REUSING TRAINABLE STATES, NOT JUST SOLUTIONS: POLICY-ARCHIVE SEARCH AT TEST TIME FOR AUTOMATED DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time discovery can accumulate two coupled histories: the solutions found during search and the trainable policies produced by successive test-time updates. Test-time search with frozen language models reuses discovered solutions, while test-time training additionally updates the model using verifier feedback on newly generated candidates. Yet such methods typically continue training from the most recently updated policy; earlier policies are never again eligible for an update. Solution reuse preserves sampled outcomes, but not the earlier proposal distributions that produced them. Moreover, a policy's proposal behavior depends on the evolving context provided by previously discovered solutions—the Solution Archive. A later Solution Archive can therefore elicit different behavior from a frozen historical policy. Empirically, re-evaluating policies saved at different training stages under the same later Solution Archive reorders their relative performance—earlier states can even outperform the latest one. The starting point of each update is therefore a search decision, not a fixed sequence. We introduce PAST (Policy-Archive Search at Test Time), which extends reuse from solutions to trainable policies. Its Policy Archive allows learning to resume from a historical state under the current Solution Archive, creating a new continuation branch. We instantiate PAST with ProgressEMA, which uses observed capability and delayed parent-to-child progress to choose which archived policy receives the next update. Across three verifier-driven mathematical discovery tasks, PAST–ProgressEMA reaches better frontiers than the Current-only method, which always continues from the latest policy, while maintaining high per-epoch proposal quality even when the cumulative frontier plateaus. These results show that reusing trainable policies, not just solutions, can improve test-time discovery. Code is available at `https://anonymous.4open.science/r/submission-artifact-105E/README.md`.

1 INTRODUCTION

How should test-time discovery reuse both the solutions it finds and the trainable policies produced along the way? Frozen-policy search methods such as FunSearch (Romera-Paredes et al., 2024) and AlphaEvolve (Novikov et al., 2025) reuse discovered solutions as contexts for generating new candidates. Test-time training methods such as ThetaEvolve (Wang et al., 2025) and TTT-Discover (Yuksekgonul et al., 2026) additionally adapt the proposal policy using verifier feedback on newly generated candidates.

A Solution Archive preserves sampled outcomes and keeps them available as future generation contexts, while successive trainable states record how the proposal policy changes through updates. Current approaches typically preserve the first history for reuse while restricting future updates to policies from the latest training stage. In the **Current-only** case, writing p_t for the policy at round t, learning proceeds along $p_0 \to \cdots \to p_t$, and only p_t remains eligible for the next update. We call this serial update path **single-chain continuation**. TTT-Discover follows this pattern; throughout the experiments, **Current-only** denotes our local implementation of this baseline.

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.

An earlier policy can still be useful after the Solution Archive has grown. We can evaluate it with solutions from the later archive without updating its weights. In our Circle Packing 26 (CP26) probe (Appendix E), six policies rank differently under the common final archive than in their original evaluations; for some solution parents, an earlier policy outperforms the latest one (Figure 1).

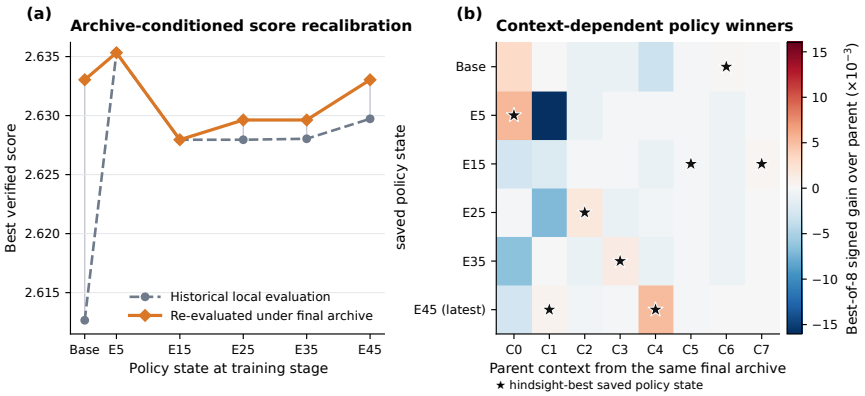


Figure 1: **Historical policies can expose different opportunities under a common later Solution Archive on 26-circle packing (CP26) with Qwen3-8B. (a)** Re-evaluating six saved policies without further updates against the same final archive changes their relative performance compared with their evaluations at creation time (Spearman $\rho = 0.493$). **(b)** Columns C0–C7 denote eight fixed solution parents from that same final Solution Archive; rows denote the six frozen policy states. Each cell reports the best-of-eight signed gain relative to its parent, with positive values indicating improvement. Stars mark the hindsight-best policy for each parent. E45 is the latest policy tested.

We therefore ask: *why should the next update be restricted to the latest policy?* We introduce **PAST (Policy-Archive Search at Test Time)**, which lets the search choose which saved policy to continue training from. Alongside the Solution Archive, PAST maintains a Policy Archive of LoRA adapters and their matching optimizer states. Each update starts from an eligible state selected from this archive. When PAST selects a historical state, it restores that state under the current Solution Archive, collects fresh on-policy rollouts, updates the restored state, and appends the resulting child as a new continuation branch.

Our contributions are:

1. We identify a missing form of reuse in current test-time discovery: methods reuse past solutions but typically do not reuse earlier policies for future updates. We show that these earlier states can still generate useful, complementary proposals when paired with later discovered solutions.
2. We introduce PAST, which organizes saved trainable states in a Policy Archive and turns the choice of continuation state into an explicit search decision. By storing recoverable adapter–optimizer states, PAST can branch future training from selected current or historical policies under the current Solution Archive.
3. We compare three continuation strategies within PAST—ProgressEMA, ScorePUCT, and Uniform—and analyze how each organizes search over the Policy Archive. Across three verifier-driven mathematical discovery tasks, PAST–ProgressEMA reaches better frontiers than Current-only, while matched-rollout, multi-seed, current-batch, and topology analyses characterize the resulting discovery behavior. These results show that reusing trainable policies, not just solutions, can improve test-time discovery.

2 Related Work

Automated discovery with solution archives. FunSearch, ReEvo, AlphaEvolve, and OpenEvolve combine language-model generation, executable verification, and program archives in iterative search loops (Romera-Paredes et al., 2024; Ye et al., 2024; Novikov et al., 2025; Algorithmic Superintelligence, 2025). Subsequent systems improve sample efficiency through diversity-aware parent

selection and candidate generation (Lange et al., 2025; Yang et al., 2026). Across this line, persistent search state lies in the program archive; the proposal model itself is not updated. PAST retains this solution-level memory and additionally archives trainable policy states, allowing later search to revisit both the generation context and the continuation state.

Test-time policy learning. ThetaEvolve integrates test-time reinforcement learning into open-problem search, allowing the proposal policy to learn from experience on the problem (Wang et al., 2025). TTT-Discover argues that this learning should target frontier improvement rather than average reward—within a finite budget, one breakthrough matters more than improving every rollout—and combines a high-reward-tail entropic objective with PUCT solution reuse (Yuksekgonul et al., 2026). Policy of Thoughts recasts reasoning as within-instance online optimization, combining MCTS for structured exploration with GRPO updates to a transient per-instance LoRA adapter (Jiao et al., 2026). These methods internalize execution feedback as policy adaptation at test time, yet once a policy state is created, most proceed from the latest state along a single chain. PAST studies the decision this leaves implicit: once test-time training has produced multiple policy states, the system chooses which one receives the next update.

Diversity and uncertainty in policy training. Existing methods organize test-time policy diversity along different axes. Vector Policy Optimization trains a single policy to emit candidate sets that specialize across scalarizations of vector-valued rewards (Bahlous-Boldi et al., 2026). UG-TTT maintains a same-stage LoRA ensemble, estimates epistemic uncertainty from adapter disagreement, and uses it as an exploration bonus (Riaz et al., 2026). These methods provide lateral exploration over a single policy or a same-stage policy set. PAST instead organizes states from different training stages along time and reopens continuation branches to provide temporal exploration.

3 METHOD

3.1 OVERVIEW

PAST treats the starting state of each test-time policy update as a search decision. We call resuming policy training from a selected state a *continuation*; unlike Current-only training, the selected state need not be the latest one. PAST therefore maintains two coupled archives: a Solution Archive of verified candidates, which records what the search has discovered, and a Policy Archive of adapter–optimizer states, which exposes where policy learning can resume. Updating a selected state appends a new child, building a branching search over trainable continuation states. ProgressEMA is one allocation strategy over this search space. Figure 2 contrasts PAST's policy-archive search with Current-only's single-chain continuation.

3.2 SEARCH STATE

Solution-side state. PAST inherits TTT-Discover's solution-side PUCT reuse mechanism and adaptive entropic training objective unchanged (Yuksekgonul et al., 2026). We formulate test-time training for automated discovery as an iterative loop over candidate generation, executable verification, and policy updates. Let x denote a verifier-evaluated rollout. If x is valid, $R(x)$ is its raw verifier score and $u(x) = \sigma R(x)$ is its oriented score, with $\sigma \in \{+1, -1\}$ selecting the direction in which higher is better (task-specific signs in Appendix F). At round t, the Solution Archive $\mathcal{A}_t$ contains the valid verified candidates retained so far and supplies generation contexts for the next batch. Appendix F gives the full inherited objective.

Policy-side state. Each Policy Archive node v stores a trainable continuation state

$$z(v) = (\theta(v), \omega(v)), \tag{1}$$

where $\theta(v)$ is a trainable LoRA adapter and $\omega(v)$ is the optimizer state matched to its training history. To continue from node v, PAST loads its LoRA adapter $\theta(v)$ and the matching optimizer state $\omega(v)$ before the next update.

PAST maintains a rooted Policy Archive $\mathcal{P}_t = (\mathcal{V}_t, \mathcal{E}_t)$. Each node $v \in \mathcal{V}_t$ stores $z(v)$, its parent identifier, creation order, and complete on-policy batch observations. An edge $(v, w) \in \mathcal{E}_t$ records

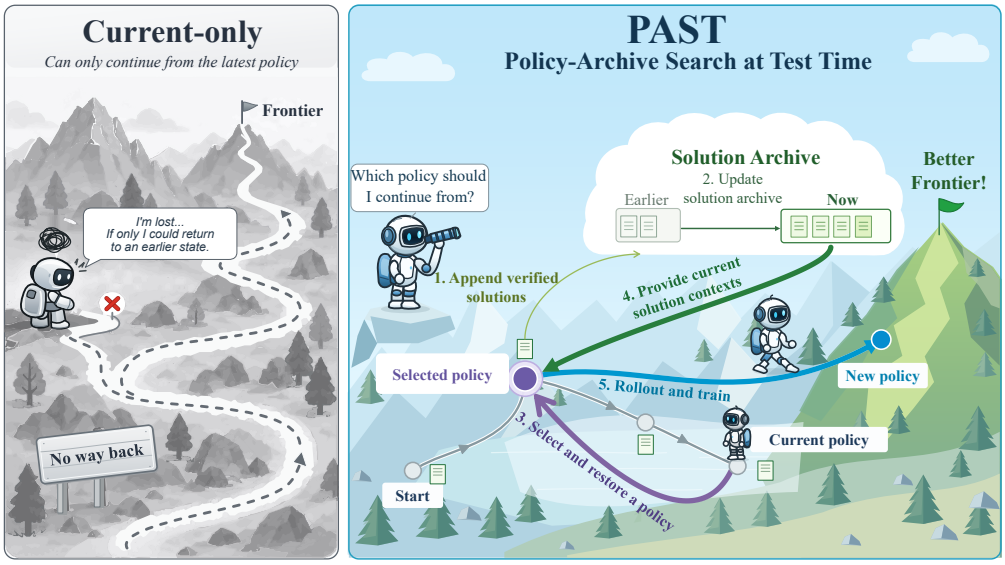


Figure 2: **Policy-archive search at test time.** Current-only can continue only from the latest policy state, leaving earlier states unavailable as continuation points. PAST instead selects a promising state from the Policy Archive, generates and verifies candidates under the current Solution Archive, and appends the resulting update as a new policy branch.

that one update from $z(v)$ produced the child state $z(w)$; delayed progress is measured on this parent–child edge (Section 3.4). A new child enters the archive as `PENDING` and becomes `ACTIVE` after its mandatory direct batch is completed in the next round.

From one chain to continuation allocation. Current-only always continues from the latest trainable state. PAST instead chooses an active policy state from the Policy Archive as the starting point for the next update. Revisiting a node created in an earlier round pairs its state with the current Solution Archive, allowing generation under a later solution context and opening a new continuation branch.

3.3 One-Round Continuation Lifecycle

Algorithm 1 gives the exact within-round chronology underlying the conceptual data flow in Figure 2; Section 3.4 then defines the selector and progress statistic used in Phase 2.

Phase 1: observe and activate the pending child. Because each round performs one policy update, let v_t be the child appended by the previous round; the root v_0 plays this role at initialization. PAST restores $z(v_t)$ and generates a mandatory direct batch D_t^{dir} under the round-start Solution Archive $\mathcal{A}_t$. The batch contains every verifier-evaluated rollout record and its validity/reward outcome; $\mathrm{Valid}(D)$ denotes the valid candidates in batch D. The Solution Archive update is

$$\mathcal{A}_t^{\mathrm{dir}} = \mathrm{Merge}\big(\mathcal{A}_t, \mathrm{Valid}(D_t^{\mathrm{dir}})\big)\,. \tag{2}$$

The node v_t then changes from `PENDING` to `ACTIVE`. If v_t is not the root, its direct batch also completes the delayed observation on edge $(\mathrm{par}(v_t), v_t)$.

Phase 2: allocate the next continuation. Let $\mathcal{P}_t^{\mathrm{dir}}$ be the Policy Archive after recording D_t^{dir} and activating v_t, and let $\mathcal{V}_t^{\mathrm{active}}$ be its active nodes. The selector chooses the training parent

$$\hat{v}_t = \mathcal{S}(\mathcal{P}_t^{\mathrm{dir}}) \in \mathcal{V}_t^{\mathrm{active}}. \tag{3}$$

This selection uses the pending child's first on-policy evidence, the newly completed edge observation, and all earlier Policy Archive statistics. The updated Solution Archive $\mathcal{A}_t^{\mathrm{dir}}$ supplies the rollout context.

Phase 3: reuse or collect a historical-fresh batch. The training batch must be on-policy for $\hat{v}_t$. If $\hat{v}_t = v_t$, PAST reuses D_t^{dir}. Otherwise, it restores $z(\hat{v}_t)$ and generates a historical-fresh batch D_t^{fresh} under $\mathcal{A}_t^{\mathrm{dir}}$. The training batch and end-of-round Solution Archive are

$$(D_t^{\mathrm{train}}, \mathcal{A}_t^{+}) = \begin{cases} (D_t^{\mathrm{dir}}, \mathcal{A}_t^{\mathrm{dir}}), & \hat{v}_t = v_t, \\ (D_t^{\mathrm{fresh}}, \mathrm{Merge}(\mathcal{A}_t^{\mathrm{dir}}, \mathrm{Valid}(D_t^{\mathrm{fresh}}))), & \hat{v}_t \neq v_t. \end{cases} \tag{4}$$

One optimizer update produces

$$z(v_{t+1}) = \mathrm{Update}\big(z(\hat{v}_t), D_t^{\mathrm{train}}\big). \tag{5}$$

PAST appends v_{t+1} as a new `PENDING` child with edge $(\hat{v}_t, v_{t+1})$ and sets $\mathcal{A}_{t+1} = \mathcal{A}_t^{+}$. Solution time therefore advances every round, while the selected parent may come from an earlier round.

Rollout accounting. Every round pays for the mandatory direct batch. Selecting a different historical parent adds one fresh batch:

$$n_t^{\mathrm{roll}} = \begin{cases} |D_t^{\mathrm{dir}}|, & \hat{v}_t = v_t, \\ |D_t^{\mathrm{dir}}| + |D_t^{\mathrm{fresh}}|, & \hat{v}_t \neq v_t. \end{cases} \tag{6}$$

We report direct-rollout and actual-rollout budgets separately. The experiments use 64 rollouts per batch; the algorithm is stated for an arbitrary batch size.

Algorithm 1 PAST: one complete continuation round

Require: Solution Archive $\mathcal{A}_t$; Policy Archive $\mathcal{P}_t$ with pending child v_t; selector $\mathcal{S}$
$D_t^{\mathrm{dir}} \leftarrow \mathrm{RolloutVerify}(z(v_t), \mathcal{A}_t)$ {mandatory direct batch}
$\mathcal{A}_t^{\mathrm{dir}} \leftarrow \mathrm{Merge}(\mathcal{A}_t, \mathrm{Valid}(D_t^{\mathrm{dir}}))$
Record D_t^{dir} for v_t and activate it to form $\mathcal{P}_t^{\mathrm{dir}}$
if $v_t \neq v_0$ **then**
  Complete the progress observation on edge $(\mathrm{par}(v_t), v_t)$; update $m(\mathrm{par}(v_t))$
end if
$\hat{v}_t \leftarrow \mathcal{S}(\mathcal{P}_t^{\mathrm{dir}})$ over active nodes
if $\hat{v}_t = v_t$ **then**
  $D_t^{\mathrm{train}} \leftarrow D_t^{\mathrm{dir}}$; $\mathcal{A}_t^{+} \leftarrow \mathcal{A}_t^{\mathrm{dir}}$
else
  Restore $z(\hat{v}_t)$
  $D_t^{\mathrm{fresh}} \leftarrow \mathrm{RolloutVerify}(z(\hat{v}_t), \mathcal{A}_t^{\mathrm{dir}})$
  Record D_t^{fresh} for $\hat{v}_t$; $D_t^{\mathrm{train}} \leftarrow D_t^{\mathrm{fresh}}$
  $\mathcal{A}_t^{+} \leftarrow \mathrm{Merge}(\mathcal{A}_t^{\mathrm{dir}}, \mathrm{Valid}(D_t^{\mathrm{fresh}}))$
end if
$z(v_{t+1}) \leftarrow \mathrm{Update}(z(\hat{v}_t), D_t^{\mathrm{train}})$
Append pending child v_{t+1} and edge $(\hat{v}_t, v_{t+1})$ to form $\mathcal{P}_{t+1}$
$\mathcal{A}_{t+1} \leftarrow \mathcal{A}_t^{+}$; record n_t^{roll} using Eq. (6)
return $\mathcal{A}_{t+1}, \mathcal{P}_{t+1}, n_t^{\mathrm{roll}}$

3.4 Continuation Allocation

Phase 2 requires the selector $\mathcal{S}$ to choose which active policy state receives the next update. We first define the ideal continuation value underlying this choice and then introduce the online capability and delayed-progress signals used by ProgressEMA.

Continuation value. For any Solution Archive $\mathcal{A}$, let $f(\mathcal{A}) = \max_{x \in \mathcal{A}} u(x)$ denote its cumulative frontier. Let the selection-time history $\mathcal{H}_t^{\mathrm{sel}}$ contain $\mathcal{A}_t^{\mathrm{dir}}$, $\mathcal{P}_t^{\mathrm{dir}}$, and all on-policy batch observations available when Phase 2 begins. Given a continuation rule π and a budget of $b \in \mathbb{N}_+$ optimizer updates, the continuation value of starting from node v is

$$V_b^{\pi}(z(v) \mid \mathcal{H}_t^{\mathrm{sel}}) = \mathbb{E}\big[f(\mathcal{A}_{t+b}^{\mathrm{dir}}) - f(\mathcal{A}_t^{\mathrm{dir}}) \mid \hat{v}_t = v, \mathcal{H}_t^{\mathrm{sel}}, \pi\big]. \tag{7}$$

This value scores a continuation point by its expected frontier improvement over those b updates. The terminal frontier includes the mandatory direct evaluation of the child produced by the bth

update. Because this multi-update value is unavailable online, ProgressEMA uses demonstrated capability as an immediate proxy and delayed parent-to-child progress as an empirical proxy for continuation response.

Delayed continuation progress. For a completed Policy Archive edge $e = (v, w)$, let $q^{\mathrm{train}}(e)$ be the batch-best task reward of the on-policy training batch that created child w, and let $q^{\mathrm{dir}}(w)$ be the batch-best task reward of the child's mandatory direct batch. The realized progress on this edge is

$$\delta(e) = q^{\mathrm{dir}}(w) - q^{\mathrm{train}}(e). \tag{8}$$

Order the completed outgoing edges of node v by observation time as $e_1(v), e_2(v), \ldots$. Its progress estimate is

$$m_1(v) = \delta(e_1(v)), \tag{9}$$

$$m_k(v) = (1-\eta)m_{k-1}(v) + \eta\delta(e_k(v)), \quad k \geq 2. \tag{10}$$

Here $\eta \in (0, 1]$ is the update weight on the newest observation, and $m(v)$ denotes the latest available $m_k(v)$. The child's next-round direct evaluation completes the edge observation, measuring realized parent-to-child progress under the evolving archive context.

Capability and progress signals. Let $s(v)$ be the largest batch-best task reward over all complete on-policy batches observed for active node v, and let $\mathcal{V}_t^{\mathrm{prog}} \subseteq \mathcal{V}_t^{\mathrm{active}}$ contain nodes with at least one completed edge observation. ProgressEMA selects

$$\hat{v}_t = \mathcal{S}_{\mathrm{ProgressEMA}}(\mathcal{P}_t^{\mathrm{dir}}) = \begin{cases} \arg\max_{v \in \mathcal{V}_t^{\mathrm{active}}} s(v), & \text{with probability } \rho, \\ \arg\max_{v \in \mathcal{V}_t^{\mathrm{prog}}} m(v), & \text{with probability } 1-\rho. \end{cases} \tag{11}$$

The capability branch follows the active node with the strongest observed batch, while the progress branch revisits the node whose recent continuations show the largest signed gain. When no progress-eligible node exists, the selector uses the capability branch; ties follow creation order and policy identifier. We use $\rho = 0.7$ and $\eta = 0.95$ for all three tasks. Appendix F gives the ScorePUCT and Uniform definitions and the selector hyperparameters.

4 Experimental Setup

Discovery problems. We evaluate continuation mechanisms on three problems sharing a program-generation, verifier re-execution, and frontier-update interface: **CP26** maximizes $\sum_{i=1}^{26} r_i$ over non-overlapping circles in the unit square; **AC1** minimizes a verifier-recomputed upper bound on the first autocorrelation constant C_1 using nonnegative step-function constructions; and **Erdős minimum-overlap** minimizes the C_5 overlap bound using variable-resolution $[0, 1]$-valued step-function constructions. They span geometric construction, harmonic analysis, and combinatorial number theory. Appendix E gives the exact domains, objectives, and executable verifier rules.

Common protocol. All main experiments use Qwen3-8B, a 32,768-token context, and 50 epochs. Each direct batch generates 8 candidates from each of 8 solution parents, for 64 rollouts; the Solution Archive retains at most 1,000 entries. Current-only and the three PAST variants share the PUCT-style solution-reuse mechanism, training objective, prompt, and adapter–optimizer configuration. Their policy-side allocation strategies determine which state receives each update and whether a historical-fresh batch is required. Appendix F lists all method-specific hyperparameters.

Compared methods. **Current-only** is our local Qwen3-8B implementation of TTT-Discover. It preserves TTT-Discover's solution-side PUCT and adaptive entropic objective while continuing each update from the latest policy state, making it the controlled comparison for latest-only versus archive-based continuation. **UG-TTT** (Riaz et al., 2026) maintains $K = 5$ LoRA members at one training stage. It represents a complementary same-stage ensemble approach and shares the task, base model, and solution-search protocol while retaining its native ensemble-specific adaptation configuration. Within PAST, **PAST–ProgressEMA** allocates updates using observed capability and delayed parent-to-child progress, while **PAST–ScorePUCT** and **PAST–Uniform** test alternative allocation strategies within the same continuation framework.

Evaluation. The primary outcome is the cumulative best verifier score found during search: higher is better for CP26, and lower is better for AC1 and Erdős. We report this frontier under two budget alignments. **Training epoch** aligns the outer-loop update horizon; **cumulative actual rollouts** aligns candidate-generation totals by counting candidates from both direct and historical-fresh batches. Current-only and UG-TTT produce 64 actual rollouts per epoch, while PAST adds a fresh 64-rollout batch whenever it selects a historical parent.

5 Results and Analysis

In this section, we evaluate PAST at both the outcome and process levels. We first compare discovery outcomes under shared update and rollout budgets, then examine how saved policies behave as the Solution Archive evolves and how continuation strategies shape the Policy Archive.

5.1 Discovery Across Update and Rollout Budgets

We begin with the primary 50-update trajectories. Figure 3 follows both the cumulative verified frontier and the best verifier score in each direct batch. The cumulative frontier records what each method has discovered by a given update, while the direct-batch trajectory shows what its current policy proposes as search proceeds.

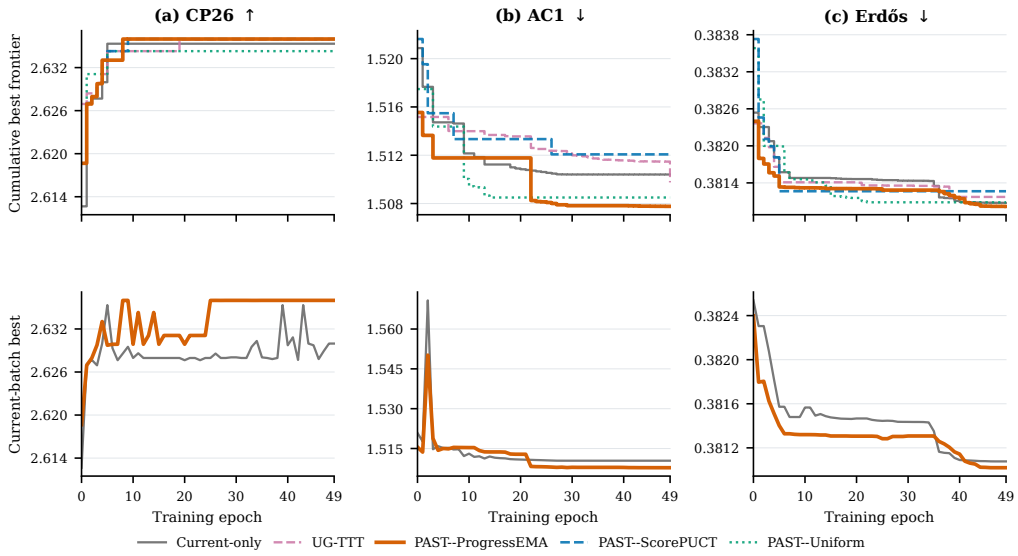


Figure 3: Primary 50-update discovery trajectories. Top: cumulative best verified frontier for Current-only, UG-TTT, and the three PAST selectors. Bottom: best verifier score in the current epoch's direct batch for Current-only and PAST–ProgressEMA. Arrows indicate the favorable direction. The lower row distinguishes current proposal quality from a cumulative frontier that can remain flat after an early discovery.

Discovery unfolds differently across tasks. CP26 plateaus early, whereas AC1 and Erdős continue to improve later in the horizon. On CP26, direct-batch quality continues to vary after the cumulative frontier has stabilized, separating the proposals available for the next update from the best solution already retained. Table 1 reports the final frontiers.

At the shared 50-update horizon, PAST–ProgressEMA reaches a better endpoint than Current-only and UG-TTT on all three tasks. Among the PAST variants, ScorePUCT ties ProgressEMA on CP26 but trails Uniform on AC1 and Erdős; ProgressEMA is the only continuation strategy that is best or tied-best on every task.

Selecting a historical policy can add a fresh rollout batch, so update count and candidate-generation total provide complementary comparisons. When Current-only and UG-TTT are extended to match

Table 1: Verified discovery outcomes for the primary 50-update trajectories. Higher is better for CP26 and lower is better for AC1 and Erdős. Bold marks the best value and underlining marks the second-best value within each task.

Method	CP26 ↑	AC1 ↓	Erdős ↓
Current-only	2.635328	1.510404	<u>0.381077</u>
UG-TTT	<u>2.635896</u>	1.509747	0.381172
PAST–Uniform	2.634292	<u>1.508499</u>	0.381085
PAST–ScorePUCT	**2.635983**	1.512072	0.381265
PAST–ProgressEMA	**2.635983**	**1.507767**	**0.381020**

the task-level actual-rollout total of PAST–ProgressEMA, PAST–ProgressEMA retains better frontiers on AC1 and Erdős, while all three methods tie on CP26 (Appendix A).

To assess sensitivity to sampling seeds, we replicate the primary trajectories across three aligned sampling-seed lanes. PAST–ProgressEMA reaches a better 50-update endpoint than Current-only on every seed for every task (Table 3).

5.2 Historical Policies Under Evolving Solution Archives

The two archives record different aspects of the same search. The Solution Archive records which programs survive verification, while the Policy Archive records where learning can resume. Figure 4 follows these histories through the primary PAST–ProgressEMA trajectories, linking verified frontier advances to newly admitted program families and historical-policy selections.

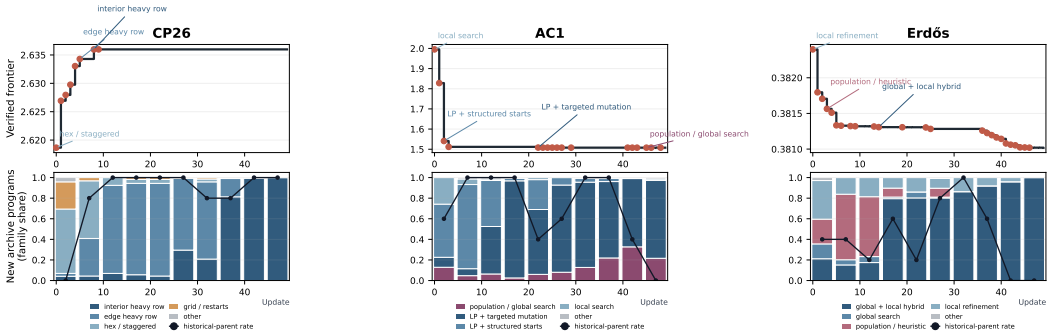


Figure 4: Search evolution in the primary PAST–ProgressEMA trajectories. Top: verified frontier over 50 updates; colored markers denote frontier events, and annotations identify the first frontier event associated with each program family. Bottom: family composition among newly admitted verified programs in five-update bins. The black curve is the fraction of updates in each bin that select a historical policy. The bars count newly admitted verified programs; each family groups related search organizations with multiple initialization and parameterization variants.

The three tasks develop along different program paths. CP26 moves from hexagonal or staggered constructions to edge-heavy and then interior-heavy rows before plateauing. AC1 progresses from local search through structured LP starts and targeted mutation to population-based or global search. Erdős increasingly combines global exploration with local refinement, integrating multiple initialization and parameterization variants within this broader search organization as the frontier improves. Appendix D.1 defines the families and extraction protocol.

The Policy Archive preserves proposal behaviors that remain useful as the Solution Archive evolves. In the controlled CP26 probe, six checkpoints trade places across eight fixed parents under the same later archive: every checkpoint is best for at least one parent, and an earlier checkpoint can

outperform E45, the latest checkpoint tested (Figure 1). Training improves the rate of producing valid programs, while parent-specific strengths remain distributed across training stages. Earlier and later checkpoints therefore provide complementary proposal options under the later archive (Appendix C).

The primary trajectories show how PAST turns these archived options back into active search. Fresh rollouts from historical policies reveal new frontiers on AC1 and Erdős, while children trained from historical parents advance the frontier on all three tasks. On CP26, one such child moves the six-circle row from the boundary to the interior, changing the five-row layout from $(5, 5, 5, 5, 6)$ to $(5, 5, 5, 6, 5)$ and raising the global frontier from 2.634292 to 2.635977. The Policy Archive thus supports both immediate proposal generation and new learning branches (Appendix D.2).

5.3 Continuation Strategies Shape Policy-Archive Structure

Continuation strategy determines how the Policy Archive grows. Uniform repeatedly reopens older states, producing shallow, highly branched genealogies across all three tasks. ScorePUCT primarily extends recent states on AC1 and Erdős, producing long, near-single-chain genealogies. ProgressEMA changes this balance with the task: it revisits broadly on CP26, combines reopening with continued development on AC1, and follows deeper branches on Erdős. Among the three PAST selectors, ProgressEMA is the only continuation strategy that is best or tied-best on every task, while its archive topology varies substantially across tasks. This comparison separates archive capacity from archive use: the same type of saved states supports different search paths under different allocation strategies. Appendix D reports the complete nine-trajectory genealogies and allocation statistics.

6 Discussion and Limitations

Trainable-state history becomes a search resource. Test-time discovery with a trainable proposal policy accumulates two coupled histories: the Solution Archive changes the contexts under which proposals are generated, while the Policy Archive preserves where policy learning can continue. The common-archive probes show that a saved policy's observed proposal quality can change under later solution contexts, and the topology results show how selectors trade off reopening earlier states against extending current branches. Together, these results suggest that continuation allocation should adapt the balance between revisiting earlier states and extending current branches to the evolving search evidence. ProgressEMA provides one lightweight online strategy for this balance; learning the allocation directly from evolving archive contexts remains open.

Archive growth requires long-horizon management. A complete continuation state includes a LoRA adapter and its matching optimizer state. In the current Qwen3-8B setup, each recoverable child state occupies about 2.537 GB, and 50 states occupy about 126.84 GB. Across the two measured trajectories, checkpoint persistence and optimizer-state restoration account for 0.048%–0.062% of recorded execution time. If every child is retained, storage grows linearly with search length. Longer runs therefore require retention, pruning, or compression policies that preserve useful continuation options within an archive budget.

7 Conclusion

PAST makes the starting state of the next policy update an explicit search decision. By retaining historical trainable states and continuing them under the current Solution Archive, it organizes test-time learning beyond a single chain while retaining accumulated discoveries. ProgressEMA provides a lightweight online instance of this framework. Across three discovery tasks, it reaches better frontiers than Current-only and UG-TTT at the shared update horizon; at matched candidate-generation totals, it ties both on CP26 and remains better on AC1 and Erdős. The alternative selectors show that available continuation options must be paired with an effective continuation strategy. These experiments support treating historical trainable states as reusable search resources alongside discovered solutions.

REPRODUCIBILITY STATEMENT

Anonymous source code and reproduction instructions are available at https://anonymous.4open.science/r/submission-artifact-105E/README.md. The repository includes the PAST runtime, Current-only and UG-TTT comparison paths, executable task verifiers, paper configurations, CUDA and Ascend launchers, and unit and runtime tests. Original PAST components are released under the MIT License, with third-party terms documented in the repository.

Executable verifiers recompute every reported task score. The Method and Appendix specify the archive/restore protocol, continuation strategies, model, optimizer, sampling procedure, hyperparameters, and budget accounting. Generation, evaluation, lineage, update, and rollout records underpin the reconstructed trajectories and batch-resolved analyses.

AI USE STATEMENT

We used the generative AI tool OpenAI Codex solely to assist with language editing and polishing of author-written text, including improving clarity, grammar, and readability. We did not use generative AI tools for literature retrieval or identification, research ideation or execution, methodological or experimental design, interpretation of results, or drafting substantive sections of the paper. All technical content, arguments, and initial drafts were produced by the authors. We reviewed all AI-assisted edits and take full responsibility for the final content of this work.

REFERENCES

Algorithmic Superintelligence. OpenEvolve: An open-source evolutionary coding agent. https://github.com/algorithmicsuperintelligence/openevolve, 2025. GitHub repository, accessed 2026-08-09.

Ryan Bahlous-Boldi, Isha Puri, Idan Shenfeld, Akarsh Kumar, Mehul Damani, Sebastian Risi, Omar Khattab, Zhang-Wei Hong, and Pulkit Agrawal. Vector policy optimization: Training for diversity improves test-time search. *arXiv preprint arXiv:2605.22817*, 2026. URL https://arxiv.org/abs/2605.22817.

Herbie Bradley, Andrew Dai, Hannah Teufel, Jenny Zhang, Koen Oostermeijer, Marco Bellagente, Jeff Clune, Kenneth Stanley, Grégory Schott, and Joel Lehman. Quality-diversity through AI feedback. *arXiv preprint arXiv:2310.13032*, 2023. URL https://arxiv.org/abs/2310.13032.

Xiaoyin Chen, Canwen Xu, Yite Wang, Boyi Liu, Zhewei Yao, and Yuxiong He. Learning to self-evolve. *arXiv preprint arXiv:2603.18620*, 2026. URL https://arxiv.org/abs/2603.18620.

Zhengbo Jiao, Hongyu Xian, Qinglong Wang, Yunpu Ma, Zhebo Wang, Zifan Zhang, Dezhang Kong, and Meng Han. Policy of thoughts: Scaling test-time training for LLM reasoning via online policy evolution. *arXiv preprint arXiv:2601.20379*, 2026. URL https://arxiv.org/abs/2601.20379.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025. URL https://arxiv.org/abs/2509.19349.

Shu Liu, Shubham Agarwal, Monishwaran Maheswaran, Mert Cemri, Zhifei Li, Qiuyang Mang, Ashwin Naren, Ethan Boneh, Audrey Cheng, Melissa Z. Pan, Alexander Du, Kurt Keutzer, Alvin Cheung, Alexandros G. Dimakis, Koushik Sen, Matei Zaharia, and Ion Stoica. EvoX: Meta-evolution for automated discovery. *arXiv preprint arXiv:2602.23413*, 2026a. URL https://arxiv.org/abs/2602.23413.

Ziyang Liu, Xinyan Guo, Xuchen Wei, Han Hao, and Liu Yang. Escher-Loop: Mutual evolution by closed-loop self-referential optimization. *arXiv preprint arXiv:2604.23472*, 2026b. URL https://arxiv.org/abs/2604.23472.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

Kainat Riaz, Muhammad Ahmed Mohsin, Ahsan Bilal, Muhammad Umer, Ayesha Mohsin, Aqib Riaz, Ali Subhan, and John M. Cioffi. Epistemic uncertainty for test-time discovery. *arXiv preprint arXiv:2605.11328*, 2026. URL https://arxiv.org/abs/2605.11328.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Yuvraj Virk, Zack Edds, Chunqiu Steven Xia, and Lingming Zhang. SwarmResearch: Orchestrating coding agents for open-ended discovery. *arXiv preprint arXiv:2607.02807*, 2026. URL https://arxiv.org/abs/2607.02807.

Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, Hao Cheng, Pengcheng He, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. ThetaEvolve: Test-time learning on open problems. *arXiv preprint arXiv:2511.23473*, 2025. URL https://arxiv.org/abs/2511.23473.

Yang Yang, Zining Zhong, Jindong Li, Jiemin Wu, Kaishen Yuan, Wenshuo Chen, Menglin Yang, and Yutao Yue. TurboEvolve: Towards fast and robust LLM-driven program evolution. *arXiv preprint arXiv:2604.18607*, 2026. URL https://arxiv.org/abs/2604.18607.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2402.01145.

Zongmin Yu and Liu Yang. Evolving ensemble of agents. *arXiv preprint arXiv:2605.09018*, 2026. URL https://arxiv.org/abs/2605.09018.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, and Yu Sun. Learning to discover at test time. *arXiv preprint arXiv:2601.16175*, 2026. URL https://arxiv.org/abs/2601.16175.

A Supplementary Results and Discovery Timing

This appendix gives the detailed online comparisons, frozen-policy probes, continuation-allocation diagnostics, and implementation settings.

A.1 Extended Baselines at Matched Task-Level Rollout Totals

PAST–ProgressEMA generates an additional fresh batch when it revisits a historical policy. To compare results at the same candidate-generation total, we extend both Current-only and UG-TTT to the observed PAST–ProgressEMA rollout total for each task while keeping each baseline's native configuration fixed. All six extended trajectories finish with a clean exit: both baselines run for 92, 85, and 71 epochs on CP26, AC1, and Erdős, respectively.

Table 2: **Best-found frontier at matched task-level rollout totals.** Higher is better for CP26; lower is better for AC1 and Erdős. Bold marks the strictly better result at the displayed precision; ties are left unbolded.

Task	Rollouts	Current-only	UG-TTT	PAST–ProgressEMA
CP26 ↑	5,888	2.635983	2.635983	2.635983
AC1 ↓	5,440	1.510404	1.509747	**1.507767**
Erdős ↓	4,544	0.381076	0.381048	**0.381020**

At the same candidate-generation total, PAST–ProgressEMA matches both baselines on CP26 and keeps a better frontier on AC1 and Erdős. This comparison aligns cumulative actual rollouts and complements the 50-update results in the main paper.

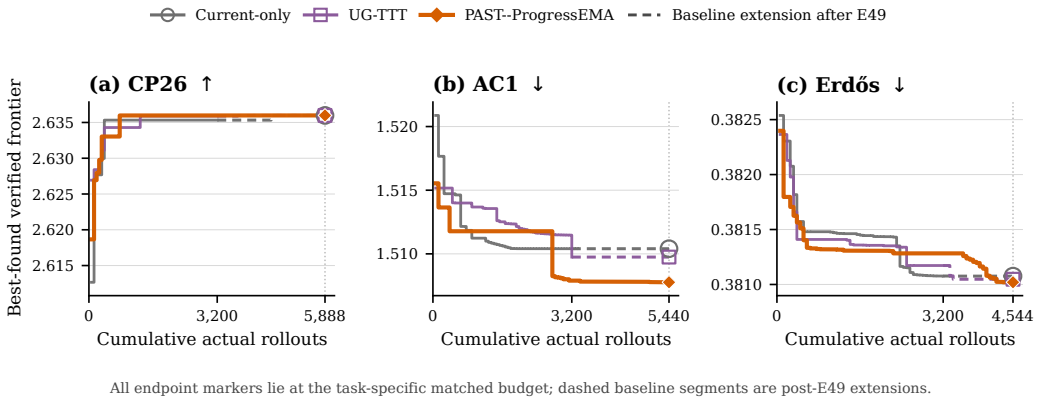


Figure 5: **Frontier trajectories at matched task-level candidate-generation rollout totals.** Solid segments show each method's first 50 epochs. The PAST–ProgressEMA curve includes direct and historical-fresh batches at their actual-rollout completion points. Dashed segments extend Current-only and UG-TTT beyond E49 until each reaches the corresponding PAST–ProgressEMA rollout total. At the matched endpoints, all three methods tie on CP26, while PAST–ProgressEMA retains the better frontier on AC1 and Erdős.

A.2 Multi-Seed Replication

We compare three aligned sampling-seed lanes at 50 optimizer updates. Current-only and PAST–ProgressEMA use sampling-seed offsets 0, 10,000,000, and 20,000,000 for S0–S2, respectively.

Table 3: **Three-task seed replication at 50 optimizer updates.** Higher is better for CP26; lower is better for AC1 and Erdős. Bold marks the better frontier in each aligned lane.

Task	Lane	Current-only	PAST–ProgressEMA
CP26 ↑	S0	2.635328412	**2.635983085**
	S1	2.635328412	**2.635983085**
	S2	2.635328412	**2.635983085**
AC1 ↓	S0	1.510404336	**1.507766741**
	S1	1.515593909	**1.512980523**
	S2	1.509315989	**1.507798697**
Erdős ↓	S0	0.381077078	**0.381019863**
	S1	0.380998042	**0.380980494**
	S2	0.381397757	**0.381023583**

PAST–ProgressEMA is better than Current-only in all nine paired 50-update lanes.

Common rollout budget. The two rollout-budget comparisons answer different questions. Section A.1 extends each baseline to the full task-specific rollout total of the primary PAST–ProgressEMA trajectory. Here, we instead compare the aligned seed lanes under a common cap of 3,200 actual rollouts. The 50-update comparisons do not align candidate-generation totals because PAST–ProgressEMA may generate an additional historical-fresh batch. We therefore truncate each PAST–ProgressEMA trajectory at the last complete round whose cumulative actual rollouts do not exceed 3,200, while Current-only reaches exactly 3,200 rollouts after 50 direct batches. The PAST–ProgressEMA S0/S1/S2 cutoffs are 3,200/3,200/3,136 on CP26, 3,200/3,136/3,200 on AC1, and 3,136/3,200/3,136 on Erdős.

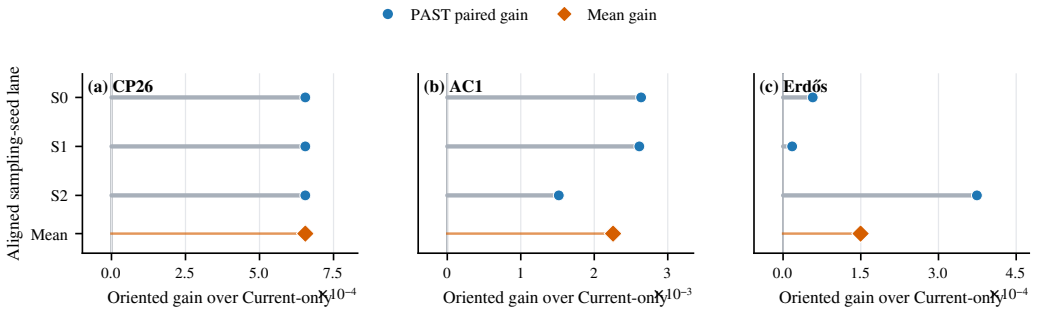


Figure 6: **Paired 50-update gains over Current-only.** Each point is one aligned sampling-seed lane, and the diamond is the mean across S0–S2. We orient each task so positive is better: PAST–ProgressEMA minus Current-only for CP26, and Current-only minus PAST–ProgressEMA for AC1 and Erdős. Each panel uses its own horizontal scale. Absolute frontier values are reported in Table 3.

Table 4: **Seed replication at a common** 3,200 **actual-rollout budget.** Means use the same three aligned sampling-seed lanes as Table 3. Higher is better for CP26; lower is better for AC1 and Erdős. "PAST wins" counts lane-wise comparisons.

Task	Current-only mean	PAST–ProgressEMA mean	PAST wins
CP26 ↑	2.635328412443	2.635983084915	3/3
AC1 ↓	1.511771411710	1.509558391519	3/3
Erdős ↓	0.381157625488	0.381097847724	2/3

At this common rollout budget, PAST–ProgressEMA has a better mean on all three tasks and wins every CP26 and AC1 lane and two of the three Erdős lanes.

A.3 Hybrid: Combining Same-Stage and Temporal Reuse

Hybrid combines UG-TTT's five-member same-stage LoRA ensemble with PAST's historical-policy continuation. We treat this as an end-to-end compatibility test rather than a one-factor ablation.

Table 5: Hybrid evaluation on three discovery tasks. Each configuration runs for 50 epochs. Higher is better for CP26 and lower is better for AC1 and Erdős. Bold marks the best result on each task.

Method	CP26 ↑	AC1 ↓	Erdős ↓
UG-TTT	2.635896	1.509747	0.381172
PAST–ProgressEMA	**2.635983**	**1.507767**	0.381020
Hybrid	2.631937	1.509119	**0.380972**

In the evaluated configurations, Hybrid outperforms both individual methods on Erdős, lies between them on AC1, and trails both on CP26; combining same-stage ensemble search with temporal continuation therefore does not guarantee additive gains.

A.4 Phase Structure of Epoch Trajectories

Table 6 reports discovery timing for Current-only, PAST–ProgressEMA, UG-TTT, and Hybrid. Each trajectory's oriented improvement from epoch 0 to its E0–E49 comparison endpoint is normalized to 100%. For Current-only and UG-TTT, E49 is the 50-update checkpoint of the longer matched-rollout trajectories in Section A.1; for PAST–ProgressEMA and Hybrid, E49 is the clean terminal endpoint. E10 and E25 report the fraction reached by those epochs, and Last event identifies the final frontier increment with $\Delta \geq 10^{-6}$.

Table 6: Frontier timing of selected trajectories. Gain share is measured against each trajectory's own improvement from epoch 0 to its E0–E49 comparison endpoint. For PAST–ProgressEMA and Hybrid, timing uses the recorded direct-evaluation frontier at each epoch; historical-fresh discoveries within an epoch are not resolved here.

Task	Method	Gain by E10	Gain by E25	Last event
CP26	Current-only	100.0%	100.0%	E5
	PAST–ProgressEMA	100.0%	100.0%	E9
	UG-TTT	82.1%	100.0%	E19
	Hybrid	100.0%	100.0%	E7
AC1	Current-only	83.3%	98.7%	E29
	PAST–ProgressEMA	24.4%	95.9%	E48
	UG-TTT	21.7%	51.5%	E49
	Hybrid	95.5%	95.5%	E49
Erdős	Current-only	72.4%	74.8%	E46
	PAST–ProgressEMA	78.2%	80.9%	E46
	UG-TTT	80.1%	84.4%	E38
	Hybrid	78.6%	97.3%	E45

The last meaningful event in all four CP26 trajectories occurs no later than E19. Hybrid reaches its endpoint by E7 and finishes below both individual methods. On AC1, PAST–ProgressEMA has completed 24.4% of its gain by E10 and reaches 95.9% by E25, followed by smaller late improvements. Hybrid gains almost entirely in the earliest stage. All four Erdős trajectories still improve after E25; Hybrid makes its main jump before E25 and keeps the best endpoint.

B Optimizer Handling Across Tasks

PAST normally restores a historical adapter together with its matching AdamW moments. We compare that rule with a fresh-on-switch variant that initializes a fresh optimizer whenever the selector chooses a historical policy. Each task has one complete 50-update pair; the selector, model, task, and training configuration are otherwise unchanged.

Table 7: Complete matching-versus-fresh-on-switch PAST–ProgressEMA trajectories. Δ is matching minus fresh in the task-native metric; higher is better on CP26 and lower is better on AC1 and Erdős. Rollouts count all direct and historical-fresh candidate generations. Bold marks the better endpoint on each task; neither arm is bolded for the numerical tie on CP26.

Task	Matching	Fresh	Δ	Rollouts (M/F)
CP26 ↑	2.635983085	2.635983085	$+6.9\times10^{-13}$	5,888/6,016
AC1 ↓	**1.507766741**	1.509772571	−0.002005831	5,440/6,144
Erdős ↓	0.381019863	**0.380954746**	+0.000065117	4,544/4,800

The direction is task-dependent: CP26 is numerically tied, matching restoration is better on AC1, and fresh-on-switch is better on Erdős. In every pair, this endpoint ordering is already established by the smaller terminal rollout total, so it is not explained by the additional candidate generations in the larger run. These are single complete trajectory pairs rather than replicated treatment estimates. They show that optimizer handling can alter the ensuing closed loop, but they do not identify one universally preferable rule.

B.1 Controlled CP26 Branch-Point Intervention

The full trajectories combine many sequential choices. To isolate one branch point, both E26 arms restore the same historical adapter under the same Solution Archive; the fresh arm initializes a new optimizer, whereas the matching arm restores the historical optimizer moments. Each arm then receives one update and is evaluated at the same 64 shared-seed rollout positions.

Table 8: Controlled CP26 E26 intervention. Difference is matching minus fresh. Candidate best considers only newly generated post-update candidates; retained frontier also includes the existing Solution Archive.

Metric	Matching	Fresh	Difference	Scope
Mean reward	1.9605	1.6327	$+0.3279$	E26 candidates
Validity	.9844	.8594	$+.1250$	E26 candidates
Candidate best	2.635983085	2.635977397	$+5.69 \times 10^{-6}$	E26 candidates
Retained frontier	2.635983085	2.635983085	numerical tie	E26 archive

The fresh arm has one evaluator timeout among 64 rollout positions. Assigning that position the geometric reward upper bound $\sqrt{26/\pi}$ and validity 1 still gives lower bounds of $+0.2829$ in mean reward and $+0.1094$ in validity for matching minus fresh. The intervention supports a local optimizer-memory effect at this branch point; it does not override the cross-task full-trajectory result above.

C Historical-Policy Opportunity: Existence, Budget Dependence, and Exploitability

The following frozen probes examine context dependence, budget dependence, and the reliability of policy rankings. They measure proposal behavior without further training; the optimizer interventions above address what happens after an update.

C.1 Cross-Task Frontier Recalibration

We apply the same six-checkpoint common-final-archive design to AC1 and Erdős. Both are minimization tasks, so each batch frontier is the lowest valid verifier objective and signed gain is parent objective minus candidate objective. The Base checkpoint's best valid batch frontier improves from 1.520873 to 1.510404 on AC1 and from 0.382538 to 0.381077 on Erdős when re-evaluated under the final archive. These correspond to relative recalibrations of 0.688% and 0.382%, respectively, compared with 0.780% on CP26. The checkpoints then reach nearly the same final-context batch frontier on AC1 and Erdős; their parent-wise frozen-policy envelopes are therefore substantially weaker than the CP26 envelope.

C.2 Existence: Policy Complementarity under a Common Context

This diagnostic freezes six checkpoints—Base, E5, E15, E25, E35, and E45—from one complete 50-epoch CP26 Current-only trajectory and places them under the same epoch-49 final Solution Archive. E5 denotes the policy after five optimizer updates, with the other labels following the same convention. We fix eight solution parents and generate eight fresh rollouts with shared seeds in every policy–parent cell, using a 10^6 seed offset independent of the training trajectory, for $6 \times 8 \times 8 = 384$ generations. Policy weights and the Solution Archive remain fixed throughout the probe.

Table 9: Local birth maxima and common-final-context aggregates for six frozen checkpoints. Valid is the legal fraction among 64 fresh rollouts; Improved counts contexts whose existing parent frontier is exceeded.

Policy	Local max	Final max	Valid	Median valid	Improved
Base	2.612660	2.633035	.625	2.626930	2
E5	2.635328	2.635328	.781	2.625892	1
E15	2.627955	2.627955	.766	2.626930	1
E25	2.627955	2.629628	.813	2.626930	1
E35	2.628037	2.629628	.875	2.626930	1
E45	2.629729	2.633035	.891	2.626930	2

The Spearman correlation between local-maximum and final-context rankings is $0.492592 \approx 0.493$; the corresponding correlation for validity rankings is $0.516100 \approx 0.516$. The hindsight winners

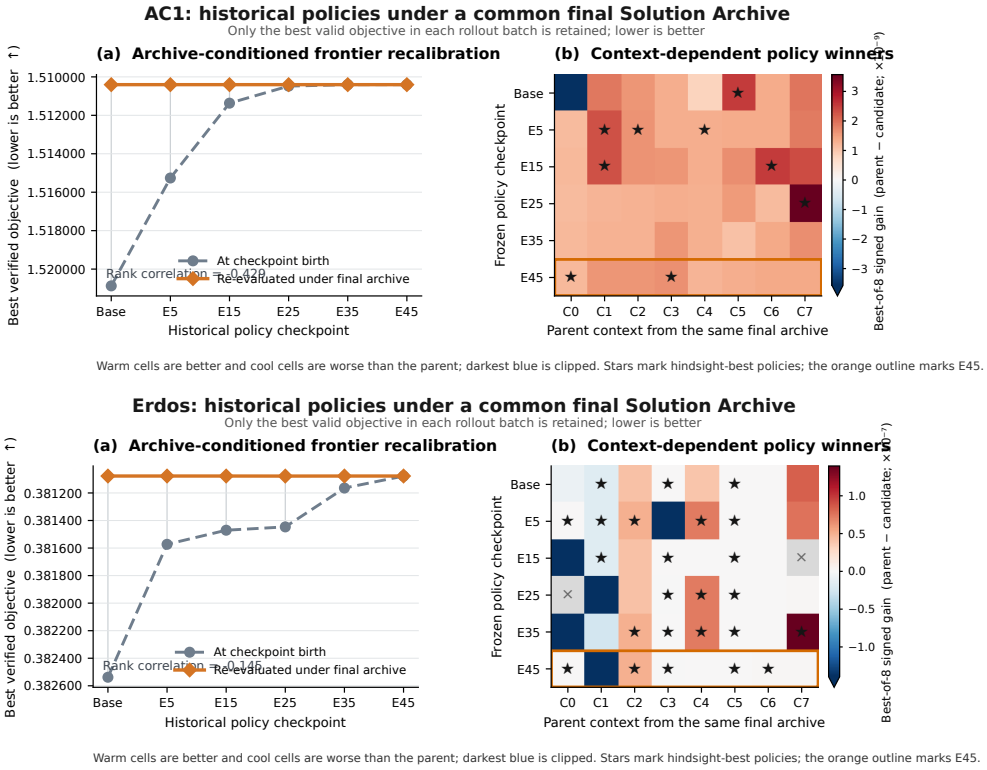


Figure 7: **Archive-conditioned frontier recalibration on AC1 and Erdős.** Top: AC1; bottom: Erdős. In each row, **(a)** compares the best valid objective in the checkpoint's original batch with a fresh re-evaluation under the common final Solution Archive. The objective axis is inverted so that upward movement indicates a lower, better frontier. **(b)** reports signed best-of-8 gain over each fixed parent, defined as parent objective minus candidate objective; warm colors indicate improvement and cool colors indicate regression. Stars mark the hindsight-best checkpoint for each parent, and the orange outline marks E45. Both tasks show birth-to-final frontier recalibration, while their parent-wise frozen-policy envelopes are weaker than the CP26 case in Figure 1.

across the eight fixed parents are E5, E45, E25, E35, E45, Base, Base, and E15. Every checkpoint wins at least one parent context, while E45 has the highest mean retained gain. The identity of the best checkpoint varies with the parent context.

C.3 Budget Dependence under Two Opportunity Estimands

Proposal-utility opportunity in a larger CP26 cross-stage probe. We decompose checkpoint-stage effects in a retrospective design with 4 trajectory contexts, 6 policies, 16 solution parents per context, and 8 rollouts per policy–parent cell, for $4 \times 6 \times 16 \times 8 = 3{,}072$ generations. The `all_policies` protocol evaluates every checkpoint in every context, including E25/E35/E45 under early Solution Archives. For valid rollouts let U be the verifier score and set $U = 0$ for invalid rollouts, so that

$$\mathbb{E}[U] = P(\text{valid})\,\mathbb{E}[\text{score} \mid \text{valid}].$$

In Table 10, $P(\text{improve})$ is the rollout-level $P(\text{valid} \wedge U > U_{\text{parent}} + 10^{-6})$; invalid rollouts count as non-improving.

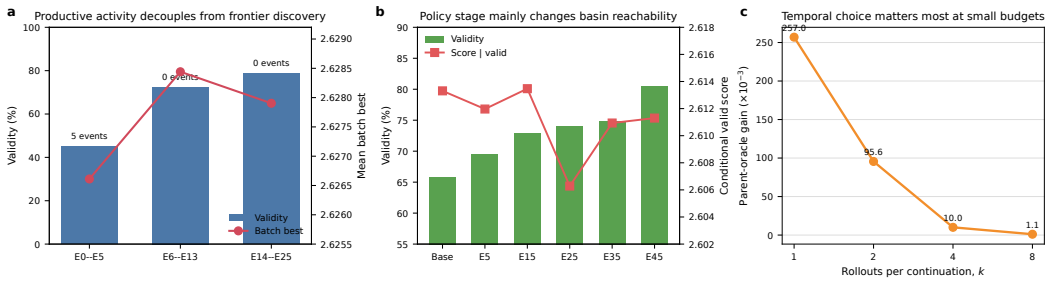


Figure 8: **Historical policies change proposal reachability, with proposal-utility opportunity concentrated at small rollout budgets.** (a) Phase decomposition of one CP26 Current-only trajectory: validity keeps increasing, while the observed 26-epoch prefix has no frontier events from E6 onward. (b) The 3,072-rollout retrospective cross-stage probe evaluates six frozen checkpoints under four Solution Archive contexts. Later checkpoints achieve higher validity, while valid-conditioned scores vary non-monotonically with stage. (c) The hindsight parent-oracle gain in proposal utility, equally weighted over 42 unique parents, shrinks rapidly with the rollout count k per continuation.

Table 10: Rollout-weighted decomposition for six CP26 policies in the cross-stage proposal-utility probe. Each row contains 512 rollouts.

Policy	Validity	$\mathbb{E}[U]$	Score | valid	$P(\text{improve})$
Base	.6582	1.7201	2.6133	.0215
E5	.6953	1.8161	2.6120	.0371
E15	.7285	1.9040	2.6135	.0371
E25	.7402	1.9293	2.6063	.0254
E35	.7480	1.9531	2.6109	.0469
E45	.8047	2.1013	2.6113	.0410

Across the 24 context–policy aggregates, the Pearson correlation between validity and expected utility is 0.999811, while the correlation between validity and conditional valid score is -0.191752. In these aggregates, checkpoint-stage differences in expected utility track changes in validity much more closely than changes in the score conditional on validity.

For the proposal-utility budget diagnostic, we enumerate all without-replacement k-subsets within each 8-rollout cell, first average repeated contexts for the same parent, and then aggregate with equal weight over 42 unique parents.

Table 11: CP26 parent-wise proposal-utility opportunity at different per-continuation rollout budgets. The hindsight parent oracle selects the policy with the best observed outcome for each parent.

k	Global E45	Parent oracle	Oracle gain
1	2.111925	2.368881	0.256956
2	2.506427	2.601993	0.095566
4	2.617416	2.627463	0.010046
8	2.627627	2.628720	0.001092

For CP26 proposal utility, the hindsight oracle gap shrinks by approximately $235\times$ from $k = 1$ to $k = 8$. Checkpoint differences in the best proposal are most visible when each cell receives few attempts.

Archive-retained-frontier opportunity in aligned cross-task probes. A second diagnostic uses aligned 6-policy $\times$ 8-parent $\times$ 8-rollout frozen matrices for CP26, AC1, and Erdős. For every policy–parent pair and budget k, we exactly enumerate the $\binom{8}{k}$ rollout-index subsets and compute

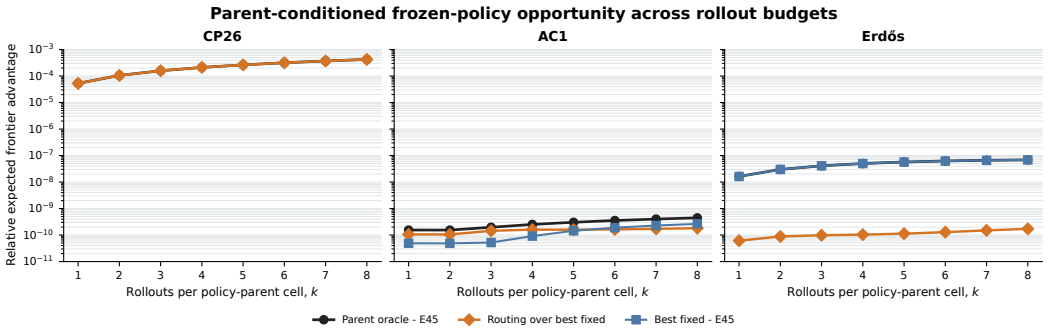


Figure 9: **Parent-conditioned frozen-policy opportunity for expected archive-retained frontier.** Each task decomposes the parent oracle's advantage over E45 into the best-fixed-checkpoint component and the additional parent-routing component. Values are divided by the task's mean absolute parent objective and shown on a shared logarithmic scale, with exact zeros omitted. CP26 retains a measurable routing component across $k = 1, \ldots, 8$, whereas AC1 and Erdős remain roughly six orders of magnitude smaller.

the expected archive-retained frontier, including the parent itself. The best fixed checkpoint and the checkpoint selected separately for each parent are chosen using these expected cell frontiers.

In absolute oriented frontier units, CP26's routing component grows from 1.386×10^{-4} at $k = 1$ to 1.109×10^{-3} at $k = 8$. AC1 changes from 1.599×10^{-10} to 2.742×10^{-10}, and Erdős from 2.338×10^{-11} to 6.546×10^{-11}. The task-normalized curves preserve this separation. Erdős' total advantage over E45 comes almost entirely from E35 as a better fixed checkpoint. The aligned matrices therefore reveal a strong parent-routing component on CP26 and much smaller components on AC1 and Erdős.

C.4 Exploitability: Portfolio Structure and Ranking Reliability

Table 12 summarizes the CP26 frozen policy–context matrices labeled A, B, and C by seed lane in the source analysis. Envelope gap compares the context-wise oracle with the best fixed policy; latest regret compares the oracle with E45; rank reversal fraction records whether each of the 15 policy pairs reverses order across contexts; winner entropy is the normalized Shannon entropy of the bootstrap winner distribution over six policies.

Table 12: Portfolio diagnostics for three frozen policy–context matrices.

Matrix	Envelope gap	Latest regret	Reversal frac.	Context corr.	Winner entropy
A	.001578	.001578	1.000	−.060	.370
B	.000217	.001868	1.000	.022	.436
C	.001172	.001494	1.000	.079	.445

Table 13 compares two summaries of the same frozen matrix. The first reduces each policy–parent cell to its best-of-8 outcome; the interaction term is computed from these cell summaries. The second retains all eight rollouts within each cell and uses positive improvement relative to the parent as utility.

Table 13: Two-scale sum-of-squares decomposition for the Seed A frozen matrix. The cell-summary row contains 6×8 best-of-8 signed group-gain summaries; the per-rollout row contains all 384 observations.

Statistical unit	Policy	Parent	Interaction	Within-cell residual
Best-of-8 signed group gain	4.0%	17.9%	78.1%	—
Per-rollout $u = \max(0, \text{child} - \text{parent})$	0.6%	2.2%	9.5%	87.7%

At the best-of-8 cell-summary level, policy–parent interaction accounts for 78.1% of the observed sum of squares. With all rollouts retained, the within-cell residual accounts for 87.7%. These two summaries capture both context-dependent checkpoint performance and substantial variation among proposals generated within the same cell.

We assess the repeatability of policy rankings in the 3,072-rollout cross-stage probe. Splitting each 8-rollout cell into two fixed 4-rollout halves gives a split-half correlation of 0.0322 for the utility-interaction residual, with top-policy agreement of 18.75%, close to the six-policy chance level of 16.67%. Across 14 exact-state repeated parents, the parent-equal policy-rank Spearman correlation is 0.0259 with a 95% parent-bootstrap interval of $[-0.1714, 0.1980]$; winner agreement is 19.05%. The selected winner thus varies substantially between repeated observations.

We perform leave-one-trajectory-out routing across the three frozen matrices. Utility is each rollout's positive improvement relative to its parent, with invalid rollouts set to zero. Evaluation aggregates 1,152 rollouts into 24 held-out solution contexts.

Table 14: Held-out policy routing in the three-seed frozen LOSO evaluation (24 held-out solution contexts). The hindsight oracle selects from the observed held-out cell outcomes. The other rows use fixed selection rules or training-fold information.

Selector	Mean EI	Oracle fraction	Oracle hit
Latest E45	2.9291×10^{-5}	26.4%	8.3%
Train-fold global best	1.0527×10^{-5}	9.5%	16.7%
Uniform random portfolio	2.4050×10^{-5}	21.7%	16.7%
Contextual ridge	1.0527×10^{-5}	9.5%	16.7%
Hindsight context oracle	1.1103×10^{-4}	100%	100%

Nested validation shrinks the contextual ridge to the train-fold global best, whose held-out mean EI is lower than the uniform random portfolio. The hindsight oracle achieves a larger mean EI than all evaluated routing rules.

The frozen probes reveal context-dependent proposal performance and variable policy rankings across repeated observations. The budget trends depend on the measured quantity: the CP26 proposal-utility envelope is largest at small k, while the archive-retained routing component grows with k. The corresponding AC1 and Erdős routing components are much smaller. These observations motivate allocating continuations using feedback acquired as search proceeds.

D Process Evidence for Temporal Search

The diagnostics in this section use the nine complete 50-update PAST trajectories underlying Table 1. Epoch-level event statistics use the recorded direct-evaluation frontier; batch-resolved source tracing also includes historical-fresh evaluations in execution order. Selector statistics use the runtime-recorded historical-selection flag and parent epoch. They describe observed trajectories and do not identify causal effects of choosing a particular parent.

D.1 Program-Family Evolution

Figure 4 summarizes the three primary PAST–ProgressEMA trajectories. Its upper row uses the recorded verified frontier. Its lower row counts only verified programs newly admitted to the So-

lution Archive, grouped into five-update bins; it does not summarize all generated rollouts. The historical-parent curve is the fraction of updates in the same bin whose recorded training parent is an earlier policy.

We assign one deterministic family label to each admitted program from the reachable code path of its `run`, `main`, or top-level entry point. CP26 families distinguish hexagonal or staggered constructions, edge-heavy row layouts, interior-heavy row layouts, and grid or restart-based programs. AC1 families distinguish local search, LP with structured starts, LP with targeted mutation, and population or global search. Erdős families distinguish local refinement, population or heuristic search, global search, and combined global–local search. More specific labels take precedence when a program contains multiple inherited helpers; representative early, middle, late, and frontier-linked programs were manually checked against their executed solver, initialization, and construction paths.

D.2 Frontier Increments by Generating Batch

We reconstruct all 248 batches in the three primary PAST–ProgressEMA trajectories: 50 direct batches per task and 42, 35, and 21 historical-fresh batches on CP26, AC1, and Erdős. Within a round, the direct batch precedes the historical-fresh batch when present. Each batch receives the positive increase over the best verified frontier observed before it; shares divide these increments by the total frontier improvement after the root evaluation.

Direct batches are grouped by the policy selected for the preceding update. This grouping identifies the immediate training edge, but a latest-parent continuation can still extend a branch created by an earlier historical selection.

Table 15: Observed frontier improvement by generating batch. Each task uses one primary PAST–ProgressEMA trajectory. Entries are percentages of the trajectory's total oriented gain after root evaluation, assigned in execution order.

Generating batch	CP26	AC1	Erdős
Historical-fresh	0.00	24.03	18.25
Direct: child of historical parent	16.98	47.98	15.68
Direct: child of latest parent	83.02	27.99	66.08

The total oriented gains are 0.0173231180 on CP26, 0.0077739127 on AC1, and 0.0013781042 on Erdős. CP26's historical-fresh increment is 2.03×10^{-13} and rounds to 0.00%. A CP26 historical continuation at E7 creates the child whose E8 direct batch advances the global frontier from 2.634292 to 2.635977; continuing that child produces a further 5.69×10^{-6} gain at E9. On AC1, historical resampling at E3 advances the frontier by 0.0018679631, and the direct evaluation of a historical-continuation child at E22 advances it by 0.0035296852. The reconstructed terminal frontiers match the reported endpoints on all three tasks.

D.3 Selector Allocation and Policy Topology

Each trajectory ends with 51 policies: one root and 50 trained children. Excluding E0 root initialization, let n_j count selections of policy j over E1–E49 and define

$$H_{\text{policy}} = -\sum_j p_j \log_2 p_j, \qquad p_j = \frac{n_j}{\sum_k n_k}.$$

We normalize by $\log_2 50$, the maximum entropy over the final 50 eligible parent policies.

Table 16: Temporal allocation and frontier conversion for the nine complete PAST trajectories. Hist. is the recorded historical selections among 49 continuation decisions; event rates count direct-evaluation increments with $\Delta \geq 10^{-6}$ per 1,000 actual rollouts.

Task	Selector	Hist.	Parents	$H/\log_2 50$	Events/1k
CP26	ProgressEMA	42/49	10	.493	1.019
CP26	ScorePUCT	30/49	33	.872	1.367
CP26	Uniform	46/49	23	.749	.326
AC1	ProgressEMA	35/49	16	.522	3.493
AC1	ScorePUCT	8/49	41	.930	1.078
AC1	Uniform	44/49	25	.755	1.330
Erdős	ProgressEMA	21/49	29	.792	5.062
Erdős	ScorePUCT	3/49	46	.970	1.474
Erdős	Uniform	44/49	25	.755	2.826

ProgressEMA's historical-selection share decreases from 85.7% on CP26 to 71.4% on AC1 and 42.9% on Erdős, while its direct-frontier event rate increases. Uniform frequently revisits historical policies on every task but has a lower event rate than ProgressEMA. ScorePUCT spreads selections across many parents and is nearly current-following on AC1 and Erdős. Archive use and frontier conversion are therefore distinct quantities.

Table 17: Parent-link topology for the nine complete PAST trajectories; every archive has 51 policies. Longest path counts policies on a root-to-leaf path, and refresh edges are children produced after selecting a historical parent.

Task	Selector	Longest path	Branch points	Leaves	Refresh edges
CP26	ProgressEMA	9	9	41	42
CP26	ScorePUCT	22	14	17	30
CP26	Uniform	7	13	28	46
AC1	ProgressEMA	12	7	35	35
AC1	ScorePUCT	43	6	9	8
AC1	Uniform	7	10	26	44
Erdős	ProgressEMA	22	9	22	21
Erdős	ScorePUCT	48	2	4	3
Erdős	Uniform	7	10	26	44

ScorePUCT forms deep, nearly single-chain genealogies on AC1 and Erdős; Uniform repeatedly returns to older policies; ProgressEMA forms multi-branch genealogies whose depth and revisit rate vary by task. These are one-trajectory descriptions for each task–selector cell, so tree shape alone cannot establish a frontier effect.

Table 18: Sensitivity of frontier-event counts to the numerical threshold for all nine PAST trajectories. The main-paper convention is 10^{-6}.

Task	Selector	10^{-12}	10^{-9}	10^{-6}	10^{-5}	10^{-4}
CP26	ProgressEMA	6	6	6	5	5
CP26	ScorePUCT	7	7	7	6	6
CP26	Uniform	3	3	2	2	2
AC1	ProgressEMA	20	20	19	12	4
AC1	ScorePUCT	4	4	4	4	4
AC1	Uniform	8	8	8	7	5
Erdős	ProgressEMA	34	34	23	14	4
Erdős	ScorePUCT	5	5	5	5	5
Erdős	Uniform	19	19	17	12	4

Threshold variation changes the number of counted micro-improvements on AC1 and Erdős. At 10^{-6}, ProgressEMA has the highest event rate among PAST selectors on AC1 and Erdős, whereas ScorePUCT is highest on CP26; this ordering is not created by a single floating-point threshold.

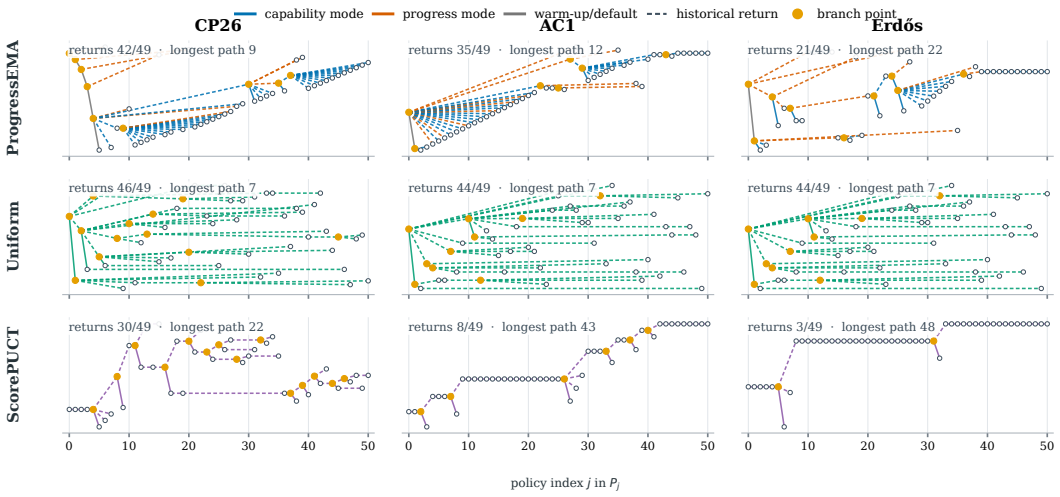


Figure 10: Parent-link topology for all nine complete PAST trajectories. Columns are tasks and rows are selectors. Node P_j is saved policy j; solid edges continue from the latest policy, dashed edges return to an earlier policy, and orange nodes are branch points. Vertical position only separates branches.

E Task Definitions and Verifiers

All three tasks require an executable candidate construction. The cumulative frontier tracks the best legal objective recomputed by the task verifier.

Circle Packing 26 (CP26). The task places 26 non-overlapping circles in the unit square and maximizes

$$F_{\mathrm{CP26}} = \sum_{i=1}^{26} r_i.$$

The verifier requires center and radius shapes $(26, 2)$ and $(26,)$, finite real values, $r_i \geq 0$, every circle fully contained in $[0, 1]^2$, and pairwise non-overlap. It computes the score directly from the radii. The prompt uses 2.636 as a search target.

Autocorrelation 1 (AC1). The task searches for a nonnegative real sequence $a = (a_0, \ldots, a_{n-1})$ minimizing

$$J(a) = \frac{2n \max_j (a * a)_j}{\left(\sum_i a_i\right)^2},$$

where $a * a$ is discrete convolution. The evaluator accepts only a nonempty `list[float]`, rejects booleans, nonnumeric values, NaNs, and infinities, clips coefficients to $[0, 1000]$, and requires a sum of at least 0.01. Paper tables report the verifier-recomputed raw bound, where lower is better. The engine uses a reciprocal reward to express improvement in the maximization direction during training. The Solution Archive admits AC1 constructions with 1,000 to 100,000 coefficients.

Erdős minimum-overlap problem. The task searches for a step function satisfying

$$h : [0, 2] \to [0, 1], \qquad \int_0^2 h(x)\,\mathrm{d}x = 1$$

and minimizes

$$C_5(h) = \max_k \int h(x)\big(1 - h(x + k)\big)\,\mathrm{d}x.$$

The discretized evaluator first checks dimensionality, length, finite real values, and the range $[0, 1]$; when needed, it normalizes the sample sum to $n/2$ and checks the range again. For the resulting samples $h_0, \ldots, h_{n-1}$, it computes

$$\widehat{C}_{5,n}(h) = \Delta x \max_{\ell \in \{-(n-1),\ldots,n-1\}} \sum_{\substack{0 \le i < n \\ 0 \le i+\ell < n}} h_{i+\ell}(1 - h_i), \qquad \Delta x = \frac{2}{n}.$$

This is the finite, non-circular full correlation implemented by `np.correlate(h, 1-h, mode="full")`. The candidate's declared C_5 must agree with the recomputed value within `atol=1e-4`; both the archive and paper use the recomputed bound, where lower is better. The Solution Archive admits variable-resolution constructions with at most 1,000 piece values.

F Implementation and Hyperparameters

Shared solution-search and training setup. Current-only and all PAST selectors share the solution-side PUCT, candidate generation, and adaptive entropic objective of TTT-Discover (Yuksekgonul et al., 2026). For every verifier-evaluated rollout x, the task reward recorded by the policy learner and continuation selectors is

$$r(x) = \begin{cases} R(x), & x \text{ is valid on CP26,} \\ 1/(R(x) + 10^{-8}), & x \text{ is valid on AC1 or Erdős,} \\ 0, & x \text{ is invalid.} \end{cases}$$

Only valid candidates enter the Solution Archive. On a fresh run, the archive is initialized with eight task-provided root states, matching the eight solution-parent slots. Valid children are deduplicated using task-specific code or construction keys, and at most the two highest-valued children of each parent are retained. The initial roots are preserved; if the archive exceeds 1,000 entries, the remaining non-root states are ranked by task-oriented archive value and truncated to capacity. Given a selected continuation state $z(v)$, solution-side PUCT chooses eight solution parents and generates $M = 8$ candidates from each. For rewards $\{r_{g,j}\}_{j=1}^{M}$ from solution parent g, the adaptive inverse temperature β_g satisfies

$$p_{g,j}(\beta_g) = \frac{\exp(\beta_g r_{g,j})}{\sum_{\ell=1}^{M} \exp(\beta_g r_{g,\ell})}, \qquad D_{\mathrm{KL}}(p_g(\beta_g) \,\|\, \mathrm{Unif}_M) \approx \log 2. \tag{12}$$

Groups with constant recorded rewards are removed before this fit. If the target KL is not reached within the numerical search range, the implementation uses the largest searched inverse temperature. The corresponding leave-one-out entropic advantage is

$$A^{\mathrm{ent}}_{g,j} = \frac{\exp(\beta_g r_{g,j})}{\frac{1}{M-1}\sum_{\ell \neq j} \exp(\beta_g r_{g,\ell})} - 1. \tag{13}$$

After a KL adjustment relative to the base policy, this advantage enters the importance-ratio objective for one LoRA/optimizer update. Current-only and all PAST selectors share this solution search and training objective; PAST changes which continuation parent receives the update.

Table 19: Formal common configuration for Current-only and PAST.

Component	Configuration
Base model / context	Qwen3-8B / 32,768 tokens
Search horizon	50 epochs; one optimizer update per epoch
Direct generation	8 parents × 8 rollouts; temperature 1.0
Solution Archive	maximum 1,000 entries
Optimizer	AdamW; lr 4×10^{-5}; $(\beta_1, \beta_2) = (0.9, 0.95)$
Optimizer regularization	$\epsilon = 10^{-8}$; weight decay 0; KL coefficient 0.1
LoRA	rank 32; alpha 32; dropout 0
LoRA targets	q/k/v/o, gate/up/down projections, and LM head
ProgressEMA (main runs)	$\rho = 0.7$; $\eta = 0.95$; selector seed 0
ScorePUCT	exploration coefficient $c_{\mathrm{puct}} = 1.0$

ProgressEMA schedule. All reported PAST–ProgressEMA trajectories start from scratch at the policy root. The three main-task trajectories use $\rho = 0.7$, $\eta = 0.95$, and selector seed 0. Stochastic selector choices use this fixed seed, and ties follow a fixed creation-order and policy-identifier order. Equation (10) is updated after the child's direct evaluation finishes in the next round, and every selection uses the observations available at that time.

Policy Archive storage and checkpoint I/O. We measure the recoverable states in the completed PAST–ProgressEMA trajectory for each task by enumerating its files. Every trajectory contains one root node and 50 trained child nodes; each child stores a complete LoRA adapter and four-shard optimizer state. Across the tasks, one state occupies approximately 1.429 GB for the adapter and 1.108 GB for the optimizer, or 2.537 GB in total. The 50 states occupy approximately 126.841 GB, the base model is shared by all nodes, and the Policy Archive metadata is smaller than 0.001 GB (all GB values use 10^9 bytes). Let S_{adapter}, S_{opt}, and S_{meta} denote the adapter, optimizer-state, and metadata sizes. The logical storage for N recoverable states is

$$S_N = N(S_{\mathrm{adapter}} + S_{\mathrm{opt}}) + S_{\mathrm{meta}}.$$

This total archive footprint grows linearly with the number of retained states.

Table 20: Checkpoint I/O for completed PAST–ProgressEMA trajectories on AC1 and Erdős. Persistence and optimizer-shard loading are median [IQR]. Each task has 50 persistence samples; AC1 and Erdős have 28 and 46 successful optimizer-loading samples, respectively. Aggregate share divides the sum of these measured intervals by total recorded execution time over 50 epochs.

Task	Persistence (s)	Opt. restore (s)	Aggregate share
AC1	3.39 [3.28, 3.69]	0.324 [0.273, 0.345]	0.048%
Erdős	3.28 [3.04, 3.69]	0.302 [0.254, 0.359]	0.062%

Checkpoint persistence measures writing the post-update adapter and optimizer state. The optimizer-restore column times optimizer-shard loading within trainer initialization. Each restore reads one selected state, while total stored checkpoint data grows with archive size.

ScorePUCT and Uniform controls. For the kth complete on-policy batch $D_k(v)$ observed for active node v, ScorePUCT takes the task-reward batch-best and averages across the node's observations:

$$b_k(v) = \max_{x \in D_k(v)} r(x), \qquad Q(v) = \frac{1}{N(v)} \sum_{k=1}^{N(v)} b_k(v), \tag{14}$$

where $N(v)$ counts the complete direct and historical-fresh batches observed for v. Define

$$N_t^{\Sigma} = \sum_{w \in \mathcal{V}_t^{\text{active}}} N(w), \tag{15}$$

$$\Delta_{Q,t} = \max\left\{ \max_{w \in \mathcal{V}_t^{\text{active}}} Q(w) - \min_{w \in \mathcal{V}_t^{\text{active}}} Q(w), 10^{-6} \right\}, \tag{16}$$

$$U_t(v) = Q(v) + \frac{c_{\text{puct}} \Delta_{Q,t}}{|\mathcal{V}_t^{\text{active}}|} \frac{\sqrt{N_t^{\Sigma}}}{1 + N(v)}. \tag{17}$$

ScorePUCT selects

$$\hat{v}_t = \arg\max_{v \in \mathcal{V}_t^{\text{active}}} U_t(v), \qquad c_{\text{puct}} = 1.0. \tag{18}$$

The uniform prior, Q-range floor, total-visit factor, and $(1 + N(v))^{-1}$ decay match the selector used in the reported runs. Here $Q(v)$ is the mean performance across the node's observed batches, while ProgressEMA capability $s(v)$ is the largest observed batch-best reward. Uniform selects

$$\hat{v}_t \sim \text{Unif}(\mathcal{V}_t^{\text{active}}). \tag{19}$$

Both controls select active nodes, and ScorePUCT ties follow the same fixed node ordering as ProgressEMA.

UG-TTT retains the same base model, task interface, solution-search protocol, 50-epoch horizon, and learning rate, and uses its ensemble-specific configuration: AdamW $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0, KL coefficient 0.01, predictive-MI exploration coefficient 0.1, nuclear-norm coefficient 0.075, and rank-16, alpha-32, dropout-0.05 LoRA adapters on the q/k/v/o attention projections. These native UG-TTT coefficients are used unchanged across tasks, without task-specific retuning. Each of the eight solution-parent groups contributes eight rollouts. Within a group, the rollouts are assigned round-robin across the five ensemble members, with the three members receiving a second rollout rotating across epochs. Every trainable rollout is then scored under all five members, and the resulting loss-sample cohort is shared across the five member updates rather than being restricted to its generating member.