

Table 10: Sudoku ES configurations. All profiles use $G = 32$, 32 puzzles per generation, 100 generations, and $\alpha = 5 \times 10^{-4}$.

Profile	Mask count	Sigma schedule	Maximum turns
Vanilla ES	5 / 10	1.0×10^{-3} constant	15 / 30
Vanilla ES	15	7.0×10^{-4} constant	45
Agentic ESOpt	5 / 10	$1.0 \times 10^{-3} \rightarrow 2.5 \times 10^{-4}$ cosine	15 / 30
Agentic ESOpt	15	$7.0 \times 10^{-4} \rightarrow 5.0 \times 10^{-4}$ cosine	45

Agentic GRPO & GiGPO. Agentic GRPO and GiGPO run 100 rollout–update rounds with a global prompt batch of 32 and eight rollouts per prompt. Both use rollout micro-batch 8, policy-training micro-batch 2, actor learning rate 10^{-6}, KL-loss coefficient 10^{-3}, and clipping coefficient 0.2. GiGPO combines episode-level and step-group-relative advantages with discount factor $\gamma = 0.95$, and uses token-level importance-ratio clipping, an actor mini-batch size of 16, and one actor optimization epoch per rollout batch. GRPO-A samples with temperature 0.7, top-$p = 0.8$, and

top-$k = 20$, whereas GRPO-B and GiGPO use temperature 1, top-$p = 1$, and top-$k = -1$. One round consists of collecting a complete rollout batch and then optimizing the policy on that batch.

Agentic Turn-PPO. We implement Turn-PPO (Li et al., 2026) as the Agentic PPO baseline, using a separate Qwen3.5-4B critic backbone with a newly initialized value head. Turn-PPO uses the same global prompt batch of 32 but samples one rollout per prompt ($G = 1$), with an actor mini-batch size of 8. The actor and critic learning rates are 10^{-6} and 10^{-5}, respectively. Advantages are estimated using turn-level GAE with $\gamma = 0.99$ and $\lambda = 0.9$. The policy objective clips turn-level importance ratios, formed from the product of token-probability ratios within each turn, with clipping coefficient 0.2. Training uses temperature 1, top-$p = 1$, and top-$k = -1$, with a maximum of 500 training steps.

All four baselines use AdamW with betas $(0.9, 0.999)$, weight decay 0.01, gradient-clipping threshold 1.0, and entropy coefficient 0. Their learning rates remain constant without learning-rate warm-up. All RL baselines follow the same Sudoku interaction limits and are evaluated before training and every 20 rollout–update rounds, with three repeated evaluations using the Agentic ESOpt decoder: temperature 0.7, top-$p = 0.8$, and top-$k = 20$.

D.1.4 Vanilla-ES Population-Sensitivity Configuration

The 4B/9B comparison in Table 5 uses the $H^* = 15$ Sudoku environment for 100 generations and evaluates $G \in \{8, 16\}$ every 10 generations. To isolate population sensitivity, all runs use Vanilla ES with a constant perturbation scale $\sigma = 5 \times 10^{-4}$, update scale $\alpha = 5 \times 10^{-4}$, full-parameter perturbations, population z-score normalization, and eight training puzzles per direction. Each generation therefore contains $8G$ direction–puzzle rollouts. Perturbation rollouts and evaluation use temperature 0.7, top-$p = 0.8$, and top-$k = 20$, and each evaluation is repeated three times.

The environment provides a binary terminal reward: a rollout receives 1 only when the final board is a complete legal Sudoku solution preserving all given cells. Invalid formats, out-of-range values, attempts to modify given cells, and overwrites leave the board unchanged. This fixed setup makes population size the only varied ES hyperparameter in the ablation; the cosine perturbation schedule and intermediate reward shaping are not used.

D.2 Repeated-run stability

We repeat Agentic GRPO, Vanilla ES, and Agentic ESOpt with three independent training seeds at $H^* = 15$. Each final checkpoint is evaluated three times, and we first average these evaluations within each training run. We then report the mean and sample standard deviation across the three independent training runs. Table 11 reports all three independent runs, with the last column showing cross-run variability.

Table 11: Repeated runs on Sudoku ($H^* = 15$). Each run is mean$\pm$std over three evaluations; "Avg." reports mean$\pm$std across the three run means (ddof$= 0$).

Method	Run 1	Run 2	Run 3	Avg.
GRPO	40.63 ± 1.47	31.25 ± 2.95	53.13 ± 2.55	41.67 ± 8.96
Vanilla ES ($5e{-}4$)	42.71 ± 2.55	43.75 ± 2.55	47.92 ± 1.47	44.79 ± 2.25
Vanilla ES ($7e{-}4$)	47.92 ± 2.55	50.00 ± 3.90	42.71 ± 2.55	46.88 ± 3.07
Agentic ESOpt	53.13 ± 2.55	64.58 ± 2.55	46.88 ± 1.47	$\mathbf{54.86 \pm 7.33}$

D.3 Math Reasoning

All method-specific configurations described below refer to training unless otherwise stated. Final evaluation is conducted under a common protocol shared by Agentic ESOpt, Agentic GRPO, Trace2Skill, their combined variants, and all other reported baselines. For Trace2Skill, we sample 16 runs for each instance and follow the instructions from Ni et al. (2026) in using at most one error trajectory per instance to distill skill.

The configuration tables below document the matched Qwen3.5-4B comparison. The Qwen3-30B-A3B experiment in Table 2 uses the same task, reward, and final-evaluation interface, but scales full-parameter ES optimization to a sparse MoE with 30.5B total parameters and 3.3B activated per token (Yang et al., 2025).

D.3.1 TRAINING CONFIGURATION

Experiments of ReAct-style Math Reasoning are conducted on four A100 80GB GPUs. We first optimize the No Skill Qwen3.5-4B policy and record the seed of every parameter update together with the trajectories later supplied to Trace2Skill. For final evaluation, the base checkpoint is reloaded, and the update sequence is deterministically reconstructed. The No Skill and skill-conditioned evaluations therefore share an identical parameter stage and differ only in the external skill context.

Agentic GRPO. The Agentic GRPO baseline uses the same sampling configuration as Agentic ESOpt, with lr 10^{-6}, eight rollouts per prompt, KL coefficient 10^{-3} using the low-variance KL form, temperature 1, top-$p = 1$, top-$k = 40$, presence penalty 2, and repetition penalty 1.

Table 12: Training configurations for Math reasoning. Agentic ESOpt adopts the totally same parameter to Qwen-3.5-4B on Qwen-30B-A3B.

Agentic ESOpt		Agentic GRPO/GiGPO	
Parameter	Value	Parameter	Value
Backbone	Qwen3.5-4B	Backbone	Qwen3.5-4B
ES generations	25	Training records	400
Population size G	16	Prompt batch size	16
Cases per direction	16	Rollouts per prompt	8
Update scale α	5×10^{-4}	Training epochs	1
σ schedule	$10^{-3} \to 5 \times 10^{-4}$ cosine	Training rounds	25
Reward normalization	Population z-score	Total trajectories	3,200
Maximum turns	50	Learning rate	10^{-6}
Generation limit	4096 tokens / turn	KL coefficient	10^{-3}
Total generation cap	32,768 tokens / trajectory	Maximum interaction	50 user + 50 assistant turns
–	–	Generation limit	4096 tokens / assistant turn
–	–	Total generation cap	32,768 tokens / trajectory

Each direction is evaluated on 16 problems, giving $16 \times 16 = 256$ direction–problem trajectories per generation. Training uses temperature 1, top-$p = 1$, top-$k = 40$, min-$p = 0$, presence penalty 2, and repetition penalty 1. The trajectory reward is exact match on the parsed final answer. Intermediate evaluation is performed every 10 generations with one sample per problem.

Agentic GiGPO. The Qwen3.5-4B GiGPO baseline uses the same 400 training problems, global prompt batch of 16, eight rollouts per prompt, and 25 rollout–update rounds as Agentic GRPO, yielding 3,200 training trajectories. It also follows the same training decoder, task reward, and interaction limits. GiGPO combines episode-level and step-group-relative advantages with discount factor $\gamma = 0.95$, and applies token-level importance-ratio clipping with coefficient 0.2. The actor learning rate is 10^{-6}, the actor mini-batch size is 16, and each rollout batch is used for one actor optimization epoch. The explicit KL-loss coefficient is 10^{-3}.

Agentic Turn-PPO. We additionally implement Turn-PPO (Li et al., 2026) using a Qwen3.5-4B actor and a separate Qwen3.5-4B critic backbone with a newly initialized value head. Turn-PPO uses the same Math training set, with a global prompt batch of 16, one rollout per prompt ($G = 1$), and an actor mini-batch size of 8. The actor and critic learning rates are 10^{-6} and 10^{-5}, respectively. Advantages are computed using turn-level GAE with $\gamma = 0.99$ and $\lambda = 0.9$. The policy objective clips turn-level importance ratios, formed from the product of token-probability ratios within each turn, with clipping coefficient 0.2.. Final evaluation follows the common protocol below. As shown in Figure 3 (b), Turn-PPO collapses in the middle of training (finally to 0) so we exclude them from Table 2.

Agentic GRPO+LoRA. For Qwen3-30B-A3B, full-parameter RL exceeds the four-GPU memory budget under the training-state configuration considered here, so we use GRPO+LoRA with rank 32. GRPO+LoRA retains the corresponding GRPO rollout count, optimizer hyperparameters, task reward, decoder, and interaction budget while updating only the LoRA parameters. This GRPO-to-GRPO+LoRA matching rule is used for both Math and DocVQA.

D.3.2 COMMON EVALUATION CONFIGURATION

All reported methods and baselines use the same final evaluation protocol. For Math reasoning, we evaluate on 100 held-out DAPO problems and 30 AIME 2026 problems, with four samples per

problem. Each trajectory is allowed at most 50 interaction turns and up to 4096 generated tokens per turn. We use temperature 1, top-$p = 1$, top-$k = 40$, min-$p = 0$, presence penalty 2, and repetition penalty 1 for all methods. These evaluation settings are shared by Agentic ESOpt, Agentic GRPO, Trace2Skill, their combined variants, and all other reported baselines, independent of their method-specific training configurations.

D.4 DocVQA

As in Math reasoning, the configurations below specify the corresponding training procedures unless otherwise stated. For DocVQA, Agentic ESOpt and Agentic GRPO additionally use fully aligned interaction and generation settings during training, and all reported methods share the same final evaluation configuration. For Trace2Skill, we sample 16 runs for each instance and follow the instructions from Ni et al. (2026) in using at most one error trajectory and one correct trajectory per instance to distill skill.

D.4.1 Training Configuration

Experiments of ReAct-style DocVQA are also conducted on four A100 80GB GPUs. The DocVQA experiment follows the same paired reconstruction protocol as Math: the No Skill run records update seeds and source trajectories, and final evaluation replays the identical parameter sequence before adding the post-hoc skill context.

Agentic GRPO DocVQA Agentic GRPO uses the same sampling configuration and interaction budget as Agentic ESOpt, together with the same optimizer and KL settings as the Math baseline.

Table 13: Training configurations for DocVQA.

Agentic ESOpt		Agentic GRPO	
Parameter	Value	Parameter	Value
Backbone	Qwen3.5-4B	Backbone	Qwen3.5-4B
ES generations	40	Training records	50
Population size G	16	Prompt batch size	4
Cases per direction	16	Rollouts per prompt	8
Update scale α	5×10^{-4}	Training epochs	15
σ schedule	$10^{-3} \rightarrow 5 \times 10^{-4}$ cosine	Training rounds	180
Reward normalization	Population z-score	Total trajectories	5,760
Maximum turns	50	Learning rate	10^{-6}
Generation limit	512 tokens / turn	KL coefficient	10^{-3}
Total generation cap	32,768 tokens / trajectory	Maximum turns	50
–	–	Generation limit	512 tokens / assistant turn
–	–	Total generation cap	32,768 tokens / trajectory

Each generation contains $16 \times 16 = 256$ direction–question trajectories. Training uses temperature 1, top-$p = 1$, top-$k = 40$, min-$p = 0$, presence penalty 2, and repetition penalty 1. A valid trajectory must contain at least one parsed tool action before the final answer. Continuous ANLS is used as the training reward; threshold accuracy counts an answer as correct when ANLS is strictly greater than 0.5. Intermediate evaluation is performed every 10 generations with one sample per question.

Agentic GRPO and Agentic ESOpt therefore use the same 50-turn interaction limit, 512-token per-turn generation limit, total trajectory generation cap, and sampling configuration during training. The final two records do not fill a four-question batch and are omitted, leaving 12 training rounds per epoch. Continuous ANLS is used as the training reward as well. Agentic ESOpt adopts the totally same parameter to Qwen-3.5-4B on Qwen-30B-A3B.

Agentic GiGPO and Turn-PPO. We additionally evaluate Qwen3.5-4B GiGPO and Turn-PPO on the same 50-question DocVQA training set using ANLS as the task reward. Both use a global prompt batch of 16, with eight rollouts per prompt for GiGPO and one rollout per prompt for Turn-PPO. Their advantage estimators, importance-ratio granularity, learning rates, clipping coefficients, KL-loss settings, and shared optimizer settings follow those described in Appendix D.3. GiGPO uses an actor mini-batch size of 16 and one actor optimization epoch per rollout batch. Turn-PPO uses an actor mini-batch size of 8 and two actor optimization epochs per rollout batch, together with a separate Qwen3.5-4B critic and turn-level GAE. Both methods follow the common final evaluation protocol below. The four-question prompt batch, 180 training rounds, and 5,760 trajectories in Table 13 describe Agentic GRPO specifically. Similar to Figure 3 (b), Turn-PPO collapses in the middle of training so we exclude them from Table 2.

D.4.2 COMMON EVALUATION CONFIGURATION

All reported methods and baselines also follow the same final evaluation protocol on DocVQA. We evaluate on the same held-out set of 100 questions with four samples per question. Each trajectory is limited to 50 interaction turns, 512 generated tokens per turn, and 32,768 generated tokens in total. Sampling uses temperature 1, top-$p = 1$, top-$k = 40$, min-$p = 0$, presence penalty 2, and repetition penalty 1 for every method. Agentic ESOpt, Agentic GRPO, Trace2Skill, their combined variants, and all other reported baselines therefore use identical evaluation settings. In addition, the interaction and generation budgets of Agentic ESOpt and Agentic GRPO are fully aligned during DocVQA training.

D.4.3 SKILL-CONDITIONED MATH AND DOCVQA RESULTS

For completeness, Table 14 reports the experiments that combine train-time parameter optimization with the post-hoc Trace2Skill context. The parameter updates are learned under No Skill prompts; the collected training trajectories are subsequently used for skill distillation, with no additional skill-conditioned parameter-update stage.

Table 14: Math-reasoning and DocVQA results with Trace2Skill contexts. DAPO, AIME 2026, and DocVQA accuracy are reported in %. Mean@4 averages four sampled Pass@1 scores; Pass@4 or Max@4 represents the best of four samples.

		Math reasoning				DocVQA			
		DAPO		AIME 2026		ANLS		Accuracy	
Model	Method	Mean@4	Pass@4	Mean@4	Pass@4	Mean@4	Max@4	Mean@4	Pass@4
Qwen3.5-4B	Trace2Skill	64.8	82.0	50.8	83.3	0.4612	**0.6772**	47.3	**69.0**
Qwen3.5-4B	Agentic GRPO + Trace2Skill	67.8	85.0	50.0	80.0	0.4743	0.5692	49.5	60.0
Qwen3.5-4B	Agentic ESOpt + Trace2Skill	**77.3**	**86.0**	**71.7**	**96.7**	**0.5086**	0.6654	**52.8**	68.0
	Δ vs Trace2Skill	↑12.5	↑4.0	↑20.8	↑13.3	↑0.0474	↓0.0118	↑5.5	↓1.0

Agentic ESOpt composes effectively with Trace2Skill: the combined method achieves the strongest Qwen3.5-4B Mean@4 results on DAPO, AIME 2026, and DocVQA. It also outperforms Agentic GRPO + Trace2Skill on all eight reported metrics.

D.4.4 REPEATED-SAMPLING PROFILES FOR MATH AND DOCVQA

Figure 9 extends the four-sample results in Table 2 to repeated-sampling budgets up to $k = 32$, using the same evaluation decoder for each method.

Across all four panels, Agentic ESOpt remains above the matched Agentic GRPO baseline as the sampling budget increases. On AIME 2026 and both DocVQA metrics, it also preserves an advantage over the base policy at larger k, indicating that the average-performance gains are not obtained by collapsing repeated-sampling coverage.

DocVQA training-stage trajectory lengths are shown together with the Sudoku diagnostic in Figure 7.

D.5 WEBARENA-LITE

D.5.1 AGENTIC REASONING FOR WEBARENA: MODEL-SCALABILITY RESULTS

We evaluate Agentic ESOpt on WebArena-Lite, a 165-task browser benchmark derived from WebArena (Liu et al., 2025b). Each task provides a natural-language goal and an interactive website state. The agent observes a WebRL-style textual browser representation, acts in an id-based action space, and receives task-success feedback after completing a trajectory (Zhou et al., 2024; Qi et al., 2025). The evaluation set contains 21 Reddit, 32 GitLab, 35 CMS, 28 Map, 46 OSS, and 3 Wikipedia tasks; the five major categories are shown separately in Table 15, while Wikipedia is included only in the dataset average.

This setting scales Agentic ESOpt to Qwen3.5-27B on four NVIDIA H100 80GB GPUs. We compare parameter-only adaptation and its composition with Trace2Skill or SkillOpt. The Trace2Skill variant distills a skill after the shared No Skill ES run; the SkillOpt variant performs joint skill and parameter updates. GPT-5.4, GPT-5.4-mini, and GPT-5.4-nano are evaluation-only references.

All Qwen3.5-27B rollouts use temperature 0.7, a 2048-token generation budget, WebRL-style observations, and WebRL id-based actions. Agentic ESOpt uses population size $G = 8$ and task success

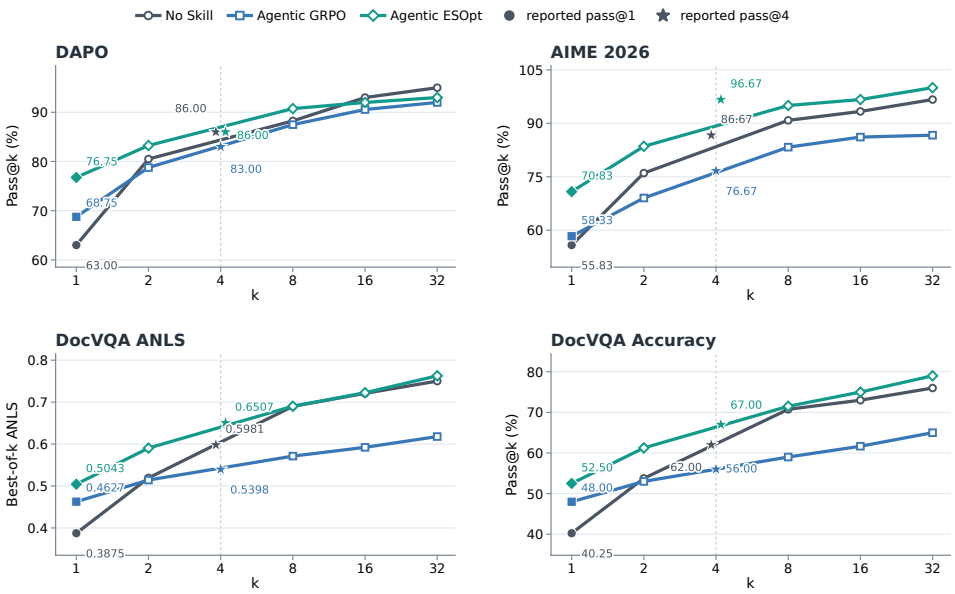


Figure 9: Repeated-sampling profiles for DAPO, AIME 2026, DocVQA ANLS, and DocVQA accuracy. The $k = 1$ markers use the reported Mean@4 values as estimates of single-sample performance, while stars show the originally reported Pass@4 or Max@4 values. Curves report the newly estimated best-of-k results for $k \in \{2, 8, 16, 32\}$; the newly computed $k = 4$ curve points are omitted to keep the original four-sample measurements visually distinct.

on sampled training cases as the reward. Trace2Skill also samples each question eight times. Final results are evaluated on the official 165-task WebArena-Lite set (Liu et al., 2025b).

Table 15: WebArena-Lite success rates (%). We report means over three runs and standard deviations. Improved Agentic ESOpt cells relative to their paired baseline are shaded green; bold denotes the best result in each column. The five displayed site categories contain 162 tasks; the remaining Wikipedia tasks are included in Dataset Avg.

Model	Method	Reddit (21)	GitLab (32)	CMS (35)	Map (28)	OSS (46)	Dataset Avg.
Strong frozen baseline (evaluation only)							
GPT-5.4	No Skill	47.62	46.88	46.67	**19.05**	21.01	34.14 ± 0.76
GPT-5.4-mini	No Skill	39.68	29.17	30.48	13.10	13.77	23.23 ± 1.14
GPT-5.4-nano	No Skill	39.68	27.08	19.05	11.90	8.70	18.79 ± 0.99
Qwen3.5-27B	No Skill	50.79	35.42	41.90	8.33	21.01	29.49 ± 1.14
Qwen3.5-27B	Agentic ESOpt + No Skill	49.21	43.75	49.52	14.29	30.43	36.16 ± 0.57
	Δ vs No Skill	↓1.58	↑8.33	↑7.62	↑5.96	↑9.42	↑6.67
Qwen3.5-27B	Trace2Skill	49.21	39.58	46.67	13.10	28.26	33.94 ± 2.76
Qwen3.5-27B	Agentic ESOpt + Trace2Skill	52.38	41.67	50.48	10.71	**32.61**	**36.36 ± 0.86**
	Δ vs Trace2Skill	↑3.17	↑2.09	↑3.81	↓2.39	↑4.35	↑2.42
Qwen3.5-27B	SkillOpt	46.00	40.60	38.10	8.30	20.30	28.89 ± 0.57
Qwen3.5-27B	Agentic ESOpt + SkillOpt	**58.70**	**47.90**	**50.50**	4.80	23.20	34.75 ± 0.76
	Δ vs SkillOpt	↑12.70	↑7.30	↑12.40	↓3.50	↑2.90	↑5.90

Agentic ESOpt improves the Qwen3.5-27B No Skill baseline from 29.49% to 36.16%, a gain of 6.67 percentage points. Combined with Trace2Skill, it improves the Trace2Skill baseline from 33.94% to 36.36%, demonstrating that parameter-space optimization can compose with skill-space optimization on larger LLMs.

D.5.2 Goal-Conditioned Task Formulation

WebArena-Lite evaluates goal-conditioned control in a partially observed browser environment. A task g specifies a user goal and an initial latent web state. At time t, the agent receives observation

o_t, samples an action

$$a_t \sim \pi_\theta(\cdot \mid \boldsymbol{o}_{\le t}, \boldsymbol{a}_{<t}, g, c),$$

and transitions to the next browser state. A trajectory $\boldsymbol{\tau} = (o_0, a_0, \ldots, o_T)$ is successful when it satisfies the goal predicate within the interaction budget. The benchmark objective is

$$J_{\text{web}}(\theta; c) = \mathbb{E}_{g\sim\mathcal{G}}\mathbb{E}_{\boldsymbol{\tau}\sim\pi_\theta(\cdot|g,c)}\left[\mathbf{1}_{\text{succ}}(\boldsymbol{\tau}, g)\right],$$

estimated by the empirical success rate over the held-out task set. The reward is sparse: intermediate actions receive no direct task credit, and the final score depends on whether the complete trajectory reaches the target web state.

The external context c represents non-parametric adaptation. In the No Skill condition, it contains the task instruction and interaction history; in the Trace2Skill condition, it additionally contains procedural knowledge distilled from previous trajectories. The No Skill pair in Table 15 isolates the parameter update, while the skill-conditioned pair compares the complete adaptation pipelines.

For website category $\mathcal{G}_k$, the reported conditional success rate is

$$\widehat{J}_{\mathcal{G}_k}(\theta; c) = \frac{1}{|\mathcal{G}_k|}\sum_{g\in\mathcal{G}_k}\mathbf{1}_{\text{succ}}(\boldsymbol{\tau}_g, g).$$

The dataset average applies the same estimator to the full set $\mathcal{G}$.

D.5.3 Rollout Protocol and Data Split

All frozen and Agentic ESOpt rows use the same browser protocol. The agent observes WebRL-style textual browser states, acts in the WebRL id-based action space, and is limited to 30 browser actions per task. Each response has a 2,048-token budget, the browser viewport is 1280×720, and the evaluator assigns binary success after termination. Sampling uses temperature 0.7, top-$p = 0.8$, top-$k = 20$, min-$p = 0$, presence penalty 1.5, and repetition penalty 1. Final results average three complete evaluations over all 165 held-out tasks.

The training split is constructed from the original 812 WebArena tasks after excluding the 165 tasks mapped to WebArena-Lite by the benchmark-provided correspondence. Site-stratified splitting of the remaining 647 tasks yields 582 training tasks and 65 validation tasks. We run each model with three seeds to distill the skill (base model, Agentic RL model, Agentic ESOpt model) and select the best one on the validation dataset as the evaluation skill. WebArena-Lite tasks never contribute parameter-update rewards or trajectory-to-skill inputs, and periodic evaluation is read-only. All six sites—Reddit, GitLab, Wikipedia, Map, Shopping, and Shopping Admin—remain enabled.

D.5.4 Parameter and Skill Adaptation Stages

Table 16: WebArena-Lite adaptation stages on Qwen3.5-27B. Trace2Skill is post-hoc; SkillOpt and Agentic ESOpt are updated jointly.

Condition	Parameter stage	Skill stage	Data per stage	Additional ES
No Skill baseline	none	none	–	none
Agentic ESOpt + No Skill	70 ES generations	none	$G = 8$ directions $\times$ 8 tasks/generation	–
Trace2Skill baseline	none	70 skill iterations	8 tasks $\times$ 8 rollouts/iteration	none
Agentic ESOpt + Trace2Skill	same 70-generation ES run	one post-hoc distillation	all completed ES trajectories	none
SkillOpt baseline	none	at most 10 SkillOpt steps	8 training tasks/step	none
Agentic ESOpt + SkillOpt	one ES update per SkillOpt step	at most 10 SkillOpt steps	8 training tasks/step	retain every ES update

The Trace2Skill baseline starts from an empty skill and runs 70 skill-evolution iterations. Each iteration uses eight training tasks with eight sampled trajectories per task and selects at most one positive and one negative trajectory per task. Validation is performed every 10 iterations, and Table 15 reports the final evaluated skill.

SkillOpt runs for at most 10 steps with eight training tasks per step and eight sampled trajectories per task. Each proposed skill is checked on the fixed 65-task validation split described above. If validation rejects the proposed skill, the previous skill is kept; in the joint Agentic ESOpt + SkillOpt condition, the ES parameter update from that step is still retained. Thus, validation gates only the skill update, not the parameter update.

The No Skill parameter stage performs 70 full-parameter ES updates with $G = 8$ on eight training tasks per generation. Each generation therefore contains $8 \times 8 = 64$ direction–task evaluations, with

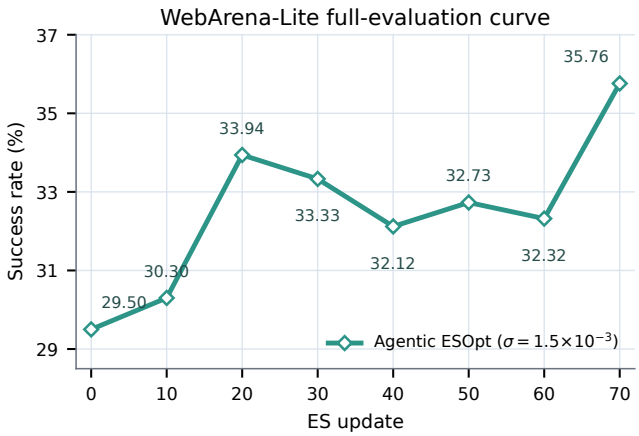


Figure 10: Full WebArena-Lite evaluation success rate during the No Skill Qwen3.5-27B Agentic ESOpt run. Evaluation is performed every 10 ES updates; the curve starts from the 29.50% base checkpoint and reaches 35.76% after 70 updates.

every perturbation scored on the same task batch. Rewards are z-score normalized, $\sigma_0 = 1.5\times10^{-3}$, $\sigma_t = 1.5 \times 10^{-3}$, $\alpha = 2.5 \times 10^{-4}$, and evaluation is performed every 10 generations.

After the parameter stage, trajectories from all completed ES generations are passed once to Trace2Skill. Final evaluation reconstructs the same No Skill update sequence and adds the distilled skill to the system prompt. This experiment implements sequential shared-trajectory composition; no additional skill-conditioned ES stage is performed.

D.5.5 Full-Evaluation Curve

The full-set curve is non-monotonic across intermediate checkpoints, but the final update reaches the best observed success rate. Table 15 reports the three-run final evaluation used for the main comparison.

D.6 Automatic Heuristic Design

D.6.1 Experimental Configuration

Experiments of AHD are done on a cluster of eight 3090 24GB GPUs. We evaluate Agentic ESOpt under two fixed outer search scaffolds: independent Sample and EoH. In each paired comparison, the proposal budget and outer search procedure are unchanged; Agentic ESOpt adds only the parameter-space update. All runs use LLaMA-3.1-8B-Instruct.

EoH maintains $N = 10$ heuristics for 25 outer generations. Operators e1 and e2 each generate N candidates, while m1 and m2 each generate kN candidates, giving a total budget of $25N(2 + 2k)$. We use $k = 1$ for $T = 1{,}000$ and $k = 3$ for $T = 2{,}000$. The Sample scaffold evaluates independent i1 proposals in batches of 20.

Table 17: AHD proposal budgets and Agentic ESOpt update units. Matched baselines use the same proposal counts. In EoH, m1 and m2 each form one ES update batch per outer generation.

Outer scaffold	T	Outer generations	Candidates/generation	ES update batches	Directions/batch	$\sigma_0 \to \sigma_T$
EoH ($k = 1$)	1,000	25	$10(2 + 2k) = 40$	50	10	$10^{-3} \to 0$
EoH ($k = 3$)	2,000	25	$10(2 + 2k) = 80$	50	30	$10^{-3} \to 0$
Independent Sample	1,000	50	20	50	20	$10^{-3} \to 0$
Independent Sample	2,000	100	20	100	20	$10^{-3} \to 0$

Agentic ESOpt uses full-parameter, one-sided Gaussian perturbations, cosine decay $\sigma : 10^{-3} \to 0$, no warmup, $\alpha = 5 \times 10^{-4}$, and population z-score normalization. In EoH, only mutation operators m1 and m2 trigger parameter updates, yielding two update batches per outer generation; e1 and e2 remain unchanged. In Sample, each batch of 20 proposals forms one update.

All objectives are converted internally to minimization costs. EoH uses the selected parent's cost minus the perturbed child's cost, while Sample uses the negative child score. Invalid or non-finite programs receive a finite within-batch penalty before normalization, and an all-invalid batch produces no update. Candidate generation uses temperature 1, top-$p = 0.98$, no top-k cutoff, and at most 768 new tokens.

D.6.2 Additional Results and Ablations

Table 18: Constructive-heuristic results at total evaluation $T \in \{1000, 2000\}$. TSP is minimized; KP and ASP are maximized. The Δ rows use the normalized gap-ratio gain defined in Section 4.3.

Method	TSP $N = 20$	TSP $N = 50$	KP $N = 50, W = 12.5$	KP $N = 100, W = 25$	ASP $N = 15, W = 10$	ASP $N = 21, W = 15$
Optimal	3.8199	5.6750	20.0370	40.2710	3,003	43,596
Greedy Construct	4.4797	6.9590	19.9850	40.2250	1,530	15,050
Total Evaluations: $T = 1000$						
Sample	4.3286	6.7110	19.9896	40.2297	2,753	30,336.67
Agentic ESOpt + Sample	4.2336	6.5488	19.9899	40.2314	2,729	31,082.67
Δ vs Sample	↓22.96%	↓18.56%	↑0.79%	↑4.42%	↓8.76%	↑5.96%
EoH	4.2481	6.5450	19.9958	40.2320	2,760	28,465.67
Agentic ESOpt + EoH	4.2170	6.4631	20.0007	40.2351	2,770	30,887.33
Δ vs EoH	↓7.83%	↓10.39%	↑13.59%	↑8.65%	↑4.29%	↑19.06%
Total Evaluations: $T = 2000$						
Sample	4.2585	6.6008	19.99118	40.2320	2,759	30,123.33
Agentic ESOpt + Sample	4.2098	6.5332	19.99120	40.2320	2,751	30,269.33
Δ vs Sample	↓12.51%	↓7.88%	↑0.04%	0.00%	↓3.17%	↑1.10%
EoH	4.2165	6.4706	19.9984	40.2356	2,764	28,016.00
Agentic ESOpt + EoH	4.1799	6.4442	19.9987	40.2358	2,784	30,124.67
Δ vs EoH	↓10.19%	↓3.43%	↑0.81%	↑0.48%	↑9.13%	↑15.65%

ACO-style results. At $T = 1{,}000$, Agentic ESOpt + EoH improves three of the six ACO-style test sets: both TSP settings and CVRP-50. At $T = 2{,}000$, it improves both CVRP and both BPP settings; the two TSP differences are small (0.08% and 0.36%). Together with the constructive Sample and EoH comparisons, Agentic ESOpt improves 28 of 36 matched method–budget settings across 12 test sets and six scenarios, with one tie and seven regressions.

Table 19: AHD ACO-style results at total evaluation budgets $T \in \{1000, 2000\}$. Lower is better for all columns. The Δ rows report the ratio $\mathrm{EoH}/(\mathrm{Agentic\ ESOpt} + \mathrm{EoH}) - 1$; positive ratios denote improvement. Improvements are shaded green with green deltas; regressions remain unshaded with neutral-gray deltas.

Method	TSP $N = 50$	TSP $N = 100$	CVRP $N = 50, C = 50$	CVRP $N = 100, C = 50$	BPP $N = 500, C = 150$	BPP $N = 1{,}000, C = 150$
ACO	5.992	8.948	11.355	18.778	208.828	417.938
Total Evaluations: $T = 1000$						
EoH	6.030	8.859	9.328	15.666	203.162	405.172
Agentic ESOpt + EoH	5.882	8.391	9.265	16.073	203.500	405.787
Δ vs EoH	↓2.53%	↓5.57%	↓0.68%	↑2.53%	↑0.17%	↑0.15%
Total Evaluations: $T = 2000$						
EoH	5.886	8.368	9.379	15.989	203.307	405.688
Agentic ESOpt + EoH	5.890	8.398	9.179	15.420	203.052	405.276
Δ vs EoH	↑0.08%	↑0.36%	↓2.18%	↓3.69%	↓0.13%	↓0.10%

Table 20: Component ablation on constructive AHD at $T = 1000$. TSP is minimized, and KP is maximized.

Method	TSP $N = 50$	KP $N = 50, W = 12.5$
EoH (baseline)	6.545	19.996
Agentic ESOpt + EoH	6.463	20.001
w/o ES (noise-only)	6.497	19.996
w/o cosine schedule	6.480	19.997

Table 21: EoH sampling-temperature ablation on constructive TSP ($N = 50$) at $T = 1000$, with Agentic ESOpt included. Lower is better.

Setting	TSP
EoH (0.6)	6.51782
EoH (1.0)	6.545
EoH (1.5)	6.95927
Agentic ESOpt	6.463

Component and sampling-temperature ablations. The component ablation in Table 20 shows that both the reward-weighted parameter update and the cosine perturbation schedule contribute to the final result: retaining noise without the ES update, or fixing the perturbation radius, weakens performance on both tasks. Table 21 further shows that the gain is not reproduced by retuning the EoH sampling temperature. Temperature 0.6 is the strongest EoH setting, yet Agentic ESOpt remains better on TSP.

Repeated-run significance. We repeat the constructive EoH comparison 20 times per method on TSP ($N = 50$) and KP ($N = 100, W = 25$). Table 22 reports the mean and sample standard deviation **on the training dataset** across runs together with a one-sided, equal-variance t-test. KP values are converted back from the internally negated solver cost to the native maximization objective. The test is computed from the same 20 per-run observations summarized in the table.

Table 22: Repeated-run analysis for constructive AHD. Each method is evaluated over 20 independent runs. Lower is better for TSP and higher is better for the native KP objective.

Task	EoH	Agentic ESOpt + EoH	p-value
TSP ($N = 50, \downarrow$)	6.5517 ± 0.0729	6.5007 ± 0.0868	0.0258
KP ($N = 100, W = 25, \uparrow$)	40.1562 ± 0.0024	40.1578 ± 0.0017	0.0100

Both comparisons are significant at the 0.05 level. The repeated-run analysis therefore supports the consistency of the AHD gains beyond a single search seed.

On-the-fly runtime overhead. We additionally measure end-to-end wall-clock time for the constructive Sample scaffold at $T = 1000$.

Table 23: End-to-end runtime for constructive Sample at $T = 1000$. All values are in minutes.

Task	Sample	Agentic ESOpt + Sample
Design constructive heuristics for TSP	54.8	60.6
Design constructive heuristics for KP	41.2	48.6
Design constructive heuristics for ASP	51.7	56.7

Across the three tasks, attaching Agentic ESOpt adds only 5.0–7.4 minutes, corresponding to a 9.7%–18.0% increase over the original Sample runtime. The parameter updates are therefore carried out nearly on the fly within the existing candidate-generation and evaluation loop, without a separate post-search training stage.

D.7 Cross-Setting Hyperparameter Summary

Table 24 consolidates the Agentic ESOpt configurations used across all reported settings. The task-specific sections above provide the corresponding interaction limits, baseline settings, and evaluation protocols.

Table 24: Cross-setting Agentic ESOpt configurations. "Update batches" counts parameter updates, and "reward load/direction" specifies how each perturbation is scored. All settings use full-parameter, one-sided Gaussian perturbations and population z-score normalization.

Setting	Backbone	Update batches	Directions/update	Reward load/direction	$\sigma_0 \to \sigma_T$	α
Sudoku, $H^* = 5/10$	Qwen3.5-4B	100	32	32 puzzles	$10^{-3} \to 2.5\times10^{-4}$	5×10^{-4}
Sudoku, $H^* = 15$	Qwen3.5-4B	100	32	32 puzzles	$7\times10^{-4} \to 5\times10^{-4}$	5×10^{-4}
Math	Qwen3.5-4B	25	16	16 problems	$10^{-3} \to 5\times10^{-4}$	5×10^{-4}
DocVQA	Qwen3.5-4B	40	16	16 examples	$10^{-3} \to 5\times10^{-4}$	5×10^{-4}
Math	Qwen-30B-A3B	25	16	16 problems	$10^{-3} \to 5\times10^{-4}$	5×10^{-4}
DocVQA	Qwen-30B-A3B	40	16	16 examples	$10^{-3} \to 5\times10^{-4}$	5×10^{-4}
WebArena-Lite	Qwen3.5-27B	70	8	8 web tasks	$1.5\times10^{-3} \to 1.5\times10^{-3}$	2.5×10^{-4}
AHD + EoH, $T = 1{,}000$	Llama-3.1-8B	50	10	one heuristic objective	$10^{-3} \to 0$	5×10^{-4}
AHD + EoH, $T = 2{,}000$	Llama-3.1-8B	50	30	one heuristic objective	$10^{-3} \to 0$	5×10^{-4}
AHD + Sample, $T = 1{,}000/2{,}000$	Llama-3.1-8B	50 / 100	20	one heuristic objective	$10^{-3} \to 0$	5×10^{-4}

For candidate i, the implementation applies $\theta + \sigma_t \epsilon_i$, evaluates the complete trajectory or generated heuristic, reverts the perturbation, and reconstructs the same direction from its integer seed during the update. The implementation uses one-sided perturbations and no explicit $1/\sigma_t$ multiplier, so α is the effective update scale.

The perturbation schedule reflects the optimization objective of each setting. Train-time experiments retain a nonzero terminal radius to preserve neighborhood smoothing: Sudoku uses horizon-specific terminal radii, Math and DocVQA decay to 5×10^{-4}. AHD decays σ to zero because the final objective is the best task-specific heuristic found during test-time search.

For WebArena-Lite, we use a fixed perturbation radius rather than the cosine schedule. Our goal in this experiment is to evaluate full-parameter scalability and compatibility with skill optimization in a costly interactive browser environment, rather than to study perturbation scheduling. Exploring decaying or adaptive perturbation schedules for web-agent adaptation is left to future work.

Population size G and the number of tasks used to score each direction are chosen jointly. Increasing either quantity can improve update stability or task coverage, while also increasing rollout cost. Across settings, binary-reward tasks use multiple tasks per direction, whereas AHD assigns one executable heuristic objective to each direction.

Population z-score normalization provides a common update interface across heterogeneous rewards. It converts binary success, continuous ANLS, and problem-dependent heuristic objectives into relative within-population scores. For AHD, invalid or non-finite programs are mapped to finite within-batch penalties before normalization, and an all-invalid batch produces no parameter update.

Hyperparameter sensitivity and adaptive schedules for G and σ are discussed in Appendix A.

E PROMPTS AND SKILLS

E.1 SUDOKU

E.1.1 ENVIRONMENT PROMPT

Sudoku is implemented as a multi-turn action environment. The agent does not submit a full board in one response. Instead, at each turn it observes the original puzzle, the current board, the current empty-cell list, and optional environment feedback from the previous action, then emits exactly one action in the format `set <row> <col> <value>`. Rows and columns are 1-indexed. The episode terminates when all masked cells are filled, or the action budget is exhausted, and the final reward is binary.

The following is a turn-0 prompt from the evaluation split with 15 masked cells. Empty cells are rendered as ".".

Prompt for Sudoku

```
You are an agent solving Sudoku one action at a time. At each turn, fill exactly one empty cell. Rows and columns are 1-indexed. The
       board is split by | and horizontal lines into nine 3x3 boxes. Every row, every column, and every 3x3 box must contain digits 1
        through 9 exactly once. Choose the cell only from the Current empty cells list. Keep all original givens fixed. Your entire
       response must be exactly one line containing one action and nothing else. Do not explain, reason aloud, wrap the answer in
       code fences, or output any extra words. Use exactly this format:
set <row> <col> <value>

Original puzzle, mask_count=15:
      c1 c2 c3 | c4 c5 c6 | c7 c8 c9
r1:   3 5 9 | 4 . 7 | . . 8
r2:   2 1 8 | . 9 5 | 4 . 6
r3:   4 7 6 | 2 8 1 | 3 5 9
      --------+---------+--------
r4:   6 2 1 | 8 5 3 | 9 4 7
r5:   . 3 5 | 9 . 4 | 6 2 1
r6:   . 4 7 | 6 1 2 | 8 3 5
      --------+---------+--------
r7:   7 6 . | 1 3 8 | 5 9 .
r8:   5 9 . | . 2 6 | 1 . 3
r9:   . 8 3 | 5 4 . | 7 6 2

Current board, turn=0, remaining_empty=15:
      c1 c2 c3 | c4 c5 c6 | c7 c8 c9
r1:   3 5 9 | 4 . 7 | . . 8
r2:   2 1 8 | . 9 5 | 4 . 6
r3:   4 7 6 | 2 8 1 | 3 5 9
      --------+---------+--------
r4:   6 2 1 | 8 5 3 | 9 4 7
r5:   . 3 5 | 9 . 4 | 6 2 1
r6:   . 4 7 | 6 1 2 | 8 3 5
      --------+---------+--------
r7:   7 6 . | 1 3 8 | 5 9 .
r8:   5 9 . | . 2 6 | 1 . 3
r9:   . 8 3 | 5 4 . | 7 6 2

Current empty cells: r1c5, r1c7, r1c8, r2c4, r2c8, r5c1, r5c5, r6c1, r7c3, r7c9, r8c3, r8c4, r8c8, r9c1, r9c6
```

For example, a valid response to this prompt is a single line such as `set 1 5 6`. The environment then applies the action and constructs the next prompt by inserting `Last environment feedback: Filled r1c5 with 6.` before the original puzzle block, updating the current board, setting `turn=1`, decreasing `remaining_empty` to 14, and removing `r1c5` from the current empty-cell list.

E.2 MATH REASONING

E.2.1 REACT PROMPT

Instead of a single direct answer call, the AIME setting is treated as an agentic ReAct environment. The model may call a command-line Python tool multiple times, receives each tool result as an observation, and stops when it emits the final boxed answer. The ES reward is the binary exact-match score of the final answer after the whole trajectory.

Prompt for AIME ReAct

```
You are solving a math reasoning problem. You can think step-by-step and use command-line
     Python when useful.

Action format:
Action:
{
  "name": "bash",
  "arguments": {"command": "<shell command>"}
}
```

```
After each Action, the environment executes the command and returns an Observation. Repeat
     Thought/Action/Observation as needed. When finished, output the final result as \boxed
     {answer}.

Problem:
Patrick started walking at a constant rate along a straight road from school to the park.
     One hour after Patrick left, Tanya started running along the same road from school to
     the park. One hour after Tanya left, Jose started bicycling along the same road from
     school to the park. Tanya ran at a constant rate of 2 miles per hour faster than
     Patrick walked, Jose bicycled at a constant rate of 7 miles per hour faster than Tanya
      ran, and all three arrived at the park at the same time. The distance from the school
      to the park is m/n miles, where m and n are relatively prime positive integers. Find
     m+n.

Thought: Let Patrick's travel time be T and speed be p. I can solve the two equal-distance
     equations with Python.
Action:
{
  "name": "bash",
  "arguments": {"command": "python -c \"import sympy as sp; T,p=sp.symbols('T p'); sol=sp.
       solve([sp.Eq(p,2*T-2), sp.Eq(p,sp.Rational(9,2)*T-9)],[T,p]); T0,p0=sol[T],sol[p]; d
       =sp.factor(T0*p0); print(d, sp.numer(d)+sp.denom(d))\""}
}
Observation: 252/25 277

Thought: The distance is 252/25, so m+n=277.
Final answer: \boxed{277}
```

E.2.2 TRAJECTORY-TO-SKILL COMPOSITION

Math considers trajectories from all 25 ES generations, keeps at most one failed trajectory for each of the 400 training problems, and excludes successful traces from skill distillation. A single post-hoc Trace2Skill pass uses `GPT-5.4-nano`, and temperature 1. We run each model with three seeds to distill the skill (base model, Agentic RL model, Agentic ESOpt model) and select the best result on another 100-size DAPO/DocVQA held-out validation set to report. Final evaluation reconstructs the same No Skill ES updates and injects the distilled skill only into the system context; it performs no further parameter optimization.

E.3 DOCVQA

E.3.1 REACT PROMPT

DocVQA is also an agentic ReAct setting. The task exposes the document image path, and the agent may use command-line or Python tools to inspect the image, crop regions, or run OCR before submitting a short answer. The evaluator scores the final submitted answer with ANLS.

Prompt for DocVQA ReAct

```
Answer the document visual question. Inspect the referenced document image carefully. You
     can use command-line and Python tools over the image file.

[Optional skill instructions are inserted here.]

Action format:
Action:
{
  "name": "bash",
  "arguments": {"command": "<shell command>"}
}
After each Action, the environment returns an Observation. Repeat Thought/Action/
     Observation as needed. When finished, output only the short final answer.

Image path: data\trace2skill\docvqa\images\49153.png
Question: What is the 'actual' value per 1000, during the year 1975?

Thought: I need inspect the document chart and locate the 1975 value for the series labeled
      "actual".
Action:
{
  "name": "bash",
  "arguments": {"command": "python -c \"from PIL import Image; p=r'data\\trace2skill\\
       docvqa\\images\\49153.png'; img=Image.open(p); print(img.size)\""}
}
```

```
Observation: (1695, 2200)

Thought: I should zoom into the chart area around the 1975 tick and read the actual series
     label/value before answering.
Action:
{
  "name": "bash",
  "arguments": {"command": "python -c \"from PIL import Image; p=r'data\\trace2skill\\
       docvqa\\images\\49153.png'; img=Image.open(p); crop=img.crop((250,650,1450,1350));
       crop.save('docvqa_49153_chart_crop.png'); print('saved docvqa_49153_chart_crop.png')
       \""}
}
Observation: saved docvqa_49153_chart_crop.png

Thought: The chart crop shows the 1975 actual value as 0.28 per 1000.
Final answer: 0.28
```

E.3.2 TRAJECTORY-TO-SKILL COMPOSITION

DocVQA selects the final 50 task occurrences and keeps at most one failed and one successful trace per occurrence; when one outcome type is unavailable, only the observed type is retained. A single post-hoc Trace2Skill pass uses `GPT-5.4-nano` and temperature 1. We run each model with three seeds to distill the skill (base model, Agentic RL model, Agentic ESOpt model) and select the best result to report. Final evaluation reconstructs the same No Skill ES updates and injects the distilled skill only into the system context; it performs no further parameter optimization.

E.4 WEBARENA-LITE

E.4.1 ENVIRONMENT PROMPT

WebArena-Lite uses the WebArena id-based accessibility-tree prompt. The full observation can be long, so the box shows the stable prompt structure and one real WebArena-Lite task objective.

Prompt for WebArena-Lite

```
You are an autonomous intelligent agent tasked with navigating a web browser. You will be
     given web-based tasks. The information includes the user's objective, the current
     accessibility tree, the current page address, open tabs, and the previous action.

Available actions include:
click [id]
type [id] [content] [press_enter_after=0|1]
hover [id]
press [key_comb]
scroll [down|up]
goto [url]
go_back
go_forward
stop [answer]

OBSERVATION:
[accessibility-tree nodes with element IDs]
CURRENT PAGE: [administration interface]
OBJECTIVE: What is the top-1 best-selling product in 2022
PREVIOUS ACTION: None

The response should reason briefly and end with:
In summary, the next action I will perform is ```<action>```
```

E.5 AUTOMATIC HEURISTIC DESIGN

E.5.1 TSP-CONSTRUCT PROMPT

For AHD, the agent writes a heuristic function instead of a direct task solution. The following box shows the TSP constructive setting and representative EoH operator prompts. The concrete parent algorithms and code blocks are inserted at the placeholders for crossover or mutation operators.

Prompt for AHD TSP-Construct

```
Task:
Given a set of nodes with their coordinates, find the shortest route that visits each node
     once and returns to the starting node. The task can be solved step-by-step by starting
      from the current node and iteratively choosing the next node. Help me design a novel
     algorithm that is different from algorithms in literature to select the next node in
     each step.

Function:
def select_next_node(current_node, destination_node, unvisited_nodes, distance_matrix):
    return next_node

Input/output information:
'current_node', 'destination_node', 'next_node', and 'unvisited_nodes' are node IDs. '
     distance_matrix' is the distance matrix of nodes. All are Numpy arrays.

EoH operator prompts:
[i1] First, describe your new algorithm and main steps in one sentence. The description
     must be inside a brace. Next, implement it in Python as select_next_node(...). Do not
     give additional explanations.
[e1] Given several existing algorithms and their code, create a new algorithm that has a
     totally different form from the given ones.
[e2] Given several existing algorithms and their code, identify their common backbone idea,
      then create a new algorithm with a different form that is motivated by that backbone.
[m1] Given one existing algorithm and its code, create a modified version with a different
     form.
[m2] Given one existing algorithm and its code, identify the main algorithm parameters and
     create a new algorithm with different parameter settings of the score function.
```

F Licensing and External Components

F.1 Licensing Scope

The code introduced by this study is released under the MIT License. External code, benchmark assets, datasets, model checkpoints, and API services remain governed by their respective upstream licenses or terms. The table below is intentionally limited to repositories whose code or task data are directly invoked, vendored, or copied by the released workflow. Repositories consulted only as references, and baseline repositories whose implementations were not executed, are excluded.

The experiments additionally use Llama and Qwen checkpoints, API-hosted models, DAPO/AIME Math data, and DocVQA data under the terms of their respective authors and providers. Nothing in this release alters or supersedes those terms.

F.2 Upstream Repositories Used by the Released Workflow

Table 25: A summary of licenses and external resources used in this work.

Resources	Type	License	URL
Qwen3.5-4B	Base LLM	Apache-2.0 License	https://huggingface.co/Qwen/Qwen3.5-4B
Qwen3.5-9B	Base LLM	Apache-2.0 License	https://huggingface.co/Qwen/Qwen3.5-9B
Qwen3.5-27B	Base LLM	Apache-2.0 License	https://huggingface.co/Qwen/Qwen3.5-27B
Qwen3-30B-A3B	Base MoE LLM	Apache-2.0 License	https://huggingface.co/Qwen/Qwen3-30B-A3B
LLaMA-3.1-8B-Instruct	Base LLM	Llama 3.1 Community License	https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
verl	RL framework	Apache-2.0 License	https://github.com/volcengine/verl
EoH	AHD framework	MIT License	https://github.com/FeiLiu36/EoH
Trace2Skill	framework	Apache-2.0 License	https://github.com/Qwen-Applications/Trace2Skill
VisualAgentBench	Web-agent runtime	Apache-2.0 License	https://github.com/THUDM/VisualAgentBench
VisualWebArena	Web-agent environment	MIT License	https://github.com/web-arena-x/visualwebarena
AHD Problems	Dataset / Benchmark	MIT License	https://github.com/zz1358m/MCTS-AHD-master
DAPO-Math-17k	Dataset	Apache-2.0 License	https://huggingface.co/datasets/BytedTsinghua-SIA/DAPO-Math-17k
AIME 2026	Dataset	Apache-2.0 License	https://huggingface.co/datasets/math-ai/aime26
DocVQA	Dataset	Challenge Terms	https://site.docvqa.org/datasets
WebArena(-Lite)	Dataset / Benchmark	Research use only	https://github.com/liushiliushi/JitRL