VERIFICATION AS A BUDGET: POSTERIOR-SEPARATION SCHEDULING FOR SELF-EVOLVING AGENT SKILLS

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing a natural-language skill document instead of model weights is a low-cost path for adapting frozen LLM-based agents to downstream tasks, and mainstream methods share one outer loop of target-model rollouts, optimizer proposals, validation and accept-or-reject decisions. This paper identifies a structural deficiency in that loop: validation is fixed as a full-scale step. Every candidate is evaluated once on the full validation set and the gate compares that noisy reading against the incumbent's recorded score, a score systematically inflated by selection bias whose raised threshold crowds out genuine late-training gains; rejected candidates cannot be resurrected and their measurements accumulate no evidence. Meanwhile target-model rollouts and optimizer calls, two operations with very different cost structures, are bound at a fixed ratio set by a step counter and insensitive to the current estimation quality. We propose VeriTree, which restructures the outer loop as a skill tree with uncertainty estimates. Each round performs either additional validation or the generation of a new candidate, and the switch is decided in closed loop by a two-arm criterion on posterior separation: validation stops and expansion proceeds once the leading candidate is statistically separated from its strongest competitor, or once separating them would cost more validation than the remaining budget allows. Validation targets are chosen by uncertainty-aware sampling, optimizer calls are subject to data gating, and delivery favors well-evidenced nodes; the whole behavior is controlled by a single parameter set from the task configuration. Across three optimizer–target configurations and five tasks, compared with SkillOpt and GEPA under identical accounting, VeriTree attains comparable or better accuracy, while reducing training cost to 10%–63% of SkillOpt's on every task and under every configuration. The results support the paper's central claim: validation in self-evolving agents should be treated not as a fixed full-scale step but as a scarce budget scheduled by posterior state.

1 INTRODUCTION

Adapting large language models (LLMs) to downstream tasks by reinforcement learning typically needs thousands to tens of thousands of rollouts per task (Shao et al., 2024; Agrawal et al., 2026). An alternative uses language itself as the learning medium, since execution trajectories can be serialized as natural language and reflected upon directly by a model, converting a handful of rollouts into substantial gains (Agrawal et al., 2026). Agent skills follow this direction, packaging tool policies, domain conventions and failure modes into reusable natural-language artifacts (Wang et al., 2024; Zhao et al., 2024; Anthropic, 2025); a skill enters the prompt directly at deployment and adds no inference calls, which for frozen frontier models is one of the few available adaptation mechanisms.

Letting a stronger and more expensive optimizer model read the target model's trajectories and propose edits to the skill document improves skill quality effectively (Yang et al., 2026; Ni et al., 2026). Most such methods share one outer loop: execute with the current skill, have the optimizer analyze successes and failures and propose edits, evaluate the edited version on a validation set, and accept or reject it by validation score, the decision we call the gate (Zhou et al., 2023; Pryzant et al., 2023; Guo et al., 2024; Agrawal et al., 2026); SkillOpt (Yang et al., 2026) is among the most effective instances. This paper studies the cost structure of that loop and asks a per-round scheduling

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.

question: given the evidence accumulated by each version and the gaps among them, should the next unit of budget measure existing versions more accurately, or generate new ones?

The validation stage of these methods has two structural limitations. Evaluation is noisy, with independent re-evaluations of one version fluctuating by several percentage points, and since acceptance requires a single reading to exceed all prior records, the recorded score is an inflated reading selected by bias rather than the true quality, cached as a threshold that never decreases; later candidates then face an inflated benchmark and genuine late-training gains inside the noise band are systematically killed. Separately, target-model rollouts and optimizer calls have different cost structures yet are bound to a fixed cadence blind to the evidence state. Section 2 quantifies both.

This paper proposes **VeriTree** (Figure 1), which replaces the chain with a tree carrying uncertainty estimates. Each node maintains an uncertainty-aware estimate of its own accuracy and is never deleted for a single low score. Each round performs exactly one action, either adding validation to an existing node to tighten its estimate, or having the optimizer generate a new node from some node's unused trajectories. The switch is governed not by a round count but by whether the leading node is statistically separated from its strongest competitor: once separated, further validation cannot change the outcome, and when separation is infeasible within the remaining budget the switch fires as well. Validation flows preferentially to the nodes most likely to change the ranking, the optimizer is called only once a node has enough unused trajectories, and the best-evidenced node among all historical nodes is delivered.

To summarize, our contributions are as follows:

- **Diagnosis.** We attribute the token waste of chain-style gates to three testable mechanisms, a monotonically rising threshold inflated by selection bias, full-scale validation insensitive to version quality, and the non-reusability of rejected versions, and we measure the noise floor of the validation stage by re-evaluation (Section 2).
- **Method.** We propose VeriTree, a closed-loop validation-budget scheduler in which a posterior-separation criterion decides the switch, uncertainty-aware sampling picks validation targets, data gating governs optimizer calls, and delivery favors well-evidenced nodes (Section 3).
- **Experiments.** Across three optimizer–target configurations and five tasks, against SkillOpt and GEPA under identical accounting, VeriTree attains comparable or better accuracy, at 10%–63% of SkillOpt's training cost (median 20%) and the lowest cost in all 15 cells; in raw tokens it is likewise below both baselines in all 15 cells (14%–78% of SkillOpt's, 35%–92% of GEPA's). A three-point ablation verifies that the scheduling behavior is predictable (Section 4).

2 Linear Skill Self-Evolution and Its Cost

Linear skill self-evolution involves two models and three data splits. The frozen target model M executes tasks with the skill $\mathcal{S}$, which enters the prompt directly at deployment; the stronger and more expensive optimizer model reads execution trajectories and proposes edits. The training set D_{tr} supplies the rollouts of each step and its trajectories are the optimizer's material, the validation set D_{val} serves acceptance decisions, and the test set D_{te} is locked until the final evaluation. Write $\rho(\mathcal{S})$ for the true accuracy of a skill; since sampling and the execution environment are stochastic, a single evaluation $\mathrm{acc}(\mathcal{S}; D)$ is only a noisy reading of it.

Training proceeds as a chain of skills $\{\mathcal{S}_t\}_{t\geq 0}$, with $\mathcal{S}_0$ the initial skill and $\mathcal{S}_t$ the incumbent at step t. Each step executes the incumbent on the training batch $Q_t \subset D_{\mathrm{tr}}$ to produce trajectories τ_t, has the optimizer emit a candidate $\tilde{\mathcal{S}}_t = \mathrm{Opt}(\mathcal{S}_t, \tau_t)$, evaluates it once on the validation set for $\tilde{y}_t = \mathrm{acc}(\tilde{\mathcal{S}}_t; D_{\mathrm{val}})$, and compares the reading against the incumbent's recorded score y_t^{rec}:

$$\mathcal{S}_{t+1} = \begin{cases} \tilde{\mathcal{S}}_t, & \tilde{y}_t > y_t^{\mathrm{rec}}, \\ \mathcal{S}_t, & \text{otherwise}, \end{cases} \qquad y_{t+1}^{\mathrm{rec}} = \max(\tilde{y}_t,\ y_t^{\mathrm{rec}})\,. \tag{1}$$

A candidate becomes the incumbent only when its reading strictly exceeds the record, which is then updated; otherwise it is discarded and nothing changes. Neither the record nor the incumbent is ever

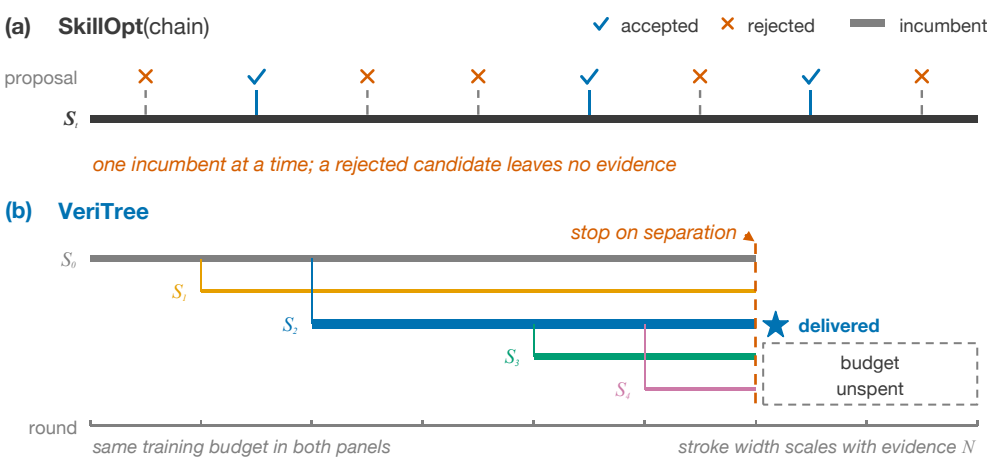


Figure 1: The two outer loops as version lifelines. (a): SkillOpt: a single chain in which one incumbent is active at a time; every step consumes a fixed bundle of one rollout batch, one proposal, and one full validation, and a rejected candidate is discarded, leaving no evidence behind. (b): VeriTree: lifelines of all candidates coexist, with the width of each lifeline proportional to its accumulated evidence N; each round consumes only one batch of s questions and performs either validation or expansion, chosen by the switching criterion, and no node is deleted.

re-evaluated. The objective is to maximize the true accuracy of the delivered skill, the version with the highest recorded score, within a budget of steps. SkillOpt disciplines this paradigm with explicit epochs and batches, a train–validation split and a strict held-out gate, and is the direct baseline here. Three structural features of the loop drive the analysis below: rollout, proposal and full validation are bound at a fixed 1:1:1 cadence driven by step counts that reads no estimation quality; only one skill is active at a time, so a rejected candidate is discarded and later proposals can depart only from the incumbent; and the gate always compares a single noisy reading against one cached value.

These features apply the same measurement resources to candidates of every quality: whatever its true gap to the incumbent, a candidate receives exactly one full-validation evaluation. The central question is therefore where the next unit of budget should go, and the linear structure answers it once and for all, independently of the training state, wasting budget statistically and structurally.

The statistical failure stems from opposing a single noisy reading to a cached point value. On OlympiadBench, five parallel re-evaluations of the same $\mathcal{S}_0$ on one validation batch differ by 5.47 pp between their highest and lowest readings (Figure 2a), and across the five tasks this range falls in 2–5 pp; the gate's decision resolution is by contrast one question's worth, $1/|D_{\text{val}}| \approx 0.78$ pp for the OlympiadBench split of 128, an order of magnitude finer than the noise floor. Eq. 1 further makes y_t^{rec} a running maximum of accepted readings: acceptance requires exceeding the record, so the record never decreases, its expectation exceeds the true accuracy ρ, and the deviation scales with the re-evaluation noise. Late in training the genuine gaps between successive versions shrink to 1–2 pp (measured over ten adjacent version pairs; Appendix D), so they fall inside the noise band and are rejected by the inflated threshold. The errors are asymmetric: a false acceptance entrenches noise as the new threshold, whereas a false rejection is permanent, since rejected candidates cannot be resurrected and leave no accumulable evidence. Requiring a decision finer than the measurement can resolve is the statistical root of the failure, and deciding from single-evaluation point values is the common convention across iterative prompt optimization (Appendix A).

The structural failure lies in the fixed spending template. Target-model rollouts are cheap per unit and each yields a batch of trajectories, whereas optimizer calls are expensive and their proposal quality depends on whether that material covers enough failure modes. Fixing their cadence at 1:1 blinds the schedule to two states that change over training: whether the incumbent is already statistically separated from its strongest challenger, after which further measurement carries almost no information, and whether the accumulated trajectories suffice for a reliable proposal, below which

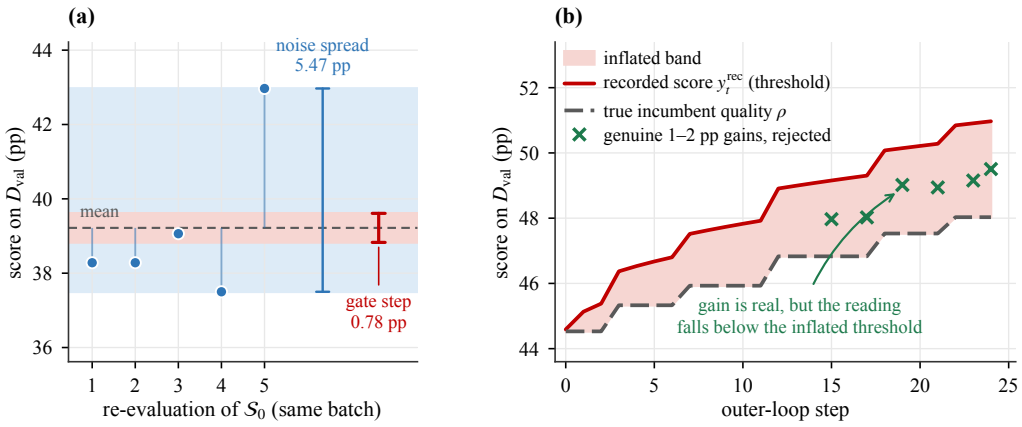


Figure 2: The noise band of validation against the decision resolution of the gate, on OlympiadBench. (a): the scores of five parallel re-evaluations of the initial skill $\mathcal{S}_0$ on the same validation batch of 128 questions, whose highest and lowest readings differ by 5.47 pp in this instance, drawn against the gate resolution of 0.78 pp, the width of a single question in that split; across the five tasks the corresponding range falls in 2–5 pp. (b): selection bias. The recorded score that serves as the gate threshold is a running maximum, so it rises with the acceptance history and its expectation drifts above the true quality of the incumbent, and late-training gains of 1–2 pp then fall inside the inflated band and are rejected.

proposals are near-random perturbations. Under the first, full validation keeps paying for a settled decision; under the second, the optimizer call and the validation that follows both produce nothing.

The two failures share a root: the decision layer consumes point values and accumulates no evidence, while the budget layer spends by a fixed template and reads no evidence state. Material and measurement also sit in separate pools, since training trajectories are consumed once and accumulate no estimate, while validation rollouts yield only decisions and never become proposal material. The remedy is therefore not to tune the existing loop but to change how the budget is allocated.

The diagnosis reduces to three design requirements. **R1**: represent scores as posterior distributions carrying uncertainty, so that erroneous readings can be corrected by later evidence, rather than as permanently cached point values. **R2**: stop validating once further sampling cannot change the delivery decision, rather than paying an equal full evaluation for every candidate. **R3**: treat supplementary validation and candidate generation as two independent budget actions, whose switching and whose optimizer calls read the estimation quality and the material state rather than a counter. Section 3 realizes the three in turn.

3 Method: VeriTree

VeriTree realizes the three requirements of Section 2 as a closed-loop scheduler: posterior node representations for R1, the two-arm switching criterion for R2, and two independent budget actions with data gating for R3. Figure 3 gives an overview; Appendix C lists every formula in order of appearance and Algorithm 1 states the full procedure.

3.1 Paradigm and Budget Accounting

VeriTree organizes candidate skills as a tree whose root is the initial skill $\mathcal{S}_0$ and whose children the optimizer proposes from trajectories accumulated at a parent node; the proposal pipeline, trajectory format and evaluation harness are in Appendix B. Training runs on a question pool of size U for E epochs, $B_q = U \cdot E$ questions in total, each round consuming a batch of s questions under a hard cap of $T = B_q/s$ rounds, with the remaining budget b_q decreasing monotonically. The test set stays independent and is evaluated exactly once after training.

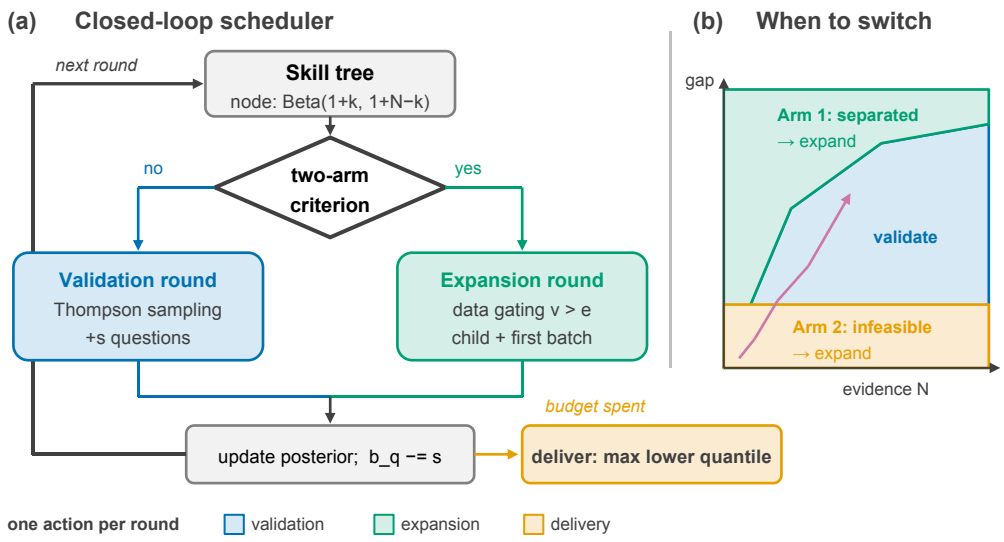


Figure 3: VeriTree as a closed-loop scheduler. (a): The loop framework. Nodes of the skill tree carry Beta posteriors; the two-arm criterion selects one action per round; a validation round updates the posterior of the sampled node, an expansion round draws a parent from the data-gated eligible set ($v_a > e_a$) and lets the optimizer emit a child that immediately receives its first validation batch; after budget exhaustion, delivery takes the node with the highest lower posterior quantile. (b): when the switch fires, drawn in the plane of the leader-to-competitor gap against accumulated evidence N. Large gaps with enough evidence fall in the separated region of Arm 1; small gaps whose resolution would exceed the remaining budget fall in the infeasible region of Arm 2; the intermediate region keeps validating. Both boundaries move as the remaining budget b_q falls.

3.2 Nodes, Posteriors, and the Two Actions

A node a is a candidate skill carrying the correct-answer count k_a, the evaluated-question count N_a, the validation-round count v_a, the number of batches consumed as a parent e_a, and its trajectory batches in chronological order. Each question's outcome is Bernoulli, so with a uniform prior the conjugate posterior and its smoothed mean are

$$\theta_a \sim \mathrm{Beta}(1 + k_a,\ 1 + N_a - k_a)\,, \qquad \hat{\mu}_a = \frac{1 + k_a}{2 + N_a}. \tag{2}$$

Equation 2 is the node representation that realizes R1. Smoothing prevents extreme frequencies from few observations from passing as trustworthy high scores, and the posterior width encodes uncertainty, so distinguishing a high score from a trustworthy one requires only reading the posterior, with no separate threshold.

Each round performs exactly one action and consumes one batch of s questions. *Validation* picks an existing node, rolls out on the next batch, updates (k, N, v) and appends the trajectories. *Expansion* picks an eligible parent (Section 3.4) and has the optimizer generate a child, which immediately receives a first validation batch, since a node without evidence cannot enter any criterion. One batch of rollouts thus serves twice, updating a posterior and remaining available as proposal material, which is the unified accounting that Section 2 calls for. No ratio is enforced between the two actions; it is set round by round by the criterion of Section 3.3 from the posterior state. Round 1 forcibly validates $\mathcal{S}_0$ as a cold start, so every later criterion has non-zero evidence.

3.3 Criterion: Switching Between Expansion and Validation

Continued validation is worth its cost only while it can change the delivery decision; once it cannot, sampling should stop, which here means expanding. This is the structure of fixed-confidence best-

arm identification (Even-Dar et al., 2006; Kaufmann et al., 2016; Garivier & Kaufmann, 2016), realized here as two arms sharing one parameter δ and matching that literature's two stopping grounds: sufficient confidence, and futility.

Arm 1 (separation attained). Under the assumption that node posteriors are mutually independent, the probability that node a is the best in the tree is

$$P_{\text{top}}(a) = \int_0^1 f_a(x) \prod_{b \neq a} F_b(x)\, dx, \tag{3}$$

where f_a is the posterior density of a and F_b the distribution function of the others, evaluated by one-dimensional quadrature. When $\max_a P_{\text{top}}(a) \geq 1 - \delta$ the round expands, the nodes being orderable with sufficient confidence. Stopping once the posterior probability that the recommended arm is optimal exceeds $1 - \delta$ is the standard Bayesian rule of best-arm identification (Russo, 2020; Shang et al., 2020), known to coincide at corresponding thresholds with the frequentist Chernoff rule (Garivier & Kaufmann, 2016) and standard in Bayesian adaptive trial design (Wason & Trippa, 2014). Since the criterion reads the separation of the whole tree, it may fire when the leader's posterior is narrow and refuse to fire when the competitor's is too wide to confirm, making the information bottleneck the object of validation.

Arm 2 (infeasible within the remaining budget). The separation criterion fails on ties: when the leader and its strongest competitor are close in true quality, P_{top} never reaches $1 - \delta$ and the budget is exhausted confirming nothing. With L the node of highest posterior mean, C the second, and $\hat{\Delta} = \hat{\mu}_L - \hat{\mu}_C$, the additional questions per node needed to separate them at confidence $1 - \delta$, a plug-in estimate under the normal approximation, is

$$n^* = \frac{z_{1-\delta}^2 \left[\hat{\mu}_L(1 - \hat{\mu}_L) + \hat{\mu}_C(1 - \hat{\mu}_C)\right]}{\hat{\Delta}^2}, \tag{4}$$

where $z_{1-\delta} = \Phi^{-1}(1 - \delta)$; this is the standard sample-size formula for comparing two proportions with the unknown parameters replaced by current estimates (Fleiss et al., 2003). When $\left[(n^* - N_L)^+ + (n^* - N_C)^+\right] > b_q$ the round expands, the tie being indistinguishable within the remaining budget; when $\hat{\Delta} \leq 0$, $n^* = \infty$ and expansion proceeds directly. Terminating when the lookahead from current data exceeds the available sample size is futility stopping by conditional power (Lachin, 2005; Gallo et al., 2014), except that here the lookahead targets the remaining question budget and termination redirects that budget to expansion rather than to exit.

When fewer than two nodes have $N > 0$, the round expands. There is no livelock: b_q decreases monotonically and Arm 2's condition loosens as the budget tightens, so training ends no later than budget exhaustion. Unlike open-loop rules that time expansion by a counting power law over evaluations and nodes, this criterion reads posterior separation and budget feasibility, so tree size is determined by the evidence state rather than a fixed cadence.

3.4 Validation-Target and Parent Selection

Validation tokens should flow to nodes that can still change the ranking and whose posteriors remain wide, so VeriTree draws one independent posterior sample per node with $N > 0$,

$$S_a \sim \text{Beta}(1 + k_a,\ 1 + N_a - k_a)\,, \tag{5}$$

and validates the node with the largest sample (Thompson sampling; Thompson, 1933; Chapelle & Li, 2011). Nodes with few rounds have thick right tails and are sampled with some probability, while nodes with many rounds but low means are rarely selected yet never deleted, so a wide posterior preserves a channel of re-assessment: this is R1 realized on the validation side. Replacing the sampler with $\arg\max \hat{\mu}$ would let early noise peaks monopolize validation and the entrenchment of Section 2 would recur (Appendix G).

At expansion time, the parent is drawn from the eligible set $\mathcal{E} = \{a : v_a > e_a \wedge \hat{\mu}_a > 0\}$ with weights proportional to the posterior means,

$$P(\text{selecting } a) = \frac{\hat{\mu}_a}{\sum_{b \in \mathcal{E}} \hat{\mu}_b}, \tag{6}$$

where $v_a > e_a$ is the data gate: a node is eligible as a parent only while it holds unconsumed batches of trajectories, and each expansion consumes its earliest unused batch. The gate prevents optimizer calls on insufficient material and makes repeated expansions of the same parent draw on different evidence, the material side of R3. The weights use the smoothed mean so that extreme frequencies from few observations cannot monopolize priority. A failed proposal falls back to validation without incrementing e_a, so failure does not consume material eligibility.

3.5 Final Selection

After training ends, the node with the highest lower quantile of the posterior is delivered among all historical nodes with $N > 0$:

$$a^* = \arg \max_{a:\, N_a > 0} \; F^{-1}_{\mathrm{Beta}(1+k_a,\, 1+N_a-k_a)}\left(\frac{\varepsilon}{100}\right), \tag{7}$$

where $\varepsilon = 1$ is a fixed convention. Delivering by the lower quantile rather than the point estimate instantiates pessimism in the face of uncertainty: the lower confidence bound bounds the suboptimality of the delivered node by its own estimation error and keeps noise-inflated values at thinly evidenced nodes from being selected (Jin et al., 2021; Yang et al., 2023). It is the delivery-side counterpart of optimistic sampling for exploration (Auer et al., 2002), both acting on the same posteriors, optimistically to acquire information (Section 3.4) and pessimistically to avoid risk. The rule generalizes "penalize small N when means are tied" to "penalize small N whenever the difference is statistically unresolvable": the 1% quantile of a $3/3$ node (mean 1.0) lies far below that of a $150/160$ node (mean 0.9375). Because selection ranges over all historical nodes rather than the end of a chain, noise can delay confirmation but never entrench a direction permanently.

4 Experiments

4.1 Setup

Tasks. Five tasks cover mathematical reasoning, search QA, knowledge QA, scientific reasoning and spreadsheet manipulation: OlympiadBench (Oly; He et al., 2024), SearchQA (SQA; Dunn et al., 2017), TriviaQA (TQA; Joshi et al., 2017), GPQA-Diamond (GPQA; Rein et al., 2024) and SpreadsheetBench (SSB; Ma et al., 2024). Per-task values of the pool size U, test-set size, epochs E, batch size s, total rounds T, effective budget B_{eff} and δ are listed in Appendix D. The test set is always independent; SkillOpt and GEPA keep their original data splits while VeriTree trains on the question pool (Section 3.1).

Model configurations (optimizer → target). Three configurations cover different strength pairings and target scales: (1) DeepSeek-V4-Flash(DeepSeek-AI, 2026) → MiMo-v2.5(Xiaomi MiMo Team, 2026); (2) GLM-5.3GLM-5 Team (2026) → Qwen3.6-35B-A3B(Qwen Team, 2026b); (3) GLM-5.3 → Qwen3.5-4B(Qwen Team, 2026a).

Baselines. S0 is the same initial skill without any training and is the zero point of all Δ values; SkillOpt is the original implementation with full-validation gating; GEPA is reflective prompt evolution with a Pareto outer loop (Agrawal et al., 2026).

Metrics and accounting. We report test accuracy, the change Δ relative to S0 in pp, and training cost. Each accuracy is the mean of three independent re-evaluations of the same delivered skill, which averages out decoding randomness; training is a single run per task with seed 42, so training-side randomness is not covered (Section 5). Cost is the RMB value of a method's training tokens, split by role into target-model rollouts and validations against optimizer analysis and proposals and priced at list prices (Appendix F). All methods share this accounting and exclude the test evaluation, so Δ and cost are comparable within a configuration but not across configurations, where model pairs and prices differ.

Implementation. δ is set per task from the pool size and the base accuracy by the calibration of Appendix E, giving 0.20 on Oly, 0.15 on SQA and TQA, and 0.30 on GPQA and SSB. The final-selection quantile is fixed at $\varepsilon = 1$, the maximum number of completion tokens is 8192 across all methods and tasks, and all methods share the same proposer and evaluation harness.

Table 1: Main results. Each accuracy is the mean of three independent re-evaluations of the same delivered skill and Δ is its change relative to S0 in pp; cost is the RMB value of a method's training tokens under identical role-wise accounting (Appendix F), lower being better. Bold marks the best accuracy and the lowest cost per task column, with all tied entries bold. Where GEPA's final candidate differs from its Pareto best, the table follows the final-candidate convention. Comparisons are valid only within a configuration.

	Oly		SQA		TQA		GPQA		SSB	
Method	acc (Δ)	Cost	acc (Δ)	Cost	acc (Δ)	Cost	acc (Δ)	Cost	acc (Δ)	Cost
Configuration 1: DeepSeek-V4-Flash → MiMo-v2.5										
S0	0.4453	–	0.7336	–	0.8171	–	0.8182	–	0.3464	–
SkillOpt	0.4656 (+2.03)	72.52	0.8193 (+8.57)	68.01	0.8271 (+1.00)	34.07	0.7677 (−5.05)	40.65	**0.5857** (+23.93)	23.71
GEPA	0.4547 (+0.94)	29.78	**0.8321** (+9.85)	20.33	**0.8314** (+1.43)	9.56	0.8232 (+0.50)	29.84	0.4214 (+7.50)	3.56
VeriTree	**0.4734** (+2.81)	**16.65**	0.8071 (+7.35)	**13.75**	0.8214 (+0.43)	**6.72**	**0.8333** (+1.51)	**11.68**	0.4821 (+13.57)	**2.72**
Configuration 2: GLM-5.3 → Qwen3.6-35B-A3B										
S0	0.5109	–	0.7414	–	0.6679	–	0.7525	–	0.4071	–
SkillOpt	0.5781 (+6.72)	518.74	0.8086 (+6.72)	100.06	**0.7271** (+5.92)	263.63	0.7929 (+4.04)	157.63	0.4250 (+1.79)	244.76
GEPA	0.5547 (+4.38)	346.07	0.8100 (+6.86)	47.87	**0.7271** (+5.92)	69.37	**0.8030** (+5.05)	89.63	**0.5286** (+12.15)	44.07
VeriTree	**0.6062** (+9.53)	**303.16**	**0.8193** (+7.79)	**39.28**	0.7168 (+4.89)	**59.85**	**0.8030** (+5.05)	**71.38**	0.5177 (+11.06)	**24.49**
Configuration 3: GLM-5.3 → Qwen3.5-4B										
S0	0.4891	–	0.6507	–	0.4286	–	0.6667	–	0.1464	–
SkillOpt	0.5250 (+3.59)	128.70	0.7429 (+9.22)	109.46	**0.5114** (+8.28)	93.38	0.6818 (+1.51)	27.67	0.1750 (+2.86)	29.03
GEPA	**0.5406** (+5.15)	31.43	**0.7564** (+10.57)	12.15	0.4986 (+7.00)	14.49	**0.7172** (+5.05)	19.82	0.2000 (+5.36)	3.41
VeriTree	0.5190 (+2.99)	**25.59**	0.7500 (+9.93)	**11.53**	0.4986 (+7.00)	**13.32**	0.6818 (+1.51)	**17.48**	**0.2464** (+10.00)	**3.37**

4.2 Main Results

Table 1 reports three configuration blocks with rows {S0, SkillOpt, GEPA, VeriTree} and, per task, the accuracy with its change Δ and the training cost; the token-level breakdown of every cell is in Appendix F.

Result 1: comparable or better accuracy, with positive gains retained on the task where SkillOpt regresses. The most informative cell is GPQA under Configuration 1: SkillOpt reaches $\Delta = -5.05$ pp (0.7677, below S0's 0.8182), GEPA $+0.50$ pp, and VeriTree $+1.51$ pp, the best of the column. Section 2 explains the regression: under noisy measurement and a monotone threshold the accept decision is sensitive to validation-set fluctuation, whereas VeriTree's final selection requires the delivered node to carry sufficient evidence and its separation criterion does not declare leadership lightly. Averaged over the 15 cells, Δ is $+6.36$ pp for VeriTree against $+5.85$ for GEPA and $+5.41$ for SkillOpt, and the only negative cell belongs to SkillOpt. Where VeriTree does not lead, the shortfall is small and the cost gap large: on SSB under Configuration 1 SkillOpt is best at $+23.93$ pp but costs 23.71 RMB, 8.7 times VeriTree's 2.72, while VeriTree's $+13.57$ pp is second and 1.8 times GEPA's $+7.50$ pp; on SSB under Configuration 2 it trails GEPA by 1.1 pp. It leads on Oly under Configuration 2 ($+9.53$ against $+6.72$ and $+4.38$ pp) and on SSB under Configuration 3.

Result 2: training cost drops substantially under identical accounting. Across all 15 configuration×task cells VeriTree attains the lowest cost, at 10%–63% of SkillOpt's with a median of 20%, and 39%–99% of GEPA's with a median of 81%. One representative cell per configuration: Oly under Configuration 1, 16.65 RMB against SkillOpt's 72.52 and GEPA's 29.78; SSB under Configuration 2, 24.49 against 244.76 and 44.07; SQA under Configuration 3, 11.53 against 109.46 and 12.15. The ordering also holds in raw training tokens, where VeriTree is below both baselines in all 15 cells (14%–78% of SkillOpt's, 35%–92% of GEPA's; Appendix F), so the conclusion does not rest on the price table. Relative to SkillOpt the cost reduction exceeds the token reduction in 13 of 15 cells (median 20% against 36%), since cost prices optimizer calls above target-model rollouts and gating those calls on evidence removes exactly the most expensive rounds; relative to GEPA, whose optimizer share is already small, the two reductions are comparable. As all methods share the proposer, harness and data, the differences come from the outer-loop scheduling.

Result 3: on high-baseline tasks, all methods' improvements are small. On TQA under Configuration 1 (S0 = 0.8171) the three methods reach $+1.00$, $+1.43$ and $+0.43$ pp, none above 1.5 pp, the remaining headroom being comparable to the validation noise of 2–5 pp. VeriTree behaves as designed here: Arm 2 triggers earlier, so budget flows to expansion instead of forcing gaps the measurement cannot resolve.

Table 2: δ ablation on Oly (Configuration 1). K denotes the number of nodes in the final tree, counting $\mathcal{S}_0$. "Val:Exp" is the ratio of validation rounds to expansion rounds, where an expansion round is one in which the criterion selected expansion; rounds whose proposal failed and fell back to validation are counted as expansion rounds but produce no node, so K is smaller than one plus the expansion count. "Del. N" is the number of evaluated questions of the delivered node. Tokens are the total training tokens of the run, target-model and optimizer-model combined; the round budget $T = 42$ is identical across the three settings, so the differences come from the per-call token lengths rather than from the number of rounds.

δ	Test acc	Δ (pp)	Tokens	K	Val:Exp	Del. N	Arm 1 / Arm 2
0.10 (tighter)	0.4688	$+2.35$	8.67M	18	23:19	128	5 / 14
0.20 (default)	0.4734	$+2.81$	8.22M	12	29:13	224	9 / 4
0.30 (looser)	0.4609	$+1.56$	7.95M	15	26:16	96	13 / 3

Result 4: the three diagnosed mechanisms are visible in the accounting. For R1, a fixed full validation on every candidate achieves its accuracy that a state-dependent schedule reaches with far less measurement: on Oly under Configuration 1, VeriTree consumes 8.22M training tokens against SkillOpt's 30.06M, 27%, while reaching the larger gain ($+2.81$ against $+2.03$ pp), and since proposer, harness and data are shared the difference is attributable to the schedule alone. For R2, a schedule reading the posterior state yields a ratio that moves with that state: the realized validation-to-expansion ratio is 1.2:1, 2.2:1 and 1.6:1 at $\delta = 0.10$, 0.20 and 0.30, with the two arms firing in opposite directions across those settings (Table 2). For R3, separating material from measurement wastes the expensive role: SkillOpt spends a median of 21.0% of its training tokens on the optimizer (6.0%–44.7%) against 5.2% for VeriTree (2.1%–19.0%; Appendix F), since a VeriTree validation batch both updates a posterior and stays available as proposal material.

4.3 Ablation Study

The configuration is Oly under Configuration 1 of the main table (pool 448, test 640, $s = 32$, $E = 3$, $T = 42$), with validation-target selection, parent sampling and final selection frozen; only δ varies, at the default 0.20, one step tighter (0.10) and one step looser (0.30).

The scheduling behavior moves as the criterion predicts (Table 2). Tightening to $\delta = 0.10$ raises $z_{1-\delta}$ from 0.84 to 1.28 and the competition cost by roughly $2.3\times$ (Table 4), so Arm 2 concedes infeasibility 14 times instead of 4, the tree widens to $K = 18$ nodes and the delivered node's evidence thins to $N = 128$. Loosening to $\delta = 0.30$ lets noise-magnitude gaps satisfy Arm 1, which fires 13 times instead of 9, so validation stops earlier (26 rounds against 29) and the delivered node rests on only $N = 96$. The two arms therefore respond to δ in opposite directions, as Eqs. 3 and 4 imply, and the calibrated default is the setting that yields the deepest evidence for the delivered node. Accuracy tracks evidence depth, the default being highest at $+2.81$ pp against $+2.35$ and $+1.56$, but those differences sit within the re-evaluation noise of the test set and we do not read them as a ranking; the scheduling counts, whose spread far exceeds any measurement noise, carry the conclusion.

5 Conclusion

This paper restructures validation in the skill self-evolution loop, moving it from a fixed full-scale evaluation at every step, whose single noisy reading becomes a permanent threshold, to a scarce budget scheduled by posterior state. VeriTree organizes candidates as a tree with Beta posteriors, switches between expansion and validation by the δ posterior-separation two-arm criterion, samples validation targets by Thompson sampling, gates optimizer calls on data, and delivers by the lower posterior quantile, with δ determined from the task configuration (Appendix E). Across three optimizer–target configurations and five tasks, VeriTree attains comparable or better accuracy at 10%–63% of SkillOpt's training cost (median 20%), with the δ ablation confirming that the scheduling behavior follows the calibration. In this view, validation is not removed from the loop; it becomes a scarce budget scheduled by posterior state.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation from the authors' Chinese draft into English, and to edit the manuscript for readability, including grammar, wording and typographical corrections. We have not used generative AI tools to generate synthetic data, to develop the theoretical model or conceptual framework, to formulate or prove mathematical claims, to propose or refine hypotheses, to design the research methodology or experiments, to implement the method, to clean or reformat datasets, or to interpret results; the remaining tasks with required disclosure are not applicable to this work. We have reviewed all AI-assisted text: every translated passage was checked against the original draft by the authors, and all numbers, claims and technical statements were independently verified against the experiment logs. We take full responsibility for the final content of this work.

REFERENCES

Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi. PromptWizard: Task-aware agent-driven prompt optimization framework, 2024. arXiv:2405.18369.

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 (ICLR)*, 2026. arXiv:2507.19457.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems, 2026. arXiv:2603.02766.

Anthropic. Agent skills, 2025. Technical documentation.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3):235–256, 2002.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. arXiv:2606.19348.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3369–3391, 2022.

Matthew Dunn et al. SearchQA: A new Q&A dataset augmented with context from a search engine, 2017. arXiv:1704.05179.

Shie Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.

Chrisantha Fernando et al. PromptBreeder: Self-referential self-improvement via prompt evolution. In *International Conference on Learning Representations (ICLR)*, 2024.

Joseph L. Fleiss, Bruce Levin, and Myunghee Cho Paik. *Statistical Methods for Rates and Proportions*. Wiley, 3rd edition, 2003.

Ping Gallo, Lu Mao, and Vincent H. Shih. Alternative views on setting clinical trial futility criteria. *Journal of Biopharmaceutical Statistics*, 24(5):976–993, 2014.

Aurélien Garivier and Emilie Kaufmann. Optimal best-arm identification with fixed confidence. In *Proceedings of the 29th Conference on Learning Theory (COLT)*, 2016.

GLM-5 Team. GLM-5: From vibe coding to agentic engineering, 2026. arXiv:2602.15763.

Qingyan Guo et al. EvoPrompt: Connecting large language models with evolutionary algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pp. 3828–3850, 2024. arXiv:2402.14008.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2408.08435.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline RL? In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 5084–5096, 2021.

Mandar Joshi et al. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best-arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, 17(1):1–42, 2016.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.03714.

John M. Lachin. A review of methods for futility stopping based on conditional power. *Statistics in Medicine*, 24(18):2747–2764, 2005.

Liang Ma et al. SpreadsheetBench: Introducing spreadsheet tasks to evaluate LLMs' multi-step reasoning, 2024. arXiv:2407.09013.

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. arXiv:2603.25158.

Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9340–9366, 2024. arXiv:2406.11695.

Archiki Prasad et al. GrIPS: Gradient-free, edit-based instruction search for prompt optimization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Reid Pryzant et al. Automatic prompt optimization with gradient descent and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Qwen Team. Qwen3.5-4B. Model card, `https://huggingface.co/Qwen/Qwen3.5-4B`, 2026a. Accessed 2026-09-26.

Qwen Team. Qwen3.6-35B-A3B. Model card, `https://huggingface.co/Qwen/Qwen3.6-35B-A3B`, 2026b. Accessed 2026-09-26.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2311.12022.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A self-improving coding agent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2504.15228.

Daniel Russo. Simple bayesian algorithms for best-arm identification. *Operations Research*, 68(6): 1625–1647, 2020.

Xuedong Shang, Rianne de Heide, Pierre Ménard, Emilie Kaufmann, and Michal Valko. Fixed-confidence guarantees for Bayesian best-arm identification. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Zhihong Shao et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. arXiv:2402.03300.

Taylor Shin et al. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Noah Shinn et al. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2303.11366.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

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. *Transactions on Machine Learning Research (TMLR)*, 2024. arXiv:2305.16291.

Wenyi Wang, Piotr Piekos, Li Nanbo, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-Gödel machine: Human-level coding agent development by an approximation of the optimal self-improving machine. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.21614.

James M. S. Wason and Lorenzo Trippa. A comparison of Bayesian adaptive randomization and multi-stage designs for multi-arm clinical trials. *Statistics in Medicine*, 33(13):2206–2221, 2014.

Xiaomi MiMo Team. MiMo-V2.5. Model card, `https://huggingface.co/XiaomiMiMo/MiMo-V2.5`, 2026. Accessed 2026-09-26.

Chao-Han Huck Yang, Zhengling Qi, Yifan Cui, and Pin-Yu Chen. Pessimistic model selection for offline deep reinforcement learning. In *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2023.

Chengrun Yang et al. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

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, 2026. arXiv:2605.23904.

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(8055):609–616, 2025.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2505.22954.

Andrew Zhao et al. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. arXiv:2308.10144.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2211.01910.

A Related Work

Iterative prompt and instruction optimization. Automatic prompt optimization first appeared in search form: APE (Zhou et al., 2023) has an LLM propose candidate instructions, scores them on held-out samples, and takes the best; OPRO (Yang et al., 2024) makes the LLM itself the optimizer, carrying an instruction–score history in the meta-prompt for natural-language hill climbing. Feedback-based editing methods convert execution errors into modification grounds: ProTeGi (Pryzant et al., 2023) induces textual gradients on mini-batches and retains beam candidates by validation performance; TextGrad (Yuksekgonul et al., 2025) back-propagates textual critiques along the LLM call graph; GrIPS (Prasad et al., 2023) performs greedy search over a sentence-level edit space; PromptWizard (Agarwal et al., 2024) jointly optimizes instructions and in-context examples. There are also gradient-based discrete search (Shin et al., 2020) and reinforcement-learning approaches (Deng et al., 2022). This family established the principle that edits must be screened by validation; but its validation is generally fixed full-scale scoring with point-value comparison, candidates share the same acceptance rule, acceptance decisions do not adjust with estimation quality, and the effect of measurement noise on selection is rarely discussed. The diagnosis of Section 2 targets exactly this common implementation convention.

Evolutionary and skill-level self-evolution. Evolutionary methods organize search through candidate pools and fitness-based elimination: EvoPrompt (Guo et al., 2024) uses the LLM as an evolutionary operator; PromptBreeder (Fernando et al., 2024) co-evolves task prompts and mutation prompts; MIPROv2 (Khattab et al., 2024; Opsahl-Ong et al., 2024) jointly tunes instructions and demonstrations with Bayesian optimization and selects on the dev set; GEPA (Agrawal et al., 2026) maintains candidates that each excel on different subsets via a Pareto front with reflective mutation, and is one of our experimental baselines. At the skill level, Reflexion (Shinn et al., 2023) turns a failed episode into a verbal lesson kept in an episodic buffer for the next attempt, without any persistent artifact across tasks; Voyager (Wang et al., 2024) maintains a growing skill library in a game environment; ExpeL (Zhao et al., 2024) distills transferable knowledge from experience; EvoSkill (Alzubi et al., 2026) evolves skill folders through a frontier set, failure-driven proposals, and validation-set admission; Trace2Skill (Ni et al., 2026) analyzes a large batch of trajectories in parallel and distills them into a single skill, avoiding the order dependence of sequential editing. SkillOpt (Yang et al., 2026) provides the most complete training discipline and is our direct baseline. Across this family, the outer loop is always rollout–analysis–validation, with differences concentrated in proposal format and acceptance rules; how the validation budget should vary with estimation quality has not been systematically studied.

Tree-search self-improvement and uncertainty modeling. DGM (Zhang et al., 2026) and SICA (Robeyns et al., 2025) organize self-improving coding agents as archive-based evolution; ADAS (Hu et al., 2025) searches the agent design space with a meta-agent programmer; HGM (Wang et al., 2026) further formalizes self-improvement as tree search, decomposing each decision into three sub-policies, namely choosing the action, choosing the parent, and choosing the node to evaluate, and estimates lineage improvement potential through clade-level aggregation. The tree organization and action decomposition of this paper adopt the same decision decomposition as HGM; the two differ essentially: HGM's expansion condition is an open-loop counting power law $N^{\alpha} \geq |T|$ and its validation selection uses temperature annealing, whereas our expansion condition reads posterior separation and budget feasibility, and our validation selection is temperature-free Thompson sampling. The goals also differ: HGM estimates how representative the current score is of long-run improvement potential, while we solve the allocation of validation tokens. Using the Beta conjugate posterior for Bernoulli observations and sampling from the posterior to balance exploration and exploitation is a classical uncertainty-sampling technique (Thompson, 1933; Chapelle & Li, 2011), in the same family as upper-confidence bounds (Auer et al., 2002) and best-arm identification (Even-Dar et al., 2006; Kaufmann et al., 2016) in multi-armed bandits; we apply it to validation-target selection and let the switching criterion and the sampler share the same family of beliefs.

B Complete Pseudocode

Algorithm 1 VeriTree main loop

Require: question-pool size U (an environmental constant); number of epochs E; batch size s; final-selection quantile $\varepsilon = 1$ (fixed); switching parameter δ (Appendix E); the data pool cycles per epoch, one batch of s questions at a time

$B_q^{\text{total}} \leftarrow U \cdot E$; $b_q \leftarrow B_q^{\text{total}}$; $T \leftarrow B_q^{\text{total}}/s$
nodes $\leftarrow \{S_0\}$ $\{k = N = v = e = 0$, trajectories $= []\}$
completed $\leftarrow 0$
$\{$Round 1: forced validation of $S_0\}$
results $\leftarrow$ TARGETROLLOUT(S_0.skill, NEXTBATCH()); UPDATEPOSTERIOR(S_0, results)
$b_q \leftarrow b_q - s$; completed $\leftarrow 1$
while completed $< T$ **and** $b_q \geq s$ **do**
  if SHOULDEXPAND(δ, b_q) $\{$Algorithm 2$\}$ **then**
    ok $\leftarrow$ DOEXPAND() $\{$Algorithm 3$\}$
    if not ok **then**
      DOVALIDATE() $\{$Algorithm 3$\}$
    end if
  else
    DOVALIDATE()
  end if
  completed $\leftarrow$ completed $+ 1$
end while
delivered $\leftarrow \arg\max_{a:\, N_a>0} \text{Beta.ppf}(\varepsilon/100,\ 1{+}a.k,\ 1{+}a.N{-}a.k)$ $\{$lower-quantile delivery, Eq. 7$\}$
return INDEPENDENTTEST(delivered.skill)

Algorithm 2 SHOULDEXPAND(δ, b_q): the two-arm switching criterion

alive $\leftarrow \{a \in$ nodes $: a.N > 0\}$
if |alive| < 2 **then**
  return completed ≥ 1 $\{$cold start: the tree can grow after S0's first validation$\}$
end if
for $a \in$ alive **do**
  $P_{\text{top}}[a] \leftarrow \int_0^1 f_a(x) \prod_{b \neq a} F_b(x)\, dx$ $\{$one-dimensional quadrature, Eq. 3$\}$
end for
if $\max_a P_{\text{top}}[a] \geq 1 - \delta$ **then**
  return true$\{$Arm 1: separation attained$\}$
end if
$L, C \leftarrow$ top two nodes by posterior mean $\hat{\mu}$ $\{$leader and strongest competitor$\}$
$\hat{\Delta} \leftarrow \hat{\mu}_L - \hat{\mu}_C$
if $\hat{\Delta} \leq 0$ **then**
  return true$\{$tie: separation requires unbounded questions$\}$
end if
$z \leftarrow \Phi^{-1}(1 - \delta)$
$n^* \leftarrow z^2[\hat{\mu}_L(1 - \hat{\mu}_L) + \hat{\mu}_C(1 - \hat{\mu}_C)]/\hat{\Delta}^2$
need $\leftarrow \max(n^* - L.N, 0) + \max(n^* - C.N, 0)$
return need $> b_q$ $\{$Arm 2: infeasible within the remaining budget$\}$

C FORMULA LIST (IN ORDER OF APPEARANCE)

1. Selection bias: $\mathbb{E}[\text{recorded} \mid \text{accepted}] > \mathbb{E}[\text{true}]$
2. Posterior: $\theta_a \sim \text{Beta}(1 + k_a,\ 1 + N_a - k_a)$
3. Posterior mean: $\hat{\mu}_a = (1 + k_a)/(2 + N_a)$
4. Best-in-tree probability: $P_{\text{top}}(a) = \int_0^1 f_a(x) \prod_{b \neq a} F_b(x)\, dx$
5. Arm 1: $\max_a P_{\text{top}}(a) \geq 1 - \delta \Rightarrow$ expand

Algorithm 3 DoValidate, DoExpand, and UpdatePosterior

procedure DoValidate()
  pool $\leftarrow \{a \in \text{nodes} : a.N > 0\}$
  target $\leftarrow \arg\max_{a \in \text{pool}}$ sample $\sim \text{Beta}(1 + a.k,\ 1 + a.N - a.k)$ {Thompson sampling, Eq. 5}
  results $\leftarrow$ TargetRollout(target.skill, NextBatch())
  UpdatePosterior(target, results); $b_q \leftarrow b_q - s$
procedure DoExpand() $\rightarrow$ bool
  eligible $\leftarrow \{a : a.v > a.e \wedge \hat{\mu}_a > 0\}$ {data gating}
if eligible is empty **then**
  return false
end if
  parent $\leftarrow$ weighted sample from eligible, weights $\hat{\mu} = (1 + k)/(2 + N)$ {Eq. 6}
  editBasis $\leftarrow$ parent.trajectories[parent.e] {edit-basis rotation: earliest unused batch}
  childSkill $\leftarrow$ ReflectMergeRankApply(parent.skill, editBasis)
if childSkill is None **then**
  return false{failure does not consume the parent's material eligibility}
end if
  add child node to nodes; parent.$e \leftarrow$ parent.$e + 1$
  results $\leftarrow$ TargetRollout(child.skill, NextBatch()) {the child's first validation}
  UpdatePosterior(child, results); $b_q \leftarrow b_q - s$
return true
procedure UpdatePosterior(a, results)
  $a.k \leftarrow a.k +$ NumCorrect(results); $a.N \leftarrow a.N + |\text{results}|$; $a.v \leftarrow a.v + 1$
  a.trajectories.append(results)

6. $n^* = z^2_{1-\delta}\,[\hat{\mu}_L(1 - \hat{\mu}_L) + \hat{\mu}_C(1 - \hat{\mu}_C)]/\hat{\Delta}^2$
7. Arm 2: $[(n^* - N_L)^+ + (n^* - N_C)^+] > b_q \Rightarrow$ expand
8. Validation sampling: $S_a \sim \text{Beta}(1 + k_a,\ 1 + N_a - k_a)$
9. Parent draw: $P(a) = \hat{\mu}_a / \sum_{b \in \mathcal{E}} \hat{\mu}_b$
10. Final selection: $a^* = \arg\max_a F^{-1}_{\text{Beta}}(\varepsilon/100)$

D Task and Budget Settings

Table 3 lists the per-task settings used in all experiments. The test set is independent of the question pool in every task. The per-epoch round counts are 14, 15, 15, 8, and 3 for Oly, SQA, TQA, GPQA, and SSB, respectively.

Table 3: Tasks and budget settings. U denotes the question-pool size, E the number of epochs, s the batch size, $T = B_q/s$ the total number of training rounds, B_{eff} the effective budget (Eq. 8), and δ the confidence parameter calibrated by Appendix E.

Task	U	Test	E	s	T	B_{eff}	δ
Oly	448	640	3	32	42	896	0.20
SQA	600	1400	4	40	60	1200	0.15
TQA	600	1400	4	40	60	1200	0.15
GPQA	136	198	4	17	32	272	0.30
SSB	120	280	4	40	12	240	0.30

Validation splits and the two diagnostic measurements of Section 2. The gate-resolution figure $1/|D_{\text{val}}|$ quoted in Section 2 uses the validation splits of the chain-style baselines, which contain 128 questions on Oly, 200 on SQA and TQA, and 40 on GPQA and SSB, giving resolutions of 0.78, 0.50 and 2.50 pp respectively. The re-evaluation range is obtained by evaluating the same delivered $\mathcal{S}_0$ five times in parallel on one validation batch and taking the difference between the highest and

lowest readings; on Oly this difference is 5.47 pp, and across the five tasks it falls in 2–5 pp. The late-training gap of 1–2 pp is measured on the same Oly configuration by running SkillOpt five times to completion and, within each run, comparing every pair of adjacent versions over the last three steps on the full validation set, which yields ten adjacent pairs in total.

E Calibration of δ: Method and Numerical Tables

E.1 Calibration method. Let Δ_{useful} denote the smallest gap worth resolving, set to the typical re-evaluation range of S0 across the five tasks, which falls in 2–5 pp and is taken as 4 pp so that a single calibration is shared by all tasks; let φ denote the fraction of the effective budget allocated to its symmetric two-sided competition cost, set to $1/2$, reserving half of the effective budget for expansion and first validations. Solving,

$$z^* = \Delta_{\text{useful}} \sqrt{\frac{\varphi\, B_{\text{eff}}}{4\,\hat{p}(1-\hat{p})}}, \qquad \delta^* = 1 - \Phi(z^*), \qquad B_{\text{eff}} = \min(U \cdot E,\ 2U), \tag{8}$$

where U is the question-pool size and $2U$ is the pool evidence cap: the independent evidence a single node can accumulate about a given pool is bounded by U, and the competition budget of a pair of nodes is capped accordingly. $\hat{p}$ is backfilled after S0's first validation batch; when unknown it is set to 0.5, where the variance is maximal and δ is loosest. Substituting Oly (Configuration 1): $U = 448$, $E = 3$, $B_{\text{eff}} = \min(1344, 896) = 896$; $\Delta_{\text{useful}} = 4$ pp, $\varphi = 1/2$, $\hat{p} \approx 0.5$, giving $z^* \approx 0.85$ and $\delta^* \approx 0.20$.

E.2 δ–z–relative cost. The competition cost $c(\Delta, \delta) = 4z(\delta)^2\hat{p}(1-\hat{p})/\Delta^2$ (symmetric, two-sided, both nodes topped up to n^*) is quadratically sensitive to z:

Table 4: Relative competition cost and behavior as a function of δ.

δ	z	Relative cost	Behavior
0.001	3.09	3.5×	Too strict: no competition is feasible; expansion only
0.01	2.33	2.0×	Strict: small-gap competitions declared infeasible by Arm 2
0.05	1.65	1.0×	Statistical convention; affordable only with large budgets and high accuracy
0.10	1.28	0.61×	Reasonable range for small and medium budgets
0.20	0.84	0.26×	Reasonable range for small budgets
0.30	0.52	0.10×	Too loose: noise-level gaps trigger separation; wide tree, shallow evidence

E.3 Node-count model. Validation rounds outside competitions no longer occur once separation is attained; the question-count ledger is

$$B_q = \underbrace{s}_{\text{S0 first validation}} + \underbrace{X \cdot s}_{\text{expansion rounds}} + \underbrace{f \cdot X \cdot c_s}_{\text{amortized competitions}}, \qquad c_s = \varphi \cdot 2U, \tag{9}$$

where f is the fraction of children falling within Δ_{useful} of the leader and thereby triggering a competition, anchored at $f \approx 0.10$–0.15 by back-calculation from run logs; solving yields the expected number of expansion rounds $X = (B_q - s)/(s + f\varphi \cdot 2U)$ and the number of nodes $K \approx X + 1$.

E.4 Numerical values for three budget tiers ($\Delta_{\text{useful}} = 4$ pp, $\varphi = 1/2$, $\hat{p} = 0.5$ throughout; $s = 40$ for the first two tiers and $s = 32$ for the third, which reproduces the actual Oly configuration).

The scaling law and the accuracy-domain correction can be read directly from Table 5: a fivefold increase in budget (120×4 → 600×4) tightens δ^* from 0.33 to 0.16; raising $\hat{p}$ from 0.5 to 0.9 tightens δ^* by about one order of magnitude under large budgets. The node-count model of Eq. 9 gives the expected tree size in the last two rows. The third tier reproduces the Oly configuration, so its predictions can be checked against the measured run: the predicted validation-to-expansion

Table 5: Calibration values for three budget tiers. $\sqrt{0.25/U}$ is the statistical ceiling on single-node evidence depth.

	120×4	600×4	448×3
B_q / B_{eff}	480 / 240	2400 / 1200	1344 / 896
z^*	0.44	0.98	0.85
δ^* (4 pp criterion)	≈0.33	≈0.16	≈0.20
δ^* (if $\hat{p} = 0.8$)	0.29	0.11	0.14
δ^* (if $\hat{p} = 0.9$)	0.23	0.05	0.08
Ceiling $\sqrt{0.25/U}$ (pp)	±4.6	±2.0	±2.4
Expected K ($f = 0.15$)	≈8	≈19	≈14
Expected validation-to-expansion ratio	1.5	3.2	2.2

ratio of 2.2 matches the 29:13 ratio observed at the calibrated $\delta = 0.20$ (Table 2), and the predicted $K \approx 14$ is close to the 12 nodes actually produced.

F Token-Level Accounting

Table 7 reports, for every cell of the main table, the total training tokens and their split by role: target-model tokens (rollouts and validations) and optimizer-model tokens (analysis and proposals). The costs in Table 1 are computed from this split at the per-token list prices of Table 6, under the same accounting for all methods.

Table 6: Per-token list prices (RMB per million tokens, input/output).

Configuration	Model (role)	Input	Output
1	MiMo-v2.5 (target)	1	2
1	DeepSeek-V4-Flash (optimizer)	3	9
2	Qwen3.6-35B-A3B (target)	1.8	10.8
2	GLM-5.3 (optimizer)	8	28
3	Qwen3.5-4B (target)	0.21	0.21
3	GLM-5.3 (optimizer)	8	28

Table 7: Token-level accounting (millions of tokens). For each task: total training tokens, target-model tokens (Tgt), and optimizer-model tokens (Opt).

	Oly			SQA			TQA			GPQA			SSB		
Method	Total	Tgt	Opt	Total	Tgt	Opt	Total	Tgt	Opt	Total	Tgt	Opt	Total	Tgt	Opt
Configuration 1: DeepSeek-V4-Flash → MiMo-v2.5															
S0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SkillOpt	30.06	25.18	4.88	29.46	21.80	7.66	10.36	5.73	4.63	17.71	16.05	1.66	8.89	5.91	2.98
GEPA	14.46	13.89	0.57	12.55	11.92	0.63	4.17	3.83	0.34	14.44	13.98	0.46	1.77	1.63	0.14
VeriTree	8.22	7.90	0.32	10.18	9.59	0.59	2.37	1.92	0.45	5.03	4.62	0.41	1.23	1.02	0.21
Configuration 2: GLM-5.3 → Qwen3.6-35B-A3B															
S0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SkillOpt	60.49	47.78	12.71	16.88	12.99	3.89	40.22	30.09	10.13	15.80	11.61	4.19	47.39	36.04	11.35
GEPA	44.25	42.28	1.97	17.78	17.41	0.37	17.91	17.45	0.46	14.15	12.88	1.27	8.99	8.37	0.62
VeriTree	30.72	29.16	1.56	12.83	12.16	0.67	15.09	14.76	0.33	11.52	10.50	1.02	7.34	6.75	0.59
Configuration 3: GLM-5.3 → Qwen3.5-4B															
S0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SkillOpt	60.17	48.61	11.56	63.77	53.76	10.01	36.68	29.60	7.08	12.04	9.63	2.41	39.72	37.33	2.39
GEPA	41.89	40.04	1.85	24.83	24.22	0.61	31.41	30.70	0.71	16.02	14.78	1.24	7.93	7.79	0.14
VeriTree	33.48	32.14	1.34	22.91	22.34	0.57	28.69	28.08	0.61	8.49	7.40	1.09	6.63	6.44	0.19

G SCHEDULING BEHAVIOR ANALYSIS

Phase structure and the realized ratio. No structural ratio is enforced between validation and expansion; the realized ratio follows the posterior state. In the competitive phase, in which the leading candidate and its strongest competitor are not yet separated, the ratio of validation rounds to expansion rounds tends toward validation-only, since every round adds evidence to the ranking. In the separated phase, in which the two are separated, the ratio tends toward 1:1, because each expansion is still accompanied by the child's first validation batch. The expected ratio therefore varies with the evidence state within $[1{:}1, \infty)$ rather than being fixed by a counter; the realized ratios are reported in Table 2, where they range from 1.2:1 to 2.2:1 across the three δ settings.

Alternative designs for validation-target selection. Replacing the Thompson sampler with $\arg\max \hat{\mu}$ would let early noise peaks monopolize validation rounds: a node whose first batch is lucky holds both the highest mean and, until the others catch up, the narrowest posterior, and the entrenchment of Section 2 recurs on the validation side. A fixed round-robin over nodes decouples validation from the information bottleneck, spending rounds on nodes that can no longer change the ranking. The weighted parent draw of Eq. 6 has a parallel justification: the weights of high-N nodes have small variance, so low-N nodes are occasionally drawn but do not monopolize.

Open-loop versus closed-loop criteria. Tree-search-style self-improvement decides expansion timing by an open-loop counting rule, expanding when a power law of evaluation counts over the number of nodes, $N^{\alpha} \geq |T|$, is satisfied (Appendix A). Under such rules the tree size is determined by a pacing formula unrelated to whether the existing nodes have been separated, and the exponent has no state-based justification. VeriTree's criterion reads posterior separation and budget feasibility instead, so the tree size is determined by the evidence state rather than by a fixed cadence.