WHEN IS ENOUGH ENOUGH IN SELF-EVOLVING LLM SYSTEMS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving large language model (LLM) systems repeatedly propose, evaluate, and incorporate updates to prompts, skills, or other persistent artifacts. Despite their growing effectiveness, these systems typically operate under a predetermined iteration or compute budget, without a principled criterion to determine when further evolution is no longer worthwhile. This can lead to two undesirable consequences: unnecessary computation after performance has saturated and the risk of returning late updates that overfit or exploit the evaluation signal. These issues motivate us to study two fundamental questions: *when should a self-evolving system stop, and what should it output once it stops?* We address the first by formulating an online sequential testing problem and constructing an anytime-valid restart detector using the per-item paired evaluation outcomes already produced by self-evolving LLM systems. We address the second by formulating a change-point estimation problem and using the estimated transition to select an earlier artifact for output. The resulting procedure is plug-and-play and requires no modification of the underlying self-evolving algorithms. Across two self-evolving frameworks, three LLM model families, and five benchmarks, our method substantially reduces computation costs while maintaining comparable unseen-test performance. For example, on SearchQA with SkillOpt and DeepSeek V4 Flash, our method stops at round 4 rather than the full budget of 40, reducing token usage by 91.6% while achieving 82.43% unseen-test accuracy versus 82.00% under the full-budget run.

1 INTRODUCTION

LLM agents are increasingly designed to improve themselves through repeated interaction, feedback, and adaptation rather than remain fixed after their initial construction. This emerging paradigm of *self-evolving agents* has appeared in a wide range of settings, including iterative self-correction, self-training and self-play, prompt and skill evolution, memory and experience refinement, and multi-agent co-evolution (Madaan et al., 2023; Shinn et al., 2023; Wang et al., 2025; Zhao et al., 2025; Qi et al., 2025; Xia et al., 2025; Chen et al., 2025; Yang et al., 2026a; Agrawal et al., 2026; Yuksekgonul et al., 2025; Ni et al., 2026; Yang et al., 2026b; Zhang et al., 2026b; Li et al., 2026; Zhang et al., 2026c; Shi et al., 2026). Despite substantial differences in what is evolved and how updates are generated, these systems share a similar iterative structure: the system generates or applies an update, evaluates its effect using feedback, reward, or validation signals, and carries the resulting state into subsequent rounds.

In practice, self-evolving loops are typically run for a fixed number of iterations or until a prespecified compute budget is exhausted, regardless of whether further evolution remains meaningful. This can have two undesirable consequences. First, once meaningful improvement has saturated, the system may continue proposing, evaluating, and committing updates that provide little or no additional benefit, incurring substantial unnecessary computation. Second, repeatedly adapting to the same validation signal can encourage evaluation overfitting or reward hacking (Amodei et al., 2016; Skalse et al., 2022), so later updates may appear favorable under the evaluation signal without representing genuine improvement. Figure 1 illustrates how these two consequences can compound in practice: the evolution continues well after meaningful gains have saturated, consuming substantial additional computation, while continued adaptation can ultimately lead to a degraded final artifact on unseen test data.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

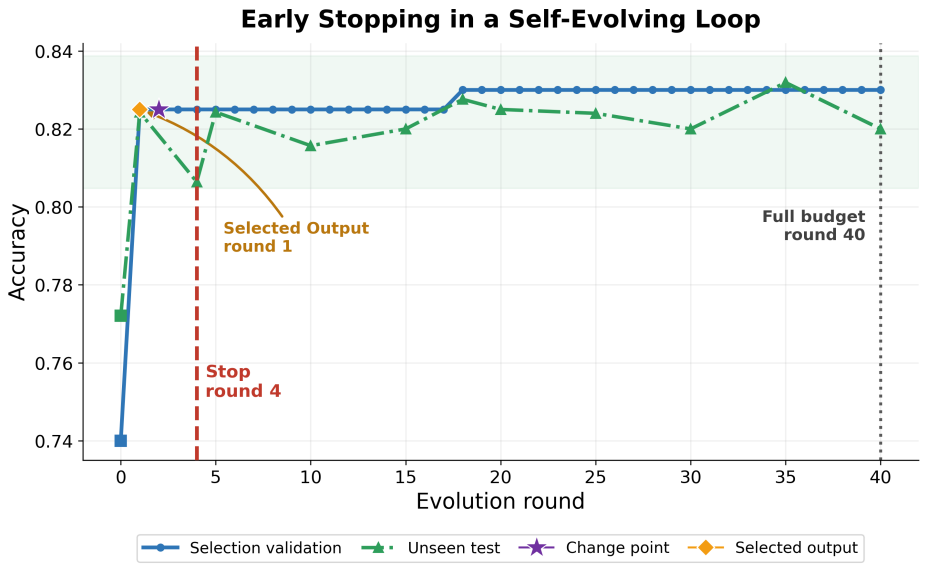


Figure 1: A self-evolving run on SearchQA with DeepSeek V4 Flash. Performance improves during the first few rounds and then plateaus, yet the fixed-budget loop continues to round 40. Our procedure stops at round 4 and selects round 1 as the output, achieving comparable unseen-test performance while avoiding most of the subsequent computation.

This limitation raises two fundamental questions that every self-evolving system must answer:

(i) Given a fixed dataset, when should we stop running the self-evolving loop?
(ii) Once we stop, which artifact/agent should we output?

The first question asks when there is sufficient evidence that further evolution is no longer worthwhile. The second is distinct: even after deciding to stop, the most recent artifact may not be the best one to return and one needs to decide what to output. Current self-evolving systems generally leave both decisions unaddressed in a fixed iteration budget.

We answer these two questions with a unified online procedure using the per-item paired outcomes already produced by the evaluation gate in most existing systems. For *when to stop*, we formulate sequential hypothesis tests and construct an anytime-valid detector that signals when evidence for continued meaningful improvement has weakened, building on testing by betting (Shafer, 2021), betting martingales (Waudby-Smith and Ramdas, 2024), and e-detectors (Shin et al., 2024). For *what to output*, we formulate a change-point estimation problem (Page, 1954; Lorden, 1971): after stopping, we estimate when meaningful improvement ended and return an earlier artifact. The procedure requires no modification to the underlying self-evolving system and uses only outcomes it already computes for incumbents and candidates. In Figure 1, for example, it stops at round 4, estimates the change point at round 2, and returns an earlier artifact from round 1 with comparable unseen-test performance to the full 40-round run while using 91.6% fewer tokens (Table 1).

In detail, we make three main contributions. First, we identify *when to stop* and *what to output* as the two fundamental decisions a self-evolving loop must make, and show that both can be addressed online using the per-item paired outcomes already produced by the system, without modifying the underlying self-evolving algorithm (Section 3.1). Second, we develop a unified statistical procedure that combines an anytime-valid sequential stopping rule with change-point-based output selection: the stopping rule determines when further improvement is no longer worthwhile, while the estimated change point determines which earlier artifact should be returned (Section 3.2). Third, we evaluate the procedure across two self-evolving frameworks, SkillOpt (Yang et al., 2026a) and GEPA (Agrawal et al., 2026), three model families, and five benchmarks, showing substantial reductions in computation while maintaining comparable unseen-test performance (Section 4.2). In

particular, in a benchmark where continued evolution eventually produces a reward-hacking update, our procedure stops before this late degraded artifact is reached and returns an earlier artifact instead.

2 Related work

Self-evolving LLM systems. A rapidly growing body of work lets an LLM improve itself by repeatedly editing a persistent artifact and evaluating it on held-out data: skill documents (Yang et al., 2026a; Ni et al., 2026; Yang et al., 2026b; Zhang et al., 2026b;a;c; Shi et al., 2026; Li et al., 2026), prompts (Agrawal et al., 2026; Yuksekgonul et al., 2025), or programs. These systems differ in what they evolve, but share an analyze–propose–gate loop and report strong gains from it. Their reliability has two known failure points. The loop can be gamed: because it optimizes against its own evaluation signal, degenerate solutions that exploit a leaky benchmark score well without real capability. This is reward hacking, identified as a core safety problem (Amodei et al., 2016), formalized by Skalse et al. (2022), catalogued across environments (Krakovna et al., 2020), and shown to worsen with optimization pressure and agent capability (Pan et al., 2022; Gao et al., 2023); it is now documented in self-evolving agents (Shao et al., 2026). And the per-round commit can be spurious, which Shawn (2026) addresses by recasting the accept-or-reject gate as a sequential test. This line of work sharpens the *individual steps* of the loop, its proposer and its gate, but takes the loop's overall length and its final output as given, leaving open when the loop should stop and which of its artifacts to return.

Sequential change-point detection and optimal stopping. Detecting online when a stream's distribution has changed is classical, from Wald's SPRT (Wald, 1945) and Page's CUSUM (Page, 1954) to the Shiryaev–Roberts procedure (Shiryaev, 1963; Roberts, 1966) and its optimality theory (Lorden, 1971; Pollak, 1985; Lai, 1998; Tartakovsky et al., 2014). These classical procedures typically rely on specified parametric models for the pre- and post-change distributions, whereas our setting is defined through a conditional mean criterion with an unknown change time. We therefore build on anytime-valid inference (Shafer, 2021; Ramdas et al., 2023), specifically the betting martingales of Waudby-Smith and Ramdas (2024) and the e-detectors of Shin et al. (2024), which provide sequential validity under optional stopping. In Appendix G, we review more related work on stopping rules for training and inference.

3 Method

3.1 Problem Formulation

We consider a self-evolving system with initial artifact $\theta_0 \in \Theta$, where Θ may contain prompts, skill documents, or programs. At each round $t \geq 1$, the system uses its current incumbent θ_{t-1} and accumulated information to propose a candidate θ'_t. The evolution mechanism may use training data, search, or reflection to generate a candidate θ'_t. The incumbent and candidate are evaluated on a fixed validation set consisting of n tasks: $D_{\text{val}} = \{x_i\}_{i=1}^n$, with binary correctness $\ell(\theta, x_i) \in \{0, 1\}$ and validation accuracy $L(\theta) = n^{-1} \sum_{i=1}^n \ell(\theta, x_i)$. A greedy gate sets $\theta_t = \theta'_t$ if $L(\theta'_t) > L(\theta_{t-1})$ and retains $\theta_t = \theta_{t-1}$ otherwise. Without a stopping rule, evolution continues until a prescribed round budget N or compute budget is exhausted. One naive approach is to monitor $L(\theta_t)$ and detect the change point. However, tracking only $L(\theta_t)$ is insufficient because rejected proposals leave it unchanged, making a flat trajectory uninformative about proposal quality. For example, an apparent plateau may later be followed by further improvement, as in Figure 3. We therefore accumulate evidence from candidate–incumbent paired outcomes for sequential detection.

To monitor whether proposed updates continue to provide worthwhile gains, let $b_{t,i} = \ell(\theta_{t-1}, x_i)$ and $c_{t,i} = \ell(\theta'_t, x_i)$ denote the incumbent and candidate outcomes on validation task x_i. Define the paired differences and their round-wise average as

$$X_{t,i} = c_{t,i} - b_{t,i} \in \{-1, 0, +1\}, \qquad \text{and} \qquad Z_t = \frac{1}{n} \sum_{i=1}^{n} X_{t,i} = L(\theta'_t) - L(\theta_{t-1}). \tag{1}$$

Each difference $X_{t,i}$ records whether the candidate worsens, preserves, or improves an outcome, regardless of whether it is accepted. Let $\varepsilon > 0$ denote the minimum expected proposal gain considered worthwhile. The gate accepts any positive observed gain, while stopping asks whether the

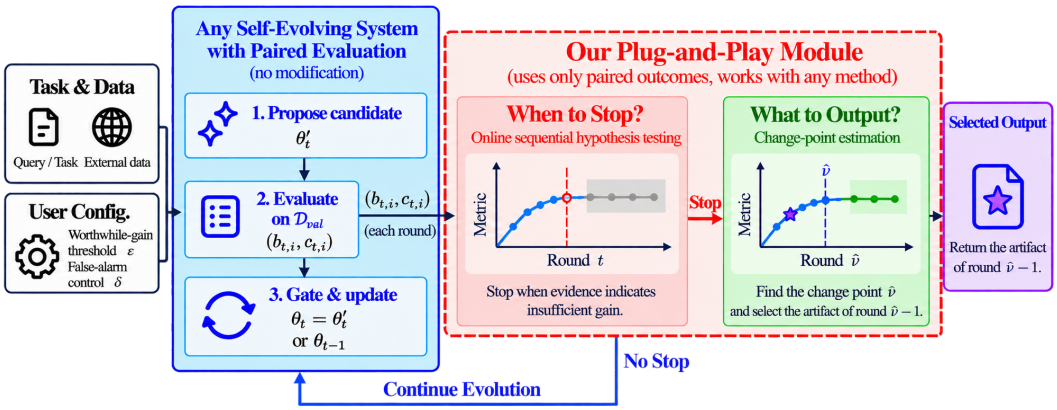


Figure 2: Our stopping and output-selection Plug-and-Play module monitors paired validation outcomes from a self-evolving loop.

conditional expected proposal gain remains at least ε. Thus, a candidate may be accepted even when its expected gain is too small to justify another round. Writing $\mathcal{F}_{t-1}$ for the information available after the round-t candidate is proposed but before its validation outcomes are observed, we define the pre-change condition as

$$\mathbb{E}[Z_t \mid \mathcal{F}_{t-1}] \geq \varepsilon. \tag{2}$$

The round average measures the proposed validation gain, while the paired outcomes retain the item-level information used by our proposed sequential tests in Section 3.2.

We seek to determine both when to stop and which retained artifact to return. Let ν be the first round from which $\mathbb{E}[Z_t \mid \mathcal{F}_{t-1}] < \varepsilon$, and assume this low-improvement regime persists once entered. Our first objective is to detect this transition online by accumulating evidence against the pre-change condition and stopping at an alarm time T_{alarm}, defined by the sequential rule in Section 3.2. Because evidence takes time to accumulate, the alarm can follow the transition by several rounds. The pre-transition output is $\theta_{\nu-1}$, the final retained artifact before expected proposal improvement becomes non-worthwhile. Our second objective is therefore to estimate ν and return $\theta^* = \theta_{\nu-1}$, the incumbent immediately preceding the estimated transition. This separates the time spent gathering evidence from the round whose artifact is selected for output. Figure 2 shows how our stopping and output-selection module interfaces with this loop.

3.2 Proposed Method

We now develop a sequential procedure based on the paired improvement signals $\{X_{t,i}\}_{i=1}^{n}$ defined in Section 3.1. We use these signals to address the two problems formulated above. First, we construct sequential tests of the pre-change condition from different starting rounds and combine their evidence to determine when evolution should stop. Second, we use the same restart statistics to estimate when the non-worthwhile-improvement regime began and determine which earlier incumbent should be returned.

3.2.1 When to Stop: Sequential Testing with Restart Processes

A weak proposal can occur even while further evolution remains worthwhile. We therefore accumulate evidence against continued worthwhile improvement using successive paired outcomes defined before. Following Shin et al. (2024), for each starting round s, we test $H_0^{(s)}$: the pre-change condition in Equation (2) holds for every subsequent round $u \geq s$.

We represent this evidence by a restart wealth process with betting fractions $\lambda_{s,u}$,

$$W_t^{(s)} = \prod_{u=s}^{t} \prod_{i=1}^{n} \left[1 + \lambda_{s,u} \left(\varepsilon - X_{u,i}\right)\right], \qquad \text{with} \qquad W_{s-1}^{(s)} = 1. \tag{3}$$

Algorithm 1 Stopping and output selection for a self-evolving loop

Require: Worthwhile-gain threshold ε, false-alarm control parameter δ.
1: Initialize restart wealth processes.
2: **for** $t = 1, 2, \ldots, N$ **do**
3: Observe $(b_{t,i}, c_{t,i})_{i=1}^{n}$ and set $X_{t,i} = c_{t,i} - b_{t,i}$.
4: Initialize a new restart at $s = t$ and update all wealths with aGRAPA or the fixed mixture.
5: $M_t \leftarrow \max_{1 \le s \le t} W_t^{(s)}$.
6: **if** $M_t \ge 1/\delta$ **then**
7: $T_{\text{alarm}} \leftarrow t$.
8: $\hat{\nu} \leftarrow \arg\max_{1 \le s \le t} W_t^{(s)}$.
9: **stop and return** $\theta_{\hat{\nu}-1}$.
10: **end if**
11: **end for**
12: If no alarm is raised, the stopping module remains inactive and the underlying system returns its final artifact.

For positive betting fractions, an outcome $X_{u,i} < \varepsilon$ contributes a factor greater than one, while an outcome above ε contributes a factor below one. Multiplication combines these contributions across items and rounds, so a large wealth represents accumulated evidence against $H_0^{(s)}$.

We use two betting schemes for $\lambda_{s,u}$. Under adaptive aGRAPA (Waudby-Smith and Ramdas, 2024), $\lambda_{s,t}$ is chosen predictably from past data. See Appendix B for more details. Under the fixed construction, we set $\lambda_{s,t} \equiv \lambda$ for a constant $\lambda \in \Lambda$, and form a fixed mixture by averaging the resulting wealth processes over $\lambda \in \Lambda$. We use $\Lambda = \{0.1, 0.2, 0.4\}$. In the main paper, we report experimental results under adaptive aGRAPA and we evaluate both schemes throughout our experiments in Appendix E.

A process started during useful early rounds may lose wealth and then need to recover those losses before detecting a later stall. We therefore start a fresh process at every round, allowing evidence after a transition to accumulate without losses from earlier gains. Following the CUSUM-style e-detector construction of Shin et al. (2024), we retain the strongest current evidence across starting rounds and raise an alarm when it crosses $1/\delta$, for a user-specified $\delta \in (0, 1)$:

$$M_t = \max_{1 \le s \le t} W_t^{(s)}, \qquad \text{and} \tag{4}$$

$$T_{\text{alarm}} = \inf \left\{ t \ge 1 : M_t \ge \frac{1}{\delta} \right\}. \tag{5}$$

The threshold determines how much evidence is required to stop: smaller δ requires larger wealth and makes the rule more conservative. Under the pre-change condition and the assumptions in Section 3.3, M_t is a valid e-detector, so this threshold guarantees an average run length to false alarm of at least $1/\delta$.

3.2.2 What to Output: Change-Point Estimation

As discussed above, restarting the sequential evidence process at each round helps the procedure respond more quickly once improvement becomes non-worthwhile. The resulting collection of restart processes also provides a natural way to locate when this transition occurred. A process started too early includes pre-transition observations that tend to work against wealth growth, whereas a process started too late misses part of the evidence accumulated after the transition. This suggests that a process started near the transition should attain relatively large wealth at the alarm time. We therefore estimate the change point as the starting round of the restart process with the largest observed wealth at T_{alarm}:

$$\hat{\nu} = \underset{1 \le s \le T_{\text{alarm}}}{\arg\max} \; W_{T_{\text{alarm}}}^{(s)}. \tag{6}$$

As shown in Section 3.3, this intuition can be formalized in the common fixed-λ setting. We then use the estimated change point $\hat{\nu}$ to determine the returned artifact according to θ^*. Thus, the same restart construction is used both to stop the evolution and to select the output artifact: T_{alarm} determines when to stop, while $\hat{\nu}$ determines which artifact to return.

Algorithm 1 summarizes the complete stopping and output-selection procedure.

3.3 Theoretical Analysis

We establish stopping validity and change-point guarantees below. All proofs are in Appendix A.

Stopping validity. Recall that $\mathcal{F}_{t-1}$ includes the proposed candidate but not its validation outcomes. Assume that $X_{t,1}, \ldots, X_{t,n}$ are conditionally independent given $\mathcal{F}_{t-1}$, and each betting fraction $\lambda_{s,t} \in [0, \frac{1}{2}]$ is predictable and fixed across items within round t.

Proposition 1 (Restart and stopping validity)**.** Under these assumptions, for each start s, $H_0^{(s)}$ implies that $W_t^{(s)}$ is a nonnegative supermartingale initialized at one and $\Pr(\sup_{t \geq s} W_t^{(s)} \geq 1/\delta) \leq \delta$ for $\delta \in (0, 1)$. If the pre-change condition holds at every round, then M_t is a valid e-detector: $\mathbb{E}[M_\tau] \leq \mathbb{E}[\tau]$ for every stopping time τ. Consequently, $\mathbb{E}[T_{\text{alarm}}] \geq \frac{1}{\delta}$.

Proposition 1 covers adaptive betting including aGRAPA, while fixed mixtures inherit the same validity by convexity, with average time to false alarm at least $1/\delta$. Intuitively, the detector stops only when enough evidence accumulates that further proposal gains are no longer worthwhile. A smaller δ raises the alarm threshold, increasing the lower bound on $\mathbb{E}[T_{\text{alarm}}]$ under the pre-change regime and lowering the probability of making the false alarm.

Change-point identification and localization. For this analysis, assume one common fixed betting fraction $\lambda \in (0, \frac{1}{2}]$ across all rounds and restarts. Define

$$V_{t,i} = \log(1 + \lambda(\varepsilon - X_{t,i})), \qquad Y_t = \frac{1}{n} \sum_{i=1}^{n} V_{t,i}, \qquad \text{and} \qquad \mu_t = \mathbb{E}[Y_t \mid \mathcal{F}_{t-1}]. \tag{7}$$

Proposition 2 (Change-point identification and localization)**.** Assume that (a) ν marks the sign change in fixed-λ expected log growth: $\mu_t < 0$ for $t < \nu$ and $\mu_t > 0$ for $t \geq \nu$; Then, for any fixed $T \geq \nu$,

$$\nu = \arg\max_{1 \leq s \leq T} \mathbb{E}[\log W_T^{(s)}].$$

For localization, further assume that (b) $\mu_t \leq -\kappa$ before ν and $\mu_t \geq \kappa$ from ν onward for some $\kappa > 0$; (c) Conditioning on $\mathcal{F}_{t-1}$, the $V_{t,i}$ are independent and each centered increment $V_{t,i} - \mathbb{E}[V_{t,i} \mid \mathcal{F}_{t-1}]$ is σ^2-sub-Gaussian ($\sigma > 0$). Then, for any $\alpha \in (0, 1)$, with probability at least $1 - \alpha$, the bound holds uniformly over all finite $T \geq \nu$, including T_{alarm} on $\{\nu \leq T_{\text{alarm}} < \infty\}$:

$$|\hat{\nu}_T - \nu| < \frac{2\sigma^2}{n\kappa^2} \log\left(\frac{2}{\alpha\left(1 - e^{-n\kappa^2/(2\sigma^2)}\right)}\right).$$

Proposition 2 justifies selecting the largest observed restart wealth for localizing the change point. It further shows that for fixed κ and σ, our change point estimation error diminishes at the rate of $1/n$.

4 Experiments

4.1 Setup

We evaluate our method with SkillOpt and GEPA across DeepSeek V4 Flash, Qwen3-32B, and GPT-5.6 Luna, using SearchQA, GSM8K, OfficeQA, LiveMath, and SpreadsheetBench. Main experiments fix SkillOpt and DeepSeek V4 Flash while varying the benchmark; ablations vary the model, evolving system, validation-set size, and detector construction.

Unless otherwise noted, we use $\varepsilon = 0.01$ and false-alarm control parameter $\delta = 0.05$; $t = 0$ denotes the initial artifact. For $\lambda_{s,u}$ in the construction of the wealth process, we run both aGRAPA and the fixed-mixture detector with $\Lambda = \{0.1, 0.2, 0.4\}$. Since the two betting schemes produce the same estimated change point $\hat{\nu}$ across matched settings, with differences appearing only in the alarm round, we report aGRAPA in the main results for conciseness and provide the comparison in Appendix E. We report the alarm round T_{alarm}, estimated change point $\hat{\nu}$, computation consumed through the alarm relative to the full run, and returned artifact $\theta_{\hat{\nu}-1}$. When an unseen test set is available, we compare the returned artifact with the one selected by the full run on the same examples, reporting their accuracies, paired accuracy difference, and its 95% confidence interval.

4.2 Main Results

Our method stops substantially before the fixed budget. The detector saves 48.4%–91.6% of tokens across the three main benchmarks (Table 1). The evolution trajectories and detailed stopping behavior are provided in Appendix H.

Early stopping retains comparable unseen-test performance while improving token efficiency. After the detector stops, our method uses the estimated change point $\hat{\nu}$ to return the incumbent immediately preceding the estimated transition, $\theta_{\hat{\nu}-1}$. We compare this returned artifact with the artifact selected by the full run on the same unseen-test examples. The paired accuracy difference is defined as $\Delta_{\text{acc}} = \text{Acc}_{\text{returned}} - \text{Acc}_{\text{full}}$, and we report its 95% confidence interval. The returned artifact slightly outperforms the full-run selection on SearchQA ($+0.43$ pp), while the differences on GSM8K and OfficeQA are -0.80 pp and -1.16 pp, respectively. In all three cases, the confidence interval includes zero, and the observed differences are small relative to their uncertainty, indicating no clear loss in unseen-test performance from stopping early. At the same time, the reduction in computation substantially improves the cost–performance tradeoff. Measured as millions of tokens per percentage point of unseen-test accuracy gained over the baseline, the returned artifacts are $12.9\times$, $9.3\times$, and $1.8\times$ more token-efficient than running the full evolution budget on SearchQA, GSM8K, and OfficeQA, respectively. The gains are largest on SearchQA and GSM8K, where most of the unseen-test improvement occurs early. Overall, the stopping rule substantially reduces evolution cost while maintaining comparable unseen-test performance in our experiments.

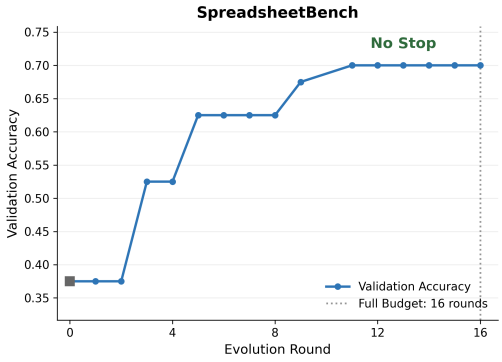


Figure 3: SpreadsheetBench trajectory under SkillOpt with DeepSeek V4 Flash.

The detector stays silent when improvement continues. A useful stopping rule should remain inactive while meaningful improvement continues. SpreadsheetBench serves as a negative control: its validation accuracy rises from 0.375 to 0.70 over 16 rounds, with further gains as late as round 11 (Figure 3). Under the reported aGRAPA configuration ($\varepsilon = 0.01$, $\delta = 0.05$), no alarm is raised within the observed horizon. This complements the main results by showing that the detector remains silent on a trajectory that continues to improve.

Early stopping avoids a degraded late artifact. LiveMath further illustrates the value of combining principled stopping with change-point-based output selection. As shown in Figure 4, SkillOpt makes an early genuine improvement and then enters a long low-improvement regime. If the system were allowed to run for the full budget, it would eventually reach round 16, where it accepts a degenerate skill that instructs the model to answer "A" without reasoning, exploiting the benchmark's skewed answer distribution and producing a large accuracy increase. Our detector instead raises an alarm at $T_{\text{alarm}} = 8$, stopping the evolution well before this late artifact is ever reached. It also estimates the change point at $\hat{\nu} = 2$, and the output-selection rule returns the artifact from round 1. Thus, the stopping rule prevents the system from continuing into the later degraded regime, while the change-point estimate determines which earlier artifact should be returned. The detector does not explicitly identify reward hacking; it stops because sustained meaningful improvement has already ceased.

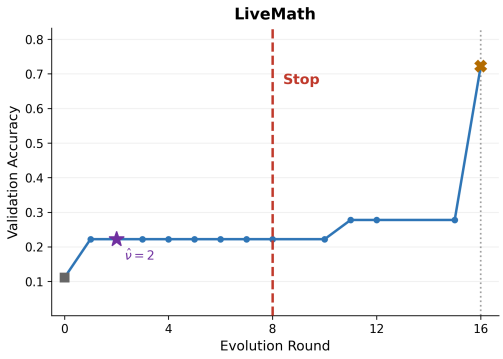


Figure 4: LiveMath trajectory under SkillOpt with DeepSeek V4 Flash.

Table 1: Main results for DeepSeek V4 Flash + SkillOpt with a shared detector configuration ($\varepsilon = 0.01$, $\delta = 0.05$, aGRAPA).

Benchmarks	**SearchQA**	**GSM8K**	**OfficeQA**
Stopping behavior			
Validation size n	200	200	24
Full-run budget N	40	40	16
Alarm round T_{alarm}	4	4	8
Change point $\hat{\nu}$	2	2	5
Computation cost			
Full-run tokens (M)	25.85	11.86	206.90
Tokens to alarm (M)	2.18	1.21	106.72
Tokens saved	**91.6%**	**89.8%**	**48.4%**
Unseen-test performance			
Test samples	1400	500	172
Baseline accuracy (%)	77.21	84.00	14.53
Full-run selected (%)	82.00	98.20	33.72
Returned accuracy (%)	82.43	97.40	32.56
Δ Accuracy (pp)	+0.43	-0.80	-1.16
95% CI of Δ (pp)	$[-1.10, 1.96]$	$[-1.91, 0.31]$	$[-8.72, 6.39]$
Token efficiency			
Full run: Tok / 1% gain (M)	5.39	0.84	10.78
Returned: Tok / 1% gain (M)	0.42	0.09	5.92
Efficiency gain	**12.9**$\times$	**9.3**$\times$	**1.8**$\times$

4.3 Ablation and Sensitivity

We next test whether the stopping rule remains reliable beyond the main experimental setting. Starting from the reference configuration used above, we vary one factor at a time while holding the others fixed. We study robustness across self-evolving systems and underlying models, and then examine sensitivity to the validation-set size and detector parameters.

Across self-evolving systems. We first test whether the stopping rule transfers across different self-evolving algorithms. Holding the underlying model and benchmarks fixed, we replace SkillOpt with GEPA, whose update and selection mechanism differs substantially from SkillOpt. We extract the same type of per-item paired evaluation outcomes from GEPA and apply the detector without changing its configuration. As shown in Table 2, the detector raises alarms at similar stages of the evolution process under both systems and still removes a large fraction of the full-run computation. On SearchQA, GEPA stops at round 5 with $\hat{\nu} = 2$ and saves 73.8% of the estimated token cost; on GSM8K, it stops at round 6 with $\hat{\nu} = 3$ and saves 65.5%. Although GEPA runs are shorter overall, the same stopping rule continues to identify a low-improvement regime well before the full budget is exhausted. GEPA token usage is estimated from metric-call counts; details are provided in Appendix D.

Across models. We next test whether the stopping rule remains effective across different underlying models. Holding SkillOpt, the benchmark, and the detector configuration fixed, we compare DeepSeek V4 Flash, Qwen3-32B, and GPT-5.6 Luna. As shown in Tables 3 and 4, the detector adapts to different evolution speeds rather than stopping at a fixed round. On GSM8K, DeepSeek V4 Flash and GPT-5.6 Luna both saturate quickly and trigger an alarm at round 4, whereas Qwen3-32B continues improving longer and stops at round 8. A similar pattern appears on SearchQA, where Qwen3-32B stops later than DeepSeek V4 Flash. Despite these differences in stopping time, the detector still removes a large fraction of the full-run computation across all model–benchmark combinations, with token savings ranging from 68.9% to 92.6%.

Additional sensitivity results are provided in Appendix F, including experiments with smaller validation sets and variations in the detector parameters (ε, δ). The detector tends to stop later with less validation data, while retaining substantial token savings and comparable unseen-test performance.

Table 2: Robustness across self-evolving systems using DeepSeek V4 Flash and the same detector configuration. We compare SkillOpt and GEPA on the same benchmarks, reporting stopping behavior, computation cost, and unseen-test performance. GEPA token counts are estimated from metric-call counts.

	SearchQA		**GSM8K**	
	SkillOpt	GEPA	SkillOpt	GEPA
Stopping behavior				
Alarm round T_{alarm}	4	5	4	6
Change point $\hat{\nu}$	2	2	2	3
Computation cost				
Tokens to alarm (M)	2.18	2.36	1.21	1.79
Full-run tokens (M)	25.85	8.99	11.86	5.18
Tokens saved	**91.6%**	**73.8%**	**89.8%**	**65.5%**
Unseen-test performance				
Baseline accuracy (%)	77.21	76.86	84.00	84.00
Full-run selected (%)	82.00	82.36	98.20	97.20
Returned accuracy (%)	82.43	82.43	97.40	97.20
Δ Accuracy (pp)	+0.43	+0.07	-0.80	0.00
95% CI of Δ (pp)	$[-1.10, 1.96]$	$[-1.17, 1.32]$	$[-1.91, 0.31]$	$[0.00, 0.00]$

Table 3: Robustness across underlying models on SearchQA using SkillOpt with the same detector configuration. The detector adapts its stopping time to different evolution dynamics while reducing computation and retaining comparable unseen-test performance.

	DeepSeek V4 Flash	Qwen3-32B	GPT-5.6 Luna
Stopping behavior			
Validation size n	200	200	200
Full-run budget N	40	40	40
Alarm round T_{alarm}	4	12	10
Change point $\hat{\nu}$	2	9	3
Computation cost			
Tokens to alarm (M)	2.18	8.08	5.30
Full-run tokens (M)	25.85	32.68	17.03
Tokens saved	**91.6%**	**75.3%**	**68.9%**
Unseen-test performance			
Baseline accuracy (%)	77.21	66.36	69.79
Full-run selected (%)	82.00	72.93	81.07
Returned accuracy (%)	82.43	73.14	80.07
Δ Accuracy (pp)	+0.43	+0.21	-1.00
95% CI (pp)	$[-1.10, 1.96]$	$[-1.45, 1.88]$	$[-2.08, 0.08]$

5 Conclusion and Future Work

We introduced a stopping and output-selection procedure for self-evolving systems that uses sequential betting to determine when further evolution is no longer worthwhile and change-point estimation to select an earlier artifact for output. The procedure operates on paired evaluation outcomes already produced during evolution and requires no modification to the underlying system. Across multiple benchmarks, models, and self-evolving systems, it stops well before the fixed budget is exhausted while substantially reducing computation and retaining comparable unseen-test performance.

Future work could extend the localization guarantee to adaptive betting schemes such as aGRAPA, study data-driven ways to calibrate the stopping rule, and generalize the framework to richer evaluation signals beyond binary correctness. It would also be useful to examine how the method behaves under different validation-set sizes, noisier feedback, and more complex self-evolving systems in which proposals are generated and evaluated through longer interaction loops. More broadly, our results suggest that self-evolving systems need not rely on predetermined fixed budgets: sequential evidence can guide when to stop, while change-point estimation can help determine which earlier artifact to return.

Ethics Statement

This work studies when to stop self-evolving AI systems in order to avoid unnecessary computation while preserving useful improvements. Our experiments use existing benchmarks and language models and do not involve human subjects or the collection of new personal data. The proposed stopping procedure may help reduce redundant model calls and associated computational cost. As with other evaluation-based stopping methods, its behavior depends on the representativeness of the validation data, and care should be taken when applying it in high-stakes settings.

Reproducibility Statement

We provide the full stopping procedure, theoretical assumptions, and proofs in the main text and appendix. Experimental settings include the worthwhile-gain threshold ε, false-alarm control parameter δ, betting constructions, validation-set sizes, model and benchmark configurations, and alarm-time comparison results. We also report both adaptive aGRAPA and fixed-mixture betting schemes, with their comparison provided in the appendix. Additional ablations cover models, evolving systems, validation-set sizes, and detector configurations.

AI Use Statement

In this work, large language models (LLMs) were used solely for language refinement during manuscript preparation, including grammar correction, clarity improvement, and the identification of potentially ambiguous wording. All research ideas, methodological decisions, experiments, theoretical derivations, numerical results, and conclusions were independently checked and approved by the authors. The authors take full responsibility for the content of the paper.

References

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

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Rich Caruana, Steve Lawrence, and C. Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2001.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhan, Mostofa Patwary, and Jiaxuan You. Multi-agent evolve: LLM self-improve through co-evolution, 2025.

Young Hyun Cho and Will Wei Sun. When should an AI workflow release? always-valid inference for black-box generate-verify systems, 2026.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning (ICML)*, 2023.

Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. Specification gaming: The flip side of AI ingenuity. Google DeepMind, 2020. URL `https://deepmind.google/blog/specification-gaming-the-flip-side-of-ai-ingenuity/`.

Tze Leung Lai. Information bounds and quick detection of parameter changes in stochastic systems. *IEEE Transactions on Information Theory*, 44(7):2917–2929, 1998.

Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. Arise: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning. *arXiv preprint arXiv:2603.16060*, 2026.

Gary Lorden. Procedures for reacting to a change in distribution. *The Annals of Mathematical Statistics*, 42(6):1897–1908, 1971.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Aman Mehta. When agents commit too soon: Diagnosing premature commitment in LLM agents, 2026.

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, 2026.

E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations (ICLR)*, 2022.

Moshe Pollak. Optimal detection of a change in distribution. *The Annals of Statistics*, 13(1):206–227, 1985.

Lutz Prechelt. Early stopping — but when? In *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 53–67. Springer, 2nd edition, 2012.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, 2025.

Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023. doi: 10.1214/23-STS894.

Garvesh Raskutti, Martin J. Wainwright, and Bin Yu. Early stopping and non-parametric regression: An optimal data-dependent stopping rule. *Journal of Machine Learning Research*, 15:335–366, 2014.

S. W. Roberts. A comparison of some control chart procedures. *Technometrics*, 8(3):411–430, 1966.

Glenn Shafer. Testing by betting: A strategy for statistical and scientific communication. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 184(2):407–431, 2021. doi: 10.1111/rssa.12647.

Shuai Shao, Qihan Ren, Dongrui Liu, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, and Jing Shao. Your agent may misevolve: Emergent risks in self-evolving llm agents. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 99728–99793, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a24cd16bc361afa78e57d31d34f3d936-Paper-Conference.pdf.

Zayx Shawn. PACE: Anytime-valid acceptance tests for self-evolving agents, 2026.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*, 2026.

Jaehyeok Shin, Aaditya Ramdas, and Alessandro Rinaldo. E-detectors: A nonparametric framework for sequential change detection. *The New England Journal of Statistics in Data Science*, 2(2):229–260, 2024. doi: 10.51387/23-NEJSDS51.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

A. N. Shiryaev. On optimum methods in quickest detection problems. *Theory of Probability & Its Applications*, 8(1):22–46, 1963.

Sahil Shrivastava. Semantic early-stopping for iterative LLM agent loops, 2026.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Alexander Tartakovsky, Igor Nikiforov, and Michèle Basseville. *Sequential Analysis: Hypothesis Testing and Changepoint Detection*. Chapman & Hall/CRC, 2014.

Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16 (2):117–186, 1945.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning, 2025.

Ian Waudby-Smith and Aaditya Ramdas. Estimating means of bounded random variables by betting. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(1):1–27, 2024. doi: 10.1093/jrsssb/qkad009.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning, 2025.

Yangxinyu Xie, Tao Wang, Soham Mallick, Yan Sun, Georgy Noarov, Mengxin Yu, Tanwi Mallick, Weijie J. Su, and Edgar Dobriban. Statistical early stopping for reasoning models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills, 2026a.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026b.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639:609–616, 2025. doi: 10.1038/s41586-025-08661-4.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. Evoskills: Self-evolving agent skills via co-evolutionary verification, 2026a.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026b.

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, Qingnan Ren, Shun Zou, Wenxuan Huang, Lin Chen, Zehui Chen, and Feng Zhao. Skillflow: Benchmarking lifelong skill discovery and evolution for autonomous agents, 2026c.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

A PROOFS

We prove the two propositions in Section 3.3. Throughout, recall that

$$Z_t = \frac{1}{n}\sum_{i=1}^{n} X_{t,i},$$

and that the pre-change condition is

$$\mathbb{E}[Z_t \mid \mathcal{F}_{t-1}] \geq \varepsilon. \tag{8}$$

We assume that $X_{t,1}, \ldots, X_{t,n}$ are conditionally independent given $\mathcal{F}_{t-1}$ and that each betting fraction $\lambda_{s,t} \in [0, \frac{1}{2}]$ is predictable.

A.1 VALIDITY OF EACH RESTART WEALTH PROCESS

Proof. Fix a restart time s. For $t \geq s$, write the multiplicative increment of the restart wealth as

$$L_t^{(s)} = \prod_{i=1}^{n} \left[1 + \lambda_{s,t}\left(\varepsilon - X_{t,i}\right)\right],$$

so that

$$W_t^{(s)} = W_{t-1}^{(s)} L_t^{(s)}, \qquad W_{s-1}^{(s)} = 1.$$

We first verify non-negativity. Since $X_{t,i} \in \{-1, 0, 1\}$, $\varepsilon \in (0, 1)$, and $\lambda_{s,t} \in [0, \frac{1}{2}]$,

$$1 + \lambda_{s,t}(\varepsilon - X_{t,i}) \geq 1 + \frac{1}{2}(\varepsilon - 1) = \frac{1+\varepsilon}{2} > 0.$$

Hence $W_t^{(s)}$ is nonnegative.

Next, define

$$\mu_{t,i} = \mathbb{E}[X_{t,i} \mid \mathcal{F}_{t-1}].$$

Because $\lambda_{s,t}$ is $\mathcal{F}_{t-1}$-measurable and the item-level signals are conditionally independent given $\mathcal{F}_{t-1}$,

$$\mathbb{E}[L_t^{(s)} \mid \mathcal{F}_{t-1}] = \prod_{i=1}^{n} \left[1 + \lambda_{s,t}\left(\varepsilon - \mu_{t,i}\right)\right]. \tag{9}$$

All factors in the product are positive, so by the arithmetic–geometric mean inequality,

$$\mathbb{E}[L_t^{(s)} \mid \mathcal{F}_{t-1}] \leq \left[\frac{1}{n}\sum_{i=1}^{n}\left(1 + \lambda_{s,t}(\varepsilon - \mu_{t,i})\right)\right]^n \tag{10}$$

$$= \left[1 + \lambda_{s,t}\left(\varepsilon - \frac{1}{n}\sum_{i=1}^{n}\mu_{t,i}\right)\right]^n. \tag{11}$$

Since

$$\frac{1}{n}\sum_{i=1}^{n}\mu_{t,i} = \mathbb{E}[Z_t \mid \mathcal{F}_{t-1}] \geq \varepsilon$$

under the pre-change condition, we obtain

$$\mathbb{E}[L_t^{(s)} \mid \mathcal{F}_{t-1}] \leq 1.$$

Therefore,

$$\mathbb{E}[W_t^{(s)} \mid \mathcal{F}_{t-1}] = W_{t-1}^{(s)}\mathbb{E}[L_t^{(s)} \mid \mathcal{F}_{t-1}] \tag{12}$$

$$\leq W_{t-1}^{(s)}. \tag{13}$$

Thus $\{W_t^{(s)}\}_{t \geq s-1}$ is a nonnegative supermartingale initialized at one.

Ville's inequality then gives, for any $h > 0$,

$$\Pr\left(\sup_{t \geq s} W_t^{(s)} \geq h\right) \leq \frac{1}{h}.$$

□

A.2 E-Detector Validity and False-Alarm Run-Length Guarantee

We next show that combining the restart processes through the maximum restart wealth produces a valid e-detector. This construction corresponds to the CUSUM-style e-detector of Shin et al. (2024).

Proof. Extend each restart process by setting $W_t^{(s)} = 1$ for $t < s$. By Proposition 1, each $\{W_t^{(s)}\}_{t\geq 0}$ is a nonnegative restart e-process under the pre-change condition.

Recall that

$$M_t = \max_{1\leq s\leq t} W_t^{(s)}.$$

For any stopping time τ,

$$M_\tau \leq \sum_{s=1}^{\tau} W_\tau^{(s)}.$$

Taking expectations and writing the random upper limit using indicators gives

$$\mathbb{E}[M_\tau] \leq \sum_{s=1}^{\infty} \mathbb{E}\left[\mathbf{1}\{s \leq \tau\} W_\tau^{(s)}\right]. \tag{14}$$

Because $\{s \leq \tau\} \in \mathcal{F}_{s-1}$ and $W^{(s)}$ is a valid restart e-process, optional stopping gives

$$\mathbb{E}\left[W_\tau^{(s)} \mid \mathcal{F}_{s-1}\right] \leq 1.$$

Therefore,

$$\mathbb{E}\left[\mathbf{1}\{s \leq \tau\} W_\tau^{(s)}\right] \leq \Pr(s \leq \tau). \tag{15}$$

Summing over s,

$$\mathbb{E}[M_\tau] \leq \sum_{s=1}^{\infty} \Pr(s \leq \tau) \tag{16}$$

$$= \mathbb{E}[\tau]. \tag{17}$$

Hence M_t satisfies the defining e-detector condition

$$\mathbb{E}[M_\tau] \leq \mathbb{E}[\tau]$$

for every stopping time τ, and is therefore a valid e-detector in the sense of Shin et al. (2024).

Now recall the alarm rule

$$T_{\text{alarm}} = \inf\left\{t \geq 1 : M_t \geq \frac{1}{\delta}\right\}.$$

If T_{alarm} is not almost surely finite, then $\mathbb{E}[T_{\text{alarm}}] = \infty$ and the desired result holds immediately. Otherwise,

$$M_{T_{\text{alarm}}} \geq \frac{1}{\delta}.$$

Applying the e-detector inequality at $\tau = T_{\text{alarm}}$ gives

$$\frac{1}{\delta} \leq \mathbb{E}[M_{T_{\text{alarm}}}] \leq \mathbb{E}[T_{\text{alarm}}],$$

and therefore

$$\mathbb{E}[T_{\text{alarm}}] \geq \frac{1}{\delta}.$$

Thus the stopping rule has average run length to false alarm of at least $1/\delta$ whenever the pre-change condition continues to hold. □

A.3 Population Characterization of the Change Point

Proof of the population conclusion in Proposition 2. For this result, consider the fixed betting fraction λ used in Proposition 2. Taking logarithms of the restart wealth gives

$$\log W_T^{(s)} = \sum_{u=s}^{T} \sum_{i=1}^{n} \log\left(1 + \lambda(\varepsilon - X_{u,i})\right).$$

Let $g_u = n\mathbb{E}[Y_u]$, with Y_u defined in Equation (7). By the tower property, $g_u < 0$ before ν and $g_u > 0$ from ν onward. Thus,

$$\mathbb{E}[\log W_T^{(s)}] = \sum_{u=s}^{T} g_u. \tag{18}$$

Suppose first that $s < \nu$. Then

$$\mathbb{E}[\log W_T^{(s)}] = \sum_{u=s}^{\nu-1} g_u + \sum_{u=\nu}^{T} g_u \tag{19}$$

$$< \sum_{u=\nu}^{T} g_u = \mathbb{E}[\log W_T^{(\nu)}], \tag{20}$$

because $g_u < 0$ for all $u < \nu$.

Now suppose that $s > \nu$. Then

$$\mathbb{E}[\log W_T^{(s)}] = \sum_{u=s}^{T} g_u \tag{21}$$

$$= \sum_{u=\nu}^{T} g_u - \sum_{u=\nu}^{s-1} g_u \tag{22}$$

$$< \sum_{u=\nu}^{T} g_u = \mathbb{E}[\log W_T^{(\nu)}], \tag{23}$$

because $g_u > 0$ for all $u \geq \nu$.

Thus every $s \neq \nu$ has strictly smaller expected log restart wealth than $s = \nu$, and hence

$$\nu = \arg\max_{1 \leq s \leq T} \mathbb{E}[\log W_T^{(s)}].$$

This is a population-level characterization for the common fixed-λ setting considered here. For the corresponding realized restart wealth, the logarithm is strictly increasing, so

$$\arg\max_s W_T^{(s)} = \arg\max_s \log W_T^{(s)}.$$

Thus realized wealth and realized log wealth induce the same restart selection in this setting. □

A.4 Change-Point Localization Error Bound

Proof of the localization bound in Proposition 2. Recall that

$$Y_t = \frac{1}{n} \sum_{i=1}^{n} V_{t,i}, \qquad V_{t,i} = \log\left(1 + \lambda(\varepsilon - X_{t,i})\right),$$

and therefore

$$\log W_T^{(s)} = n \sum_{u=s}^{T} Y_u.$$

By assumption, conditional on $\mathcal{F}_{t-1}$, the centered variables $V_{t,i} - \mathbb{E}[V_{t,i} \mid \mathcal{F}_{t-1}]$ are independent and σ^2-sub-Gaussian. Hence,

$$Y_t - \mathbb{E}[Y_t \mid \mathcal{F}_{t-1}]$$

is conditionally σ^2/n-sub-Gaussian.

First consider a starting round $s < \nu$. Since the processes initialized at s and ν share all factors from round ν through T,

$$\log W_T^{(s)} - \log W_T^{(\nu)} = n \sum_{u=s}^{\nu-1} Y_u.$$

Let $k = \nu - s$. Then

$$W_T^{(s)} \geq W_T^{(\nu)} \quad \Longleftrightarrow \quad \sum_{u=s}^{\nu-1} Y_u \geq 0.$$

Since $\mathbb{E}[Y_u \mid \mathcal{F}_{u-1}] \leq -\kappa$ for $u < \nu$,

$$\sum_{u=s}^{\nu-1} \mathbb{E}[Y_u \mid \mathcal{F}_{u-1}] \leq -k\kappa.$$

Thus, defining

$$D_u = Y_u - \mathbb{E}[Y_u \mid \mathcal{F}_{u-1}],$$

the event $\sum_{u=s}^{\nu-1} Y_u \geq 0$ implies

$$\sum_{u=s}^{\nu-1} D_u \geq k\kappa.$$

Because each D_u is conditionally σ^2/n-sub-Gaussian, the sum over k rounds is conditionally $k\sigma^2/n$-sub-Gaussian. Therefore,

$$\Pr\left(W_T^{(s)} \geq W_T^{(\nu)}\right) \leq \exp\left(-\frac{nk\kappa^2}{2\sigma^2}\right).$$

Now consider $s > \nu$. In this case,

$$\log W_T^{(s)} - \log W_T^{(\nu)} = -n \sum_{u=\nu}^{s-1} Y_u.$$

Let $k = s - \nu$. Then

$$W_T^{(s)} \geq W_T^{(\nu)} \quad \Longleftrightarrow \quad \sum_{u=\nu}^{s-1} Y_u \leq 0.$$

Since $\mathbb{E}[Y_u \mid \mathcal{F}_{u-1}] \geq \kappa$ for $u \geq \nu$,

$$\sum_{u=\nu}^{s-1} \mathbb{E}[Y_u \mid \mathcal{F}_{u-1}] \geq k\kappa.$$

Hence the event $\sum_{u=\nu}^{s-1} Y_u \leq 0$ implies

$$\sum_{u=\nu}^{s-1} D_u \leq -k\kappa.$$

Applying the lower-tail sub-Gaussian bound gives

$$\Pr\left(W_T^{(s)} \geq W_T^{(\nu)}\right) \leq \exp\left(-\frac{nk\kappa^2}{2\sigma^2}\right).$$

If $|\hat{\nu}_T - \nu| \geq m$, then there exists some starting round s with $|s - \nu| \geq m$ such that $W_T^{(s)} \geq W_T^{(\nu)}$. Applying a union bound over starting points on both sides of ν gives

$$\Pr\left(|\hat{\nu}_T - \nu| \geq m\right) \leq 2 \sum_{k=m}^{\infty} \exp\left(-\frac{nk\kappa^2}{2\sigma^2}\right).$$

Evaluating the geometric series yields

$$\Pr\left(|\hat{\nu}_T - \nu| \geq m\right) \leq \frac{2\exp\left(-\frac{nm\kappa^2}{2\sigma^2}\right)}{1-\exp\left(-\frac{n\kappa^2}{2\sigma^2}\right)}.$$

To obtain the equivalent high-probability form, let $\alpha \in (0,1)$ and choose m such that

$$\frac{2\exp\left(-\frac{nm\kappa^2}{2\sigma^2}\right)}{1-\exp\left(-\frac{n\kappa^2}{2\sigma^2}\right)} \leq \alpha.$$

Solving for m gives

$$m \geq \frac{2\sigma^2}{n\kappa^2}\log\left(\frac{2}{\alpha\left(1-\exp\left(-\frac{n\kappa^2}{2\sigma^2}\right)\right)}\right).$$

Let r_α denote the right-hand side and choose $m = \lceil r_\alpha \rceil$. Since $|\hat{\nu}_T - \nu|$ is integer-valued, the events $\{|\hat{\nu}_T - \nu| \geq r_\alpha\}$ and $\{|\hat{\nu}_T - \nu| \geq m\}$ coincide. Therefore, with probability at least $1-\alpha$,

$$|\hat{\nu}_T - \nu| < \frac{2\sigma^2}{n\kappa^2}\log\left(\frac{2}{\alpha\left(1-\exp\left(-\frac{n\kappa^2}{2\sigma^2}\right)\right)}\right),$$

which proves the stated result. □

B BETTING FRACTION AND DETECTABILITY FLOOR

By Proposition 1, any predictable $\lambda_{s,t} \in [0, \frac{1}{2}]$ yields a valid restart wealth process. We instantiate the betting rule in two ways: adaptive aGRAPA and a fixed mixture.

Fixed mixture. Fix a finite grid $\Lambda = \{\lambda^{(1)}, \ldots, \lambda^{(m)}\} \subset [0, \frac{1}{2}]$ and nonnegative weights w_j summing to 1; we use uniform weights over $\Lambda = \{0.1, 0.2, 0.4\}$. For each start s, run one wealth process per grid value and average them:

$$\overline{W}_t^{(s)} = \sum_j w_j\, W_t^{(s,\lambda^{(j)})}.$$

For the fixed-mixture betting scheme, this averaged wealth serves as the restart wealth $W_t^{(s)}$ used by the stopping and output-selection procedure.

Proposition 3 (The mixture is valid)**.** If each $W_t^{(s,\lambda^{(j)})}$ is a nonnegative supermartingale under H_0, then so is $\overline{W}_t^{(s)}$, and the false-alarm bound of Proposition 1 applies with the same threshold $1/\delta$.

Proof. Non-negativity is immediate. By linearity of conditional expectation,

$$\mathbb{E}\big[\overline{W}_t^{(s)} \mid \mathcal{F}_{t-1}\big] = \sum_j w_j\, \mathbb{E}\big[W_t^{(s,\lambda^{(j)})} \mid \mathcal{F}_{t-1}\big] \leq \sum_j w_j\, W_{t-1}^{(s,\lambda^{(j)})} = \overline{W}_{t-1}^{(s)},$$

and $\overline{W}_{s-1}^{(s)} = \sum_j w_j = 1$. A convex combination of nonnegative supermartingales is a nonnegative supermartingale. □

The mixture requires no online estimation and remains a valid supermartingale regardless of sample size.

Growth-optimal plug-in (aGRAPA). A Kelly gambler maximizes $\mathbb{E}[\log(1+\lambda G_{t,i})]$; a second-order expansion yields the approximate maximizer $\lambda^\star = \mu/(\sigma^2+\mu^2)$ with $\mu = \mathbb{E}[G_{t,i}]$ and $\sigma^2 = \mathrm{Var}(G_{t,i})$. Replacing the moments by predictable plug-ins over $[s, t-1]$ and truncating to $[0, \frac{1}{2}]$ gives, for a gambler started at s,

$$\lambda_{s,t} = \min\left(\frac{\max(\hat{\mu}_{s,t-1},\, 0)}{\hat{\sigma}^2_{s,t-1} + \hat{\mu}^2_{s,t-1}},\ \tfrac{1}{2}\right), \quad \hat{\mu}_{s,t-1} = \varepsilon - \bar{Z}_{s,t-1}, \quad \hat{\sigma}^2_{s,t-1} = \bar{q}_{s,t-1} - \bar{Z}^2_{s,t-1},$$

where $\bar{Z}_{s,t-1} = \frac{1}{t-s}\sum_{u=s}^{t-1} Z_u$ and $\bar{q}_{s,t-1} = \frac{1}{t-s}\sum_{u=s}^{t-1} \frac{n_{01}(u)+n_{10}(u)}{n}$ is the mean discordant rate; on the empty window set $\lambda_{s,s} = \lambda_0$ for a fixed $\lambda_0 \in (0, \frac{1}{2}]$. We set the ratio to zero when $\hat{\sigma}^2_{s,t-1} + \hat{\mu}^2_{s,t-1} = 0$. The moment identities follow from $G = \varepsilon - X$ with $X \in \{-1, 0, +1\}$: $\mathbb{E}[G] = \varepsilon - \mathbb{E}[X]$ gives $\hat{\mu} = \varepsilon - \bar{Z}$, while $X^2 = \mathbf{1}\{X \neq 0\}$ gives $\mathrm{Var}(G) = \mathrm{Var}(X) = \bar{q} - \bar{Z}^2$, with ε cancelling.

Proposition 4 (aGRAPA is valid)**.** The fraction $\lambda_{s,t}$ above is predictable and lies in $[0, \frac{1}{2}]$, hence the aGRAPA wealth process is a nonnegative supermartingale under H_0 by Proposition 1.

Proof. The window $[s, t-1]$ excludes round t, so $\hat{\mu}_{s,t-1}$ and $\hat{\sigma}^2_{s,t-1}$ are $\mathcal{F}_{t-1}$-measurable and $\lambda_{s,t}$ is predictable. The inner $\max(\cdot, 0)$ and outer $\min(\cdot, \frac{1}{2})$ force $\lambda_{s,t} \in [0, \frac{1}{2}]$, with $\lambda_{s,t} = 0$ when the denominator vanishes. Validity follows from Proposition 1. □

Detectability floor.

Proposition 5 (Approximate plateau length required to fire)**.** On a pure plateau with $Z_u = 0$, so every item pays $G_{u,i} = \varepsilon$, a single gambler with constant fraction λ multiplies its wealth by $(1+\lambda\varepsilon)^n$ per round. Starting from wealth of order 1 at the onset of the plateau, the approximate number of consecutive plateau rounds required to reach $1/\delta$ is

$$T_{\mathrm{need}} \approx \frac{\log(1/\delta)}{n \log(1 + \lambda\varepsilon)}.$$

For the mixture over a grid Λ, this becomes

$$T_{\mathrm{need}} \approx \frac{\log(1/\delta) + \log|\Lambda|}{n \log(1 + \lambda_{\max}\varepsilon)}, \qquad \lambda_{\max} = \max \Lambda.$$

Proof. On a plateau round every item pays ε, so the per-round wealth factor of a constant-λ gambler is $(1 + \lambda\varepsilon)^n$ and after T rounds the log-wealth grows by $T\, n \log(1 + \lambda\varepsilon)$. Setting this equal to $\log(1/\delta)$ and solving for T gives the first display; for the mixture, the fastest-growing grid value dominates up to a factor $|\Lambda|$, adding $\log|\Lambda|$. □

Corollary 1 (Resolution scales with the validation set)**.** For small $\lambda\varepsilon$, $\log(1+\lambda\varepsilon) \approx \lambda\varepsilon$, so $T_{\mathrm{need}} \approx (\log(1/\delta) + \log|\Lambda|)/(n\lambda_{\max}\varepsilon)$. For a fixed target plateau length the smallest resolvable ε scales like $1/n$: doubling D_{val} halves the worthwhile gain the detector can resolve.

When n or ε is small relative to the available plateau length, the detector correctly stays silent. This is not a failure: validity (Proposition 1) holds regardless of n, and only power degrades.

C Paired Unseen-Test Comparison

We compare the artifact returned by our stopping procedure and the artifact selected by the full run on the same unseen-test examples. Because each example produces a paired binary correctness outcome, the observations can be summarized by the 2×2 table

	Full incorrect	Full correct
Returned incorrect	n_{00}	n_{01}
Returned correct	n_{10}	n_{11}

.

Here, n_{01} counts examples answered incorrectly by the returned artifact but correctly by the full-run artifact, while n_{10} counts examples on which the reverse occurs.

Paired accuracy difference. For $N = n_{00} + n_{01} + n_{10} + n_{11}$ paired examples, the difference in test accuracy is

$$\widehat{\Delta}_{\mathrm{acc}} = \widehat{\mathrm{Acc}}_{\mathrm{returned}} - \widehat{\mathrm{Acc}}_{\mathrm{full}} = \frac{n_{10} - n_{01}}{N}.$$

Thus, positive values favor the returned artifact and negative values favor the full-run artifact. Let

Table 4: Robustness across underlying models on GSM8K using SkillOpt with the same detector configuration.

	DeepSeek	Qwen	Luna
Stopping behavior			
Validation size n	200	200	200
Full-run budget N	40	40	40
Alarm round T_{alarm}	4	8	4
Change point $\hat{\nu}$	2	2	2
Computation cost			
Tokens to alarm (M)	1.21	2.79	0.84
Full-run tokens (M)	11.86	18.04	11.33
Tokens saved	**89.8%**	**84.5%**	**92.6%**
Unseen-test performance			
Baseline accuracy (%)	84.00	78.80	24.40
Full-run selected (%)	98.20	95.80	95.80
Returned accuracy (%)	97.40	94.20	94.60
Δ Accuracy (pp)	-0.80	-1.60	-1.20
95% CI (pp)	$[-1.91, 0.31]$	$[-3.35, 0.15]$	$[-2.86, 0.46]$

$$\hat{q} = \frac{n_{01} + n_{10}}{N}$$

denote the fraction of discordant pairs. Since the per-example paired difference takes values in $\{-1, 0, +1\}$, an estimated standard error is

$$\widehat{\mathrm{SE}} = \sqrt{\frac{\hat{q} - \widehat{\Delta}_{\mathrm{acc}}^2}{N}}.$$

We report the corresponding normal-approximation 95% confidence interval

$$\widehat{\Delta}_{\mathrm{acc}} \pm 1.96\, \widehat{\mathrm{SE}}.$$

The paired accuracy difference quantifies both the magnitude and uncertainty of the performance change. We report it together with its 95% confidence interval in Table 1.

D GEPA ALIGNMENT AND TOKEN ESTIMATION

Seed alignment. SkillOpt begins from a fixed skill-document template with an empty skill body, whereas GEPA begins from an initial prompt. To make the two systems' starting points comparable, we align the GEPA seed to SkillOpt's template: the seed prompt carries the same task framing, with an empty-skill placeholder in place of SkillOpt's initial skill body. This ensures that round $t{=}0$ denotes a comparable "no learned content" state in both systems, so that alarm times and $\hat{\nu}$ are measured from the same origin.

Per-item outcomes. GEPA's gate selects along a Pareto front over minibatch subscores rather than scoring on a single full-validation pass. We read the per-item outcomes $b_{t,i}, c_{t,i}$ that the detector consumes directly from GEPA's saved validation-subscore matrix, which records each candidate's per-item correctness on the shared validation split.

Token estimation. SkillOpt logs token usage per agent call, so its token counts are exact. GEPA's public logs record the number of metric calls (LM inferences over the validation set) made during discovery, but not a per-call token breakdown. We therefore estimate GEPA token usage as

$$\text{tokens} \;\approx\; (\text{number of metric calls}) \times (\text{average tokens per call}),$$

Table 5: Alarm / $\hat{\nu}$ for a single configuration ($\varepsilon = 0.01$, $\delta = 0.05$) across SkillOpt and GEPA settings. "mix" denotes the fixed mixture with $\Lambda = \{0.1, 0.2, 0.4\}$, and "aGRAPA" denotes the adaptive growth-optimal plug-in.

System	Benchmark	Model	mix	aGRAPA
SkillOpt	SearchQA (n=200)	DeepSeek V4 Flash	5/2	4/2
	SearchQA (n=200)	Qwen3-32B	12/9	12/9
	SearchQA (n=200)	GPT-5.6 Luna	11/3	10/3
	SearchQA (n=100)	DeepSeek V4 Flash	10/4	6/4
	GSM8K (n=200)	DeepSeek V4 Flash	5/2	4/2
	GSM8K (n=200)	Qwen3-32B	9/2	8/2
	GSM8K (n=200)	GPT-5.6 Luna	5/2	4/2
	GSM8K (n=100)	DeepSeek V4 Flash	8/2	5/2
GEPA	SearchQA	DeepSeek V4 Flash	5/2	5/2
	GSM8K	DeepSeek V4 Flash	6/3	6/3

with the average tokens per call derived from the matched SkillOpt–DeepSeek rollout on the same benchmark: 2249 tokens/call on SearchQA and 1354 tokens/call on GSM8K. Because a metric call is an LM inference over the same validation set in both systems, the per-call cost is comparable across systems, so this estimate is accurate up to differences in prompt length between the two seeds. The GEPA token figures in Table 2 should be read as estimates on this basis; the alarm and $\hat{\nu}$ values, which depend only on the per-item outcomes, are exact.

E Alarm-time comparison

Table 5 compares the alarm time and change-point estimate under the two betting schemes used in our experiments: the fixed mixture and aGRAPA. Across all matched settings, the two schemes produce the same $\hat{\nu}$, while the alarm time differs by at most a few rounds. The main text reports the aGRAPA results for conciseness.

F Sensitivity Analysis

Sensitivity to validation-set size. We examine how the detector behaves when less validation data are available. Holding the self-evolving system, underlying model, and detector configuration fixed, we reduce the validation set size from $n = 200$ to $n = 100$. As shown in Table 6, the detector generally stops later with fewer validation examples, since each round provides less evidence. On SearchQA, the alarm moves from round 4 to round 6, while on GSM8K it moves from round 4 to round 5. Despite the smaller validation sets, the method still saves more than 85% of the total token cost, and the paired unseen-test accuracy differences remain small, with 95% confidence intervals containing zero.

Sensitivity to detector parameters. We further examine sensitivity to the improvement threshold ε and false-alarm control parameter δ. Table 7 reports the alarm round T_{alarm} and estimated change point $\hat{\nu}$ across a grid of parameter values. Larger values of ε or δ generally lead to earlier stopping, while the estimated change point remains relatively stable across a broad range of settings.

G Additional Related Work

Stopping rules for training and inference. Early stopping has long been used in model training to reduce computation and mitigate overfitting, typically by terminating optimization when validation performance ceases to improve according to a patience rule or threshold (Prechelt, 2012; Caruana et al., 2001). Subsequent work has studied more principled, data-dependent stopping criteria for iterative learning procedures (Yao et al., 2007; Raskutti et al., 2014). Related stopping mechanisms have also been explored in modern LLM systems, primarily at inference time. Semantic

Table 6: Sensitivity to validation-set size using SkillOpt with DeepSeek V4 Flash and the same detector configuration. Smaller validation sets provide less evidence per round and therefore lead to slightly later stopping, while substantial computation is still saved and unseen-test performance remains comparable.

	SearchQA		**GSM8K**	
	$n = 200$	$n = 100$	$n = 200$	$n = 100$
Stopping behavior				
Full-run budget N	40	40	40	40
Alarm round T_{alarm}	4	6	4	5
Change point $\hat{\nu}$	2	4	2	2
Computation cost				
Tokens to alarm (M)	2.18	2.26	1.21	0.83
Full-run tokens (M)	25.85	16.10	11.86	6.52
Tokens saved	**91.6%**	**85.9%**	**89.8%**	**87.3%**
Unseen-test performance				
Baseline accuracy (%)	77.21	77.21	84.00	84.00
Full-run selected (%)	82.00	82.43	98.20	97.20
Returned accuracy (%)	82.43	81.79	97.40	97.60
Δ Accuracy (pp)	+0.43	-0.64	-0.80	+0.40
95% CI of Δ (pp)	$[-1.10, 1.96]$	$[-1.79, 0.50]$	$[-1.91, 0.31]$	$[-0.71, 1.51]$

Table 7: Sensitivity of the alarm round T_{alarm} and estimated change point $\hat{\nu}$ to (ε, δ). Each entry reports $T_{\text{alarm}}/\hat{\nu}$; "–" indicates that no alarm is raised within the evolution horizon.

SearchQA ($n = 200$)

$\varepsilon\backslash\delta$	0.01	0.05	0.10
0.002	10/2	4/2	4/2
0.005	5/2	4/2	4/2
0.01	4/2	4/2	3/2
0.02	4/2	3/2	3/2
0.05	3/2	3/2	3/2

GSM8K ($n = 200$)

$\varepsilon\backslash\delta$	0.01	0.05	0.10
0.002	8/2	7/2	5/2
0.005	7/2	5/2	5/2
0.01	5/2	4/2	4/2
0.02	4/2	3/2	3/2
0.05	3/2	3/2	3/2

LiveMath ($n = 18$)

$\varepsilon\backslash\delta$	0.01	0.05	0.10
0.002	–	12/2	8/2
0.005	–	12/2	7/2
0.01	–	8/2	7/2
0.02	12/2	7/2	6/2
0.05	8/2	6/2	5/2

OfficeQA ($n = 24$)

$\varepsilon\backslash\delta$	0.01	0.05	0.10
0.002	9/5	8/5	7/5
0.005	9/5	8/5	7/5
0.01	9/5	8/5	7/5
0.02	9/5	8/5	7/5
0.05	8/4	6/4	6/4

patience terminates iterative multi-agent refinement when successive outputs stabilize (Shrivastava, 2026); statistical stopping rules have been proposed to limit overthinking within a reasoning trajectory (Xie et al., 2026); hidden-state diagnostics have been used to detect premature commitment during inference (Mehta, 2026); and always-valid e-processes have been applied to determine when a generate–verify procedure has accumulated sufficient evidence to release an output (Cho and Sun, 2026). These approaches study when to terminate training or inference under their respective signals and objectives. In contrast, our setting concerns a self-evolving propose–evaluate–commit loop over persistent artifacts, where stopping must be determined from repeated paired validation outcomes and is coupled with the additional problem of selecting which previously retained artifact to return.

H MAIN EVOLUTION TRAJECTORIES

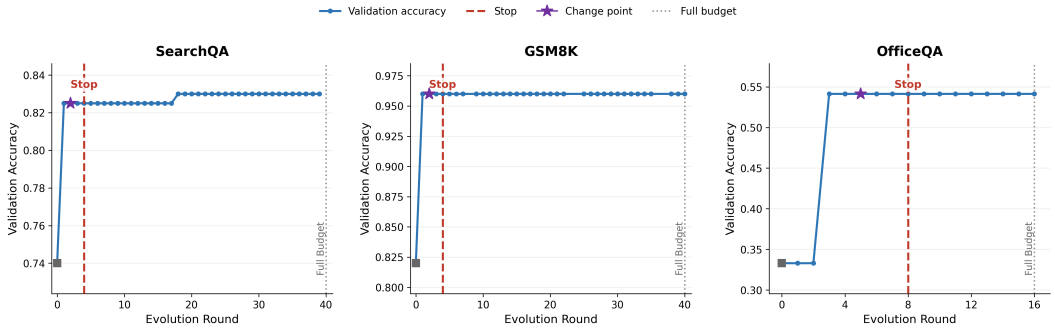


Figure 5: Evolution trajectories under SkillOpt with DeepSeek V4 Flash on SearchQA, GSM8K, and OfficeQA. The red dashed line marks T_{alarm}, the purple star marks $\hat{\nu}$, and the gray dotted line denotes the full evolution budget.

As shown in Figure 5 and Table 1, the detector raises an alarm well before the full-run budget is exhausted on all three benchmarks. On SearchQA and GSM8K, it stops at round 4 out of 40, reducing token usage from 25.85M to 2.18M and from 11.86M to 1.21M, respectively. This corresponds to token savings of 91.6% and 89.8%. OfficeQA is substantially more expensive per evolution round, yet the detector still stops at round 8 out of 16 and reduces total token usage by 48.4%. These results indicate that fixed iteration budgets can spend a substantial fraction of computation after the main performance gains have already occurred.

The estimated change points are $\hat{\nu} = 2$ for SearchQA and GSM8K and $\hat{\nu} = 5$ for OfficeQA. They precede the respective alarm rounds because the detector needs additional observations to accumulate sufficient stopping evidence.