EVOLPO: CONTINUAL REINFORCEMENT LEARNING THROUGH POLICY EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual reinforcement learning (RL) with verifiable rewards typically resumes each new task from the latest checkpoint. Although recent work reduces forgetting with replay and regularization and ranks starting checkpoints with short adaptation trials, the recovery step that follows adaptation is left out of this choice. This matters because the latest checkpoint may have the highest current score and still be a poor starting point: adaptation changes its capabilities and recovery changes them again. We therefore treat checkpoint selection as a two-stage decision, in which a controller chooses a parent from an archive of past checkpoints, observes the child produced by adaptation and then chooses how to recover earlier capabilities. Both decisions share the utility of the completed child, so each parent is valued by its outcome after recovery rather than before it. Our approach, **Evolutionary Lineage Policy Optimization (EvoLPO)**, keeps the inner RL optimizer fixed and deploys a single policy at every stage. Across mathematics, code-generation and program-optimization streams, EvoLPO improves new-task learning, retention and final score over sequential training and a sequential two-head controller and exceeds tuned replay in final score at equal total compute; the gain over the two-head controller holds across model scales and unseen model families.

1 INTRODUCTION

Continual reinforcement learning with verifiable rewards (RLVR) trains one language model on a stream of tasks and each new task must be learned without losing skills acquired earlier. Replay, regularization and constrained objectives address this trade-off by changing the data or the loss along a single training trajectory (Kirkpatrick et al., 2017; Lopez-Paz & Ranzato, 2017; Luo et al., 2026a;b; Qin et al., 2026). However, all of these methods keep one design choice fixed: the next update starts from the latest checkpoint.

The latest checkpoint is not always the best place to start. It may have the highest current score, but the next update changes what it can do: adaptation to the new task shifts both acquisition and retention and the recovery step that follows shifts them again. For example, an older checkpoint may learn the arriving task faster but forget a protected skill. If verified donor repair restores that skill, the older checkpoint becomes the better choice, but only after recovery. This leads to our central question:

Which checkpoint should initialize the next update,
when its value depends on the recovery that follows?

Forking every candidate checkpoint under every recovery answers it exactly, but costs one complete training branch per pair. Checkpoint-readiness methods are cheaper: they run short adaptation trials from several candidates (Li et al.; Lillo & Cheney, 2026). Their trials, however, stop before recovery, so they miss cases where recovery changes which candidate is best.

We introduce **Evolutionary Lineage Policy Optimization (EVOLPO)**, a controller that answers this question without running every branch at deployment: it predicts the utility of the *completed child*, the checkpoint obtained after both adaptation and recovery. EVOLPO keeps a bounded archive of past checkpoints. At each task boundary, it chooses a parent from the archive, adapts it with a fixed inner RL optimizer, observes the resulting provisional child and then chooses one of three

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

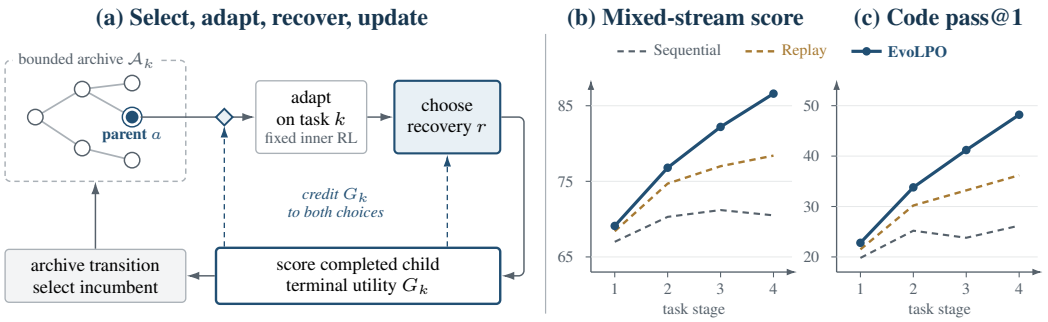


Figure 1: **EvoLPO overview. (a)** At stage k, the controller picks a parent a from the bounded archive $\mathcal{A}_k$, a fixed inner optimizer adapts it on task k, and the controller picks a recovery r. Terminal utility G_k credits both choices (dashed) and drives the archive transition, which selects the deployed incumbent. **(b–c)** Deployed score per stage on the mixed and code streams.

recovery actions: continued training, on-policy prompt replay, or verified donor repair. Both choices are credited with the utility of the completed child, as shown in Figure 1(a).

The difficulty is that a parent must be valued before its recovery is chosen. A natural two-head controller, which we call Seq-2H, values each parent under the random recovery mix used to collect its training data. At deployment, however, the learned recovery head chooses the recovery, so Seq-2H values parents for a recovery policy it never uses. EvoLPO instead weights each parent's observed outcomes by the learned recovery policy, so each parent is valued for the recovery it will actually receive. Because a fixed archive rule, not the controller, decides which completed child is deployed, we can measure separately whether EvoLPO picks better children and whether those children improve the deployed policy.

We evaluate EvoLPO on continual mathematics, code-generation and program-optimization streams with Qwen models from 1.7B to 30B parameters. Changing only the parent target improves final score over Seq-2H by 0.70 points on streams held out from controller fitting (paired 95% CI $[0.10, 1.30]$). With in-family controller fitting, EvoLPO improves final score over Seq-2H by 1.79 points and raises both new-task learning and retention. At the same total H100-hour budget, including controller preparation, it exceeds tuned replay by 0.74 points. The gains persist on twelve-stage streams and transfer to five unseen model families.

To summarize, we (i) formulate checkpoint selection for continual RLVR as a two-stage decision whose value depends on the recovery that follows adaptation, (ii) introduce EvoLPO, which trains parent selection on completed-child outcomes weighted by the learned recovery policy and (iii) show through matched, equal-compute experiments across streams, model scales and model families that recovery-aware selection improves the deployed policy.

2 Related Work

Continual learning for language models. Regularization, replay, consolidation and fast–slow models preserve earlier skills as one policy changes (Kirkpatrick et al., 2017; Lopez-Paz & Ranzato, 2017; Schwarz et al., 2018; Tiwari et al., 2026). CRG, CPO and ReMind adapt this goal to RLVR (Luo et al., 2026a;b; Qin et al., 2026) and continual RLHF and controlled-text RL modify the objective or data along the current trajectory (Zhang et al., 2024; Shulev & Sima'an, 2024). These methods decide how to train from the current checkpoint. EvoLPO instead decides which stored checkpoint to train from and it uses on-policy replay as one of its recovery actions.

Choosing a starting checkpoint. Meta-learning and policy reuse prepare models to adapt from prior experience (Finn et al., 2017; Berseth et al., 2021; Li et al., 2024; Hu et al., 2026). Checkpoint-readiness methods predict later RL potential from lightweight training or uncertainty statistics (Li et al.; Nguyen et al., 2025) and TeLAPA tests archived initializations with reset short trials (Lillo & Cheney, 2026). These methods show that the best current policy need not be the best starting point, but they score a candidate at the end of adaptation. When a recovery step follows, its effect

can reorder the candidates (Section 7). EvoLPO scores each candidate after recovery and weights the recovery outcomes by the policy that will be deployed. Table 32 compares it with a matched TeLAPA-style selector.

Two-stage decisions. Q-learning for dynamic treatment regimes fits the later decision first and then values the earlier decision under that fitted policy (Murphy, 2003; 2005; Chakraborty & Moodie, 2013). EvoLPO uses the same order, but each of its labels requires adapting a checkpoint, running a recovery and verifying the result. Instead of fitting a critic for the later decision, EvoLPO forks each provisional child, executes every recovery and weights the observed outcomes with a cross-fitted recovery policy (Section B). Lineage-Value Policy Gradients also propagate credit through descendants, but study multistep evolutionary search rather than a parent–recovery decision (Siper et al., 2026).

Populations and parameter merging. PBT, MAP-Elites, genetic distillation and HELIX evolve populations of policies (Jaderberg et al., 2017; Mouret & Clune, 2015; Gangwani & Peng, 2017; Su et al., 2026), while soups, task arithmetic, TIES and policy subspaces combine parameters (Wortsman et al., 2022; Ilharco et al., 2022; Yadav et al., 2023; Kobanda et al., 2024). EvoLPO instead keeps realized checkpoints separate, selects one parent and trains one completed child per stage. Its inner optimizer is held fixed and we test both GRPO and TROLL in that role (Shao et al., 2024; Becker et al., 2026).

3 Problem Formulation

Setting. Tasks arrive in order as distributions $\mathcal{D}_1, \ldots, \mathcal{D}_K$. Before stage k, the learner deploys one policy, the *incumbent* θ_k^{inc} and keeps a bounded *archive* $\mathcal{A}_k = \{\theta_k^1, \ldots, \theta_k^{M_k}\}$ of past checkpoints with $M_k \le M$. It may use training and development data from tasks $1{:}k$, but never future tasks or sealed evaluation data. Each encountered task j has a score $\mathsf{S}_j(\theta) \in [0,1]$ and the stage-k utility averages these scores with fixed weights,

$$\mathsf{U}_k(\theta) = \sum_{j=1}^{k} w_{k,j}\mathsf{S}_j(\theta), \qquad w_{k,j} \ge 0, \quad \sum_{j=1}^{k} w_{k,j} = 1. \tag{1}$$

A sequential learner applies the task-k adaptation Φ_k to the incumbent, $\widetilde{\theta}_k = \Phi_k(\theta_k^{\mathrm{inc}})$. Replay changes the data used by Φ_k, but it still starts from θ_k^{inc}.

Completed children. To compare starting points, we evaluate what each one becomes after training. A parent a indexes a checkpoint $\theta^a \in \mathcal{A}_k$. The fixed adaptation block turns it into a *provisional child* $\widetilde{\theta}_a = \Phi_k(\theta^a; \xi)$, where ξ collects training randomness. A development probe summarizes the child's new-task gain and earlier-task regression in a state y_k. A recovery action $r \in \mathcal{R}_k^{\mathrm{feas}}(y_k, a)$, namely continue, replay, or verified donor repair, then produces the *completed child*

$$\theta_{a,r}^{\mathrm{out}} = F_k(a, r; \xi) = \mathsf{R}_{k,r}(\widetilde{\theta}_a; \xi), \tag{2}$$

where F_k denotes adaptation followed by recovery. We call one executed parent–recovery pair a *branch*. A pair is feasible only if its measured resource vector $\mathsf{C}_k(a, r)$ stays within the stage budget B_k and both actions pass fixed feasibility masks.

Branch value. A completed branch improves on the incumbent by

$$G_k(a, r) = \mathsf{U}_k(\theta_{a,r}^{\mathrm{out}}) - \mathsf{U}_k(\theta_k^{\mathrm{inc}}) - \lambda^\top \mathsf{C}_k(a, r) - \sum_{j<k} \kappa_j[-\epsilon_j - \Delta_j]_+, \tag{3}$$

where $\lambda \ge 0$ prices resources and $\Delta_j = \mathsf{S}_j(\theta_{a,r}^{\mathrm{out}}) - \mathsf{S}_j(\theta_k^{\mathrm{inc}})$ is the score change on earlier task j, with tolerance ϵ_j and penalty weight κ_j. Utility already rewards retention; the last term adds a penalty only when a regression exceeds its tolerance. We write G_k for the realized value of an executed branch and

$$Q_k(a, r \mid x_k) = \mathbb{E}[G_k(a, r) \mid x_k, a, r] \tag{4}$$

for its expectation given the pre-adaptation state x_k, which contains only information available before a is chosen.

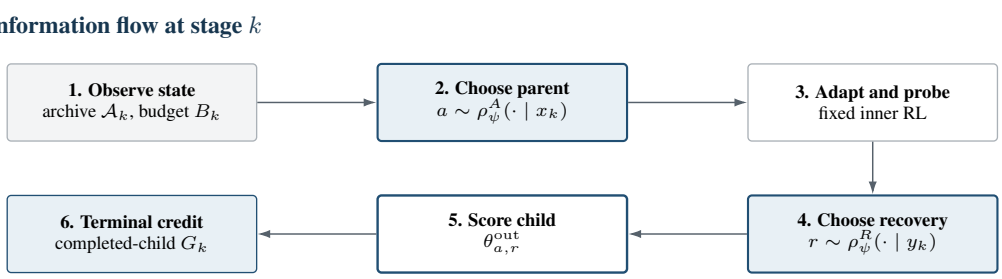


Figure 2: **One EvoLPO decision.** The archive and stage metadata determine a parent distribution. A common inner optimizer produces a provisional child. Only then does the controller observe measured learning, regression, and cost and choose recovery. The completed child supplies terminal credit; gradients do not pass through the inner optimizer.

Parent value depends on recovery. A parent must be chosen before y_k is observed, whereas recovery is chosen after. The value of a parent therefore depends on the rule $\pi(r \mid y_k, a)$ that will choose its recovery:

$$V^\pi_k(a \mid x_k) = \mathbb{E}_{y_k \mid x_k, a}\left[\sum_{r \in \mathcal{R}^{\text{feas}}_k(y_k, a)} \pi(r \mid y_k, a)\, \mathbb{E}[G_k(a, r) \mid x_k, a, y_k, r]\right]. \tag{5}$$

Parent selection needs V^π_k under the recovery rule that is actually deployed and different rules rank parents differently when the two decisions interact. If $Q_k(a, r) = f_k(a) + h_k(r)$ were additive, every rectangle of two parents and two recoveries would satisfy

$$I_k = Q_k(a_1, r_1) - Q_k(a_1, r_2) - Q_k(a_2, r_1) + Q_k(a_2, r_2) = 0. \tag{6}$$

A nonzero I_k shows interaction and a ranking reversal, where a_1 is better under r_1 but a_2 is better under r_2, shows that no parent is best independently of recovery (Section A.5).

Deployment. Not every completed child is deployed. A fixed transition T_k, whose incumbent component we write T^{inc}_k, applies archive admission, eviction and incumbent selection:

$$(\mathcal{A}_{k+1}, \theta^{\text{inc}}_{k+1}) = T_k(\mathcal{A}_k, \theta^{\text{out}}_{a,r}, \theta^{\text{inc}}_k), \qquad D_k(a, r) = \mathsf{U}_k(\theta^{\text{inc}}_{k+1}) - \mathsf{U}_k(\theta^{\text{inc}}_k). \tag{7}$$

We train the controller on the branch value G_k, which each stage observes directly and use D_k and end-to-end evaluation to test whether this credit improves the deployed policy.

4 Evolutionary Lineage Policy Optimization

Executing every parent–recovery pair would reveal V^π_k directly, but its cost grows with the archive size and the number of recovery actions. EvoLPO executes a single branch per stage and learns which one to execute. Figure 2 shows the information flow.

4.1 A two-stage controller

The controller has a parent head and a recovery head, which gives the trajectory factorization

$$p_\psi(a, y_k, r \mid x_k) = \rho^A_\psi(a \mid x_k)\, p_k(y_k \mid x_k, a)\, \rho^R_\psi(r \mid y_k, a), \tag{8}$$

where ψ are the controller parameters and p_k is the probe distribution induced by the fixed inner update, which does not depend on ψ. The pre-adaptation state x_k contains archive scores, lineage depth and age, uncertainty, task descriptors, the remaining budget and the results of short current-task probes run on disposable copies of each feasible parent. The post-adaptation state y_k adds the provisional child's new-task gain, earlier-task regression, verifier statistics and incurred cost. Both heads sample actions while the controller is fit and take their most likely action at deployment (Algorithm 1) and both receive the same credit G_k.

4.2 Recovery and archive updates

Recovery actions. Each branch copies its parent's weights, resets the optimizer and spends the same inner-training budget. After the probe, the remaining budget goes to one recovery action. *Continue* keeps training on the new task. *Replay* regenerates complete response groups for earlier-task prompts from the current child and scores them with the task verifier, so it stays on-policy. *Donor repair* selects the feasible archive member with the highest verified coverage, the share of earlier-task development prompts it solves and adds a supervised loss on its verifier-approved completions (Equation (34)). Matched controls fix attempted generations and accepted tokens (Section B.3).

Archive update. A completed child enters the archive only if paired-bootstrap bounds show nonnegative new-task gain and every earlier-task loss within tolerance. When the archive exceeds capacity M, the member that contributes least to the hypervolume of task scores is removed; the incumbent is never removed. A disjoint development panel then selects the next incumbent. These rules are fixed rather than learned (Section B.4).

4.3 Training the controller

Coverage data. The controller is fit on development streams collected with a randomized coverage policy whose parent and recovery distributions are μ^A and μ^R. For each covered provisional child, coverage forks the child and executes every feasible recovery, so record i with parent a and probe $y_i(a)$ contains the branch values $G_{i,a,r}$ of all feasible recoveries.

Recovery-aware parent target. Each record gives a one-sample estimate of the parent value V_k^π in Equation (5) for any recovery rule π:

$$\widehat{Z}_{\pi,i,a} = \sum_{r \in \mathcal{R}_k^{\mathrm{feas}}(y_i(a),a)} \pi(r \mid y_i(a), a)\, G_{i,a,r}. \tag{9}$$

The parent targets of Seq-2H and EvoLPO differ only in π. Seq-2H uses the coverage mix, $\pi = \mu^R$, which gives the target Z_μ. It can learn recovery preferences that depend on the parent, but it values each parent under a recovery mix that deployment never uses. EvoLPO first fits its recovery head out of fold and then uses that head's soft distribution, $\pi = \widetilde{\rho}_i^R$, which gives the target Z_ρ. The greedy target Z_{gr} instead puts all weight on the best recovery (Section I.8). Both Z_ρ and Z_{gr} are backward-induction targets, as in two-stage Q-learning (Murphy, 2003; 2005); our contribution is showing that the choice of target reorders parents in continual RLVR (Section 7) and estimating these targets from completed branches. Seq-2H and EvoLPO-S share traces, network size, masks and tuning budget, so their comparison isolates the parent target.

Two versions of EvoLPO. EvoLPO-*S* fits both heads by supervised regression on these targets; we use it whenever the controller is deployed on streams or model families excluded from its fitting. The full EvoLPO instead fits both heads in family with the policy gradient

$$\nabla_\psi J(\psi) = \mathbb{E}\left[\sum_{t \in \{A,R\}} \nabla_\psi \log \rho_\psi^t(z_t \mid s_t)\big(G_k - b_t(s_t)\big)\right], \tag{10}$$

where $(z_t, s_t) = (a, x_k)$ for $t = A$ and $(r, (y_k, a))$ for $t = R$. The inner update is part of the environment and the state-only baselines b_t add no bias (Proposition 1). Because both heads share G_k, the parent head is credited for the recovery that the current recovery head actually chooses. A matched control fits the same controller to the post-transition change D_k instead (Equation (27)), which tests whether the completed child is the right place to assign credit. Algorithm 1 summarizes the procedure.

5 Experiments

Tasks. We build three four-stage streams. The math stream moves from GSM8K (Cobbe et al., 2021) to disjoint MATH subjects (Hendrycks et al., 2021b). The code stream moves from APPS introductory to interview problems (Hendrycks et al., 2021a) and then to two PIE program-optimization families (Shypula et al., 2023). The mixed stream alternates mathematics and code. Mathematics uses strict

Algorithm 1 EvoLPO.

Input: development histories $\mathcal{H}_{\text{dev}}$; task stream $\mathcal{D}_{1:K}$; budgets $B_{1:K}$; archive capacity M.

Fit once. Collect feasible randomized traces $(x, a, y, r, G_k, \mathsf{C}_k)$ from development streams.
Fit the recovery policy $\rho^R(r \mid y, a)$ out of fold. For every held-out coverage record, form the parent target $\widehat{Z}_{\rho,i,a}$ of Equation (9) from its executed recovery branches.
Fit the parent policy $\rho^A(a \mid x)$ to these targets (EvoLPO-S), then freeze the controller, features, masks, budgets and archive rules. The full EvoLPO instead fits both decisions in family with Equation (10).
Deploy. Set $\theta_1^{\text{inc}} \leftarrow \theta_0$ and $\mathcal{A}_1 \leftarrow \{\theta_0\}$.
for $k = 1, \ldots, K$ **do**
    Probe disposable clones of feasible parents to form x_k; discard the clones and choose $\widehat{a} = \arg\max_{a \in \mathcal{A}_k} \rho^A(a \mid x_k)$.
    Reset and adapt the selected parent: $\widetilde{\theta}_{\widehat{a}} = \Phi_k(\theta^{\widehat{a}})$; probe the provisional child to form y_k.
    Choose $\widehat{r} = \arg\max_r \rho^R(r \mid y_k, \widehat{a})$ and execute it to obtain $\theta^{\text{out}}_{\widehat{a},\widehat{r}}$.
    Record G_k and cost. If the child passes admission, insert it; if capacity is exceeded, evict the smallest hypervolume contributor while protecting θ_k^{inc}.
    Select $\theta_{k+1}^{\text{inc}}$ from the resulting archive on the disjoint panel; break ties by creation stage, then checkpoint identifier (Section B.4).
end for; **return** $\theta_{K+1}^{\text{inc}}$. Sealed panels are used only for evaluation.

answer equivalence; code must compile and pass hidden tests and PIE rewards speedups only for correct programs. Section C.2 lists stage contents and data partitions.

Models. The main results use Qwen3-30B-A3B, a 30B-total/3B-active mixture-of-experts model, trained with LoRA (Hu et al., 2021). Dense Qwen3 models with 1.7B, 4B, 8B and 14B parameters form the scale study and five further families (Gemma 3, Llama 3.1, Mistral, Phi-4 Mini and OLMo 2) test transfer.

Baselines. We compare against sequential GRPO (Shao et al., 2024), on-policy prompt replay with and without optimizer resets, CPO-adapted (Luo et al., 2026b), historical short-trial selection (Li et al.; Nguyen et al., 2025), a matched TeLAPA-style selector (Lillo & Cheney, 2026), PBT-GRPO (Jaderberg et al., 2017), QD-GRPO (Mouret & Clune, 2015) and archive merges with model soups, task arithmetic and TIES (Wortsman et al., 2022; Ilharco et al., 2022; Yadav et al., 2023). Three controls isolate parts of EvoLPO: Seq-2H changes only the parent target, latest + matched repair forces the incumbent as parent while keeping the same recovery and archive rules and regression-threshold recovery chooses recovery with a fixed rule. All methods receive the same task data, update tokens, per-deployment accelerator budget, verifier schedule, probes and selection panel and failed attempts count toward cost (Sections F and H.3).

Protocol. Each comparison uses eight independent training runs, which we call roots, crossed with three stream families and four held-out task orders, for 96 paired deployments; intervals are clustered by root. All features, thresholds, archive rules, cost weights and controller parameters are frozen before held-out deployment and the controller never sees sealed outcomes or unexecuted branches. We report final score, $100\,\mathsf{U}_K$ of the final incumbent on sealed test data, together with new-task AUC and retention (Sections C.1 and F.2). Two comparisons isolate our contribution: EvoLPO-S versus Seq-2H under leave-one-stream-family-out (LOFO) fitting measures the parent target alone and EvoLPO versus Seq-2H in family measures the full system. Because the mixed stream contains both domains, LOFO tests transfer to a new stream composition, not to an unseen domain.

Generation. Reasoning mode is off and each completion is capped at 400 new tokens, truncated or unparseable outputs receive zero reward.

6 Results

Continual learning performance. Table 1 compares all methods on the four-stage streams. EvoLPO attains the highest final score and improves over Seq-2H by 1.79 points (95% CI $[1.17, 2.42]$). Figure 4(b) shows that this gain does not trade new-task learning against retention: EvoLPO improves both relative to replay and Seq-2H. The gain appears in mathematics, code and mixed streams and Seq-2H wins on two of the twelve held-out orders (Section H.1).

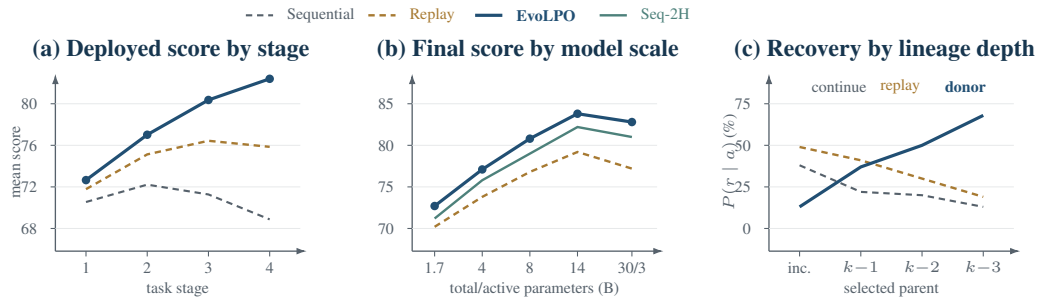


Figure 3: **Main empirical patterns.** **(a)** EvoLPO improves the single deployed policy across a four-stage mixed stream. **(b)** Its final-score advantage persists across the tested Qwen scale ladder; $30/3$ denotes the 30B-total/3B-active MoE. **(c)** Recovery changes with lineage depth: verified donor repair becomes more common for older parents. The corresponding numerical results appear in Tables 1 and 24 and Figure 6.

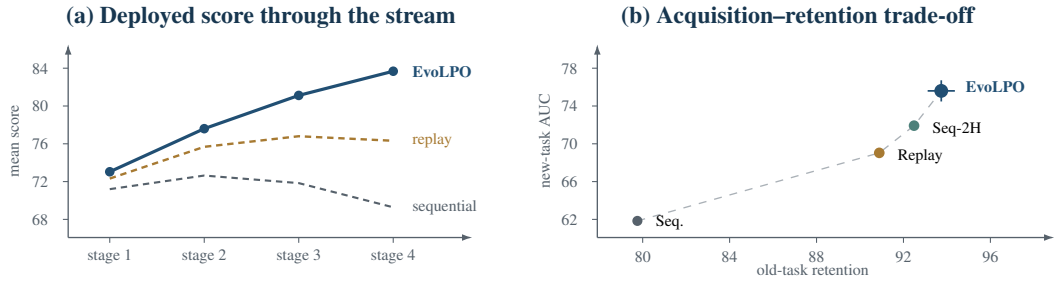


Figure 4: **Learning new tasks while retaining earlier ones.** (a) EvoLPO improves the single deployed policy across a four-stage mixed stream. (b) EvoLPO achieves both higher new-task acquisition and better earlier-task retention than replay and Seq-2H.

Table 1: **Overall Continual-Learning Performance.** Eight root clusters; $\pm$ is the 95% root-clustered interval half-width. Useful optimization is reported only for PIE boundaries. All rows use the same 62.5 H100-hour deployment budget, controller preparation is compared separately at equal total compute (Table 33).

Method	New AUC ↑	Ret. ↑	Final ↑	Useful opt. ↑
Sequential GRPO	61.8±0.62	79.6±0.65	69.4±0.57	18.7±1.20
Replay-GRPO	68.9±0.73	89.8±0.48	77.2±0.47	24.9±0.87
Replay-GRPO + reset	69.6±0.70	90.2±0.47	77.9±0.35	25.8±1.08
CPO-adapted	68.1±0.53	90.5±0.45	77.0±0.60	24.2±1.17
PBT-GRPO	70.1±0.65	88.7±0.45	78.1±0.52	27.2±0.87
QD-GRPO	70.6±0.47	89.1±0.51	78.7±0.31	27.8±0.89
Historical short trial	71.2±0.68	89.4±0.53	79.0±0.34	28.1±1.05
Latest + matched repair	70.5±0.63	91.0±0.48	79.8±1.51	29.4±1.23
TIES archive merge	71.8±0.68	91.4±0.58	80.9±0.44	30.7±0.97
Seq-2H	72.0±0.60	91.5±0.43	81.0±0.59	31.2±0.86
EvoLPO	**73.7±0.62**	**92.6±0.62**	**82.8±0.91**	**34.1±1.14**

Access to older checkpoints. Forcing the same system to resume from the latest checkpoint costs 3.01 points, $[2.15, 3.88]$. This control changes both the parent choice and the options available later in the stream, so it measures the value of the archive itself.

Comparison with checkpoint selection and merging. Scoring a parent after recovery matters more than scoring it after adaptation. Historical short-trial selection, which probes each archived parent with continued training, reaches 79.0 and the matched TeLAPA-style selector reaches 80.7, 2.1 points

Table 2: **A decision-changing parent–recovery interaction.** Mean completed-child utility over four roots. The coverage weights are $\mu^R = (.15, .70, .15)$; Z_μ and Z_{gr} select the bold parents.

Parent	Continue	Replay	Donor	Z_μ	Z_{gr}
Incumbent	0.774	0.803	0.812	0.800	0.812
History $k-1$	0.781	0.794	0.826	0.797	0.826
History $k-2$	0.768	0.821	0.807	**0.811**	0.821
History $k-3$	0.789	0.785	0.835	0.793	**0.835**

below EvoLPO at the same verifier cost (Table 32). Merging the archive instead of selecting from it is also weaker: TIES, the strongest merge, reaches 80.9 (Table 17).

Model scale and families. Figure 3(b) shows that the advantage over Seq-2H holds across the Qwen scale ladder, from the 1.7B dense model to the 30B-total/3B-active MoE (Table 24). EvoLPO-S, fit once on Qwen3-8B and then frozen, transfers to five unseen model families and improves over Seq-2H by 0.5–0.9 points on each (Table 36).

7 Analysis and Ablations

Recovery reorders parents. Table 2 shows one development state from the mechanism study. The coverage target Z_μ used by Seq-2H selects history $k-2$, because replay receives most of the coverage mass. The greedy target Z_{gr} instead selects history $k-3$ with donor repair and raises completed-child utility from 0.821 to 0.835. The replay–donor rectangle for these two parents has interaction $I_k = 0.064$, or 0.065 after cost and regression penalties (Table 30). Such reversals are common: 4, 3 and 7 of 18 rectangles qualify in the math, code and mixed matrices (Table 5). The deployed controller acts on them, choosing donor repair more often as the selected parent gets older (Figure 3(c)).

An example decision. Section D.3 follows six logged decisions. At one MATH-algebra boundary, the parent head assigns probability 0.51 to history $k-2$. Its provisional child raises algebra accuracy from .42 to .74 but lowers the earlier-task mean from .812 to .754, with the largest drop on number theory. The recovery head then assigns 0.72 to donor repair, which keeps algebra at .72 and restores retention to .90; under continued training, the same parent would have been infeasible. Admission guards against the opposite case: when recovery cannot bring an earlier-task loss back within tolerance, the child is rejected and the incumbent stays deployed.

Effect of the parent target. Changing only the parent target, EvoLPO-S improves final score over Seq-2H under LOFO by 0.70 points, $[0.10, 1.30]$. Selection quality moves in the same direction: EvoLPO-S recovers 69% of oracle opportunities versus 61% for Seq-2H and lowers selection regret from 1.4 to 1.1 (Table 15). In family, policy-gradient fitting adds a further 1.1 points (Table 12).

Where to assign credit. Crediting the parent on the provisional child, before recovery, reduces final score by 2.5 points (Table 16). Crediting it after the archive transition, with D_k instead of G_k, lowers final score by 0.38 points, because D_k maps several different children to the same incumbent.

Archive size and eviction. In an eight-stage stress test where capacity binds after stage 4, a four-member archive with hypervolume eviction scores 81.9, close to 82.1 with no eviction at $M = 8$, while oldest-first eviction drops to 81.2 and a single-member archive to 78.9 (Table 19). Exposing more candidates to incumbent selection helps up to the registered four (79.8 with one, 82.8 with four) and no further (82.7 with eight; Table 18).

Robustness. Table 3 collects the paired gain over Seq-2H under changes to the stream, data partition, inner optimizer, utility, verifier, model family and generation budget. The gain remains positive on eight- and twelve-stage streams (Table 38), across five independent PIE problem partitions (Section I.9) and when TROLL replaces GRPO's clipped update (Table 23). It does not depend on the utility definition: under recency-weighted tasks or 0.5–1.5× tolerances and costs, the gain stays positive and the same parent is chosen at 84–96% of boundaries (Table 31). It degrades gracefully with verifier noise, staying positive through 10% flipped verifier outcomes, while its interval includes

Table 3: **Robustness of the gain over Seq-2H.** Paired final-score difference with root-clustered 95% interval. Rows above the line use the full EvoLPO in family; rows below use EvoLPO-S, which transfers without refitting. The 2,048-token row is a separate cohort rather than a cap-only ablation (Section I.4).

Condition	Change	Δ final	95% CI
Registered protocol	four-stage streams	1.79	$[1.17, 2.42]$
Longer streams	$K = 8$	2.1	$[1.3, 2.9]$
	$K = 12$	2.4	$[1.5, 3.3]$
PIE data partition	five reseeded splits	1.5	$[1.0, 2.0]$
Inner optimizer	TROLL projection	1.6	$[0.8, 2.4]$
Utility definition	recency-weighted tasks	1.58	$[0.86, 2.30]$
Verifier noise	10% flipped outcomes	1.18	$[0.42, 1.94]$
	20% flipped outcomes	0.46	$[-0.35, 1.27]$
Held-out stream composition	LOFO	0.70	$[0.10, 1.30]$
Unseen model families	five families	0.5–0.9	each excludes 0
Generation budget	2,048-token cohort	0.46	$[0.06, 0.86]$

zero at 20% as the controller chooses donor repair less often (41% to 28%; Table 37). A greedy recovery target performs like the soft target (-0.1 points; Section I.8).

Compute. Table 1 matches only the 62.5 H100-hour deployment budget and does not charge controller preparation (1,920 hours for Seq-2H, 2,280 for EvoLPO). We therefore also compare at equal total compute, where replay receives the full allowance, spends 48 hours tuning its replay ratio, KL weight and update schedule and uses the rest for additional training. This tuned replay is far stronger than the replay rows of Table 1 (82.09 vs. 77.2–77.9), yet EvoLPO still leads, 82.83 to 82.09, a gain of 0.74 points, $[0.18, 1.30]$. Seq-2H remains our main contrast because it isolates the parent target; tuned replay is the strongest system baseline. With half of the coverage data, preparation takes 930 fewer H100-hours and EvoLPO still beats tuned replay; with a quarter, the interval includes zero (Section I.4). Per deployment, EvoLPO uses $1.24\times$ the reference verifier time, compared with $1.09\times$ for replay (Table 20).

Generation budget. In a separate cohort with a 2,048-token limit, truncation is below 4% (about 18% at 400 tokens) and the parent target still improves final score (Tables 3 and 35).

8 Conclusion

We introduce EvoLPO, a controller that chooses where to resume continual RLVR by predicting the outcome after both adaptation and recovery. Its parent head is trained on completed-branch outcomes weighted by the learned recovery policy, so each parent is valued for the recovery it will actually receive. Across mathematics, code and program-optimization streams, the Qwen scale ladder and five further model families, EvoLPO improves the deployed policy over a two-head controller that values parents under the data-collection recovery mix. In the main setting, it also exceeds sequential training and, at equal total compute, tuned replay. The gain comes from states where archived checkpoints adapt differently and recovery changes their ranking; when incumbent replay attains the best new-task score within every retention tolerance, the archive adds nothing.

Limitations and future work. The parent target alone improves final score by 0.70 points under LOFO and most of the in-family gain comes from policy-gradient fitting, which needs about 14 reused deployments to repay its preparation (Section H.3). EvoLPO also costs more than replay per deployment, so its equal-compute advantage depends on sufficient coverage data. Our experiments use visible task boundaries, fixed verifiers, low-rank updates and non-reasoning generation with short completions. Extending recovery-aware selection to reasoning models, unknown task boundaries and cheaper coverage are natural next steps.

REPRODUCIBILITY STATEMENT

All models and datasets are public and pinned by repository revision (Sections D.1 and H.4). Sections C.1 and C.2 give the full protocol and data partitions, Section B gives the controller and training details and Section H.1 reports the root-level effects behind every headline interval. The supplementary material contains the controller implementation.The full training and evaluation code will be released upon acceptance.

ETHICS STATEMENT

EVOLPO makes continual fine-tuning of language models on verifiable mathematics and code tasks more effective. We use only publicly available models and datasets. As with any advance in LLM training, the benefits and risks of this work depend on how the resulting models are deployed.

AI USE STATEMENT

We used LLMs to assist with revising grammar, style and text flow in this manuscript and to help write figure code.

REFERENCES

Philipp Becker, Niklas Freymuth, Serge Thilges, Fabian Otto, and Gerhard Neumann. Troll: Trust regions improve reinforcement learning for large language models. In *International Conference on Learning Representations*, volume 2026, pp. 53096–53135, 2026.

Glen Berseth, Zhiwei Zhang, Grace Zhang, Chelsea Finn, and Sergey Levine. Comps: Continual meta policy search. *arXiv preprint arXiv:2112.04467*, 2021.

Bibhas Chakraborty and Erica E. M. Moodie. Statistical methods for dynamic treatment regimes: Reinforcement learning, causal inference, and personalized medicine. 2013. URL `https://api.semanticscholar.org/CorpusID:126201516`.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Tanmay Gangwani and Jian Peng. Policy optimization by genetic distillation. *arXiv preprint arXiv:1711.01012*, 2017.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021b.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Jinwu Hu, Zihao Lian, Zhiquan Wen, Chenghao Li, Guohao Chen, Xutao Wen, Bin Xiao, and Mingkui Tan. Continual knowledge adaptation for reinforcement learning. *Advances in Neural Information Processing Systems*, 38:36283–36313, 2026.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Anthony Kobanda, Rémy Portelas, Odalric-Ambrym Maillard, and Ludovic Denoyer. Hierarchical subspaces of policies for continual offline reinforcement learning. *arXiv preprint arXiv:2412.14865*, 2024.

Siyuan Li, Hao Li, Jin Zhang, Zhen Wang, Peng Liu, and Chongjie Zhang. Iob: integrating optimization transfer and behavior transfer for multi-policy reuse. *Autonomous Agents and Multi-Agent Systems*, 38(1):3, 2024.

Xinran Li, Guangda Huzhang, Siqi Shen, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Jun Zhang. Getting your llms ready for reinforcement learning with lightweight sft. In *The Fourteenth International Conference on Learning Representations*.

Lute Lillo and Nick Cheney. Beyond single-model optimization: Preserving plasticity in continual reinforcement learning. *arXiv preprint arXiv:2604.15414*, 2026.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Lirui Luo, Guoxi Zhang, Hongming Xu, Rongqing Li, Cong Fang, and Lifeng Fan. Continual reasoning gym: Diagnosing and harnessing shared reasoning in continual rlvr. *arXiv preprint arXiv:2608.18574*, 2026a.

Mao-Lin Luo, Zhe-Xu Wang, Zi-Hao Zhou, Bo Ye, Jian Zhao, Min-Ling Zhang, and Tong Wei. Rl forgets! towards continual policy optimization. *arXiv preprint arXiv:2607.04364*, 2026b.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

S. A. Murphy. Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 65(2):331–355, 2003. doi: 10.1111/1467-9868.00389. URL https://doi.org/10.1111/1467-9868.00389.

Susan A. Murphy. A generalization error for q-learning. *Journal of Machine Learning Research*, 6 (37):1073–1097, 2005. URL http://jmlr.org/papers/v6/murphy05a.html.

Manh Nguyen, Dung Nguyen, Dai Do, Svetha Venkatesh, and Hung Le. Uncertainty-guided checkpoint selection for reinforcement finetuning of large language models. *arXiv preprint arXiv:2511.09864*, 2025.

Chuanyu Qin, Chenxu Yang, Qingyi Si, Naibin Gu, Peng Fu, and Zheng Lin. Learning to solve, forgetting to retain: Correct-set turnover in rlvr. *arXiv preprint arXiv:2606.03087*, 2026.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pp. 4528–4537. PMLR, 2018.

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

Velizar Shulev and Khalil Sima'an. Continual reinforcement learning for controlled text generation. In *International Conference on Language Resources and Evaluation*, 2024. URL https://api.semanticscholar.org/CorpusID:269804252.

Alexander Shypula, Aman Madaan, Yimeng Zeng, Uri Alon, Jacob Gardner, Milad Hashemi, Graham Neubig, Parthasarathy Ranganathan, Osbert Bastani, and Amir Yazdanbakhsh. Learning performance-improving code edits. *arXiv preprint arXiv:2302.07867*, 2023.

Matthew Siper, Ahmed Khalifa, and Julian Togelius. The time value of evolution. *arXiv preprint arXiv:2608.13297*, 2026.

Chang Su, Zhongkai Hao, Zhizhou Zhang, Zeyu Xia, Youjia Wu, Hang Su, and Jun Zhu. Helix: Evolutionary reinforcement learning for open-ended scientific problem solving. *arXiv preprint arXiv:2603.07642*, 2026.

Rishabh Tiwari, Kusha Sareen, Lakshya A Agrawal, Joseph E Gonzalez, Matei Zaharia, Kurt Keutzer, Inderjit S Dhillon, Rishabh Agarwal, and Devvrit Khatri. Learning, fast and slow: Towards llms that adapt continually. *arXiv preprint arXiv:2605.12484*, 2026.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Gyeongtae Yoo, Sanghyeok Park, Soohyuk Jang, Ik-hwan Kim, and Sungroh Yoon. Rollout-level advantage-prioritized experience replay for grpo. *arXiv preprint arXiv:2606.04560*, 2026.

Han Zhang, Yu Lei, Lin Gui, Min Yang, Yulan He, Hui Wang, and Ruifeng Xu. Cppo: Continual learning for reinforcement learning with human feedback. In *International Conference on Learning Representations*, 2024. URL https://api.semanticscholar.org/CorpusID:271746234.

A FORMALIZATION

A.1 NOTATION

Random variables are defined on the probability space induced by the task stream, model sampling, optimization, verification, evaluation and controller actions. The symbol $\mathbb{E}[\cdot]$ denotes expectation on this space; a conditional expectation lists the information fixed after the vertical bar. Every action set is finite.

Stages, tasks and policies. The stream contains K stages. Index $k \in \{1, \dots, K\}$ denotes the arriving stage, while $j \le k$ denotes any task encountered by that stage. The distribution of task j is $\mathcal{D}_j$. A parameter vector is θ and $\pi_\theta(\cdot \mid q)$ is the language-model policy over completions of prompt q. The single policy deployed before stage k is θ_k^{inc}. The bounded archive is $\mathcal{A}_k = \{\theta_k^1, \dots, \theta_k^{M_k}\}$, where $M_k \le M$ and M is the fixed capacity. A parent action $a \in \{1, \dots, M_k\}$ indexes checkpoint θ^a; we use "parent a" as shorthand for that checkpoint.

Adaptation and recovery. The fixed inner update for stage k is the stochastic operator Φ_k. It maps θ^a to provisional child $\widetilde{\theta}_a = \Phi_k(\theta^a; \xi)$, where ξ collects all rollout, minibatch, optimizer and decoding randomness not already conditioned upon. The calligraphic set $\mathcal{R}_k$ contains the recovery actions: continue, replay and verified donor repair. The roman operator $\mathsf{R}_{k,r}$ applies action $r \in \mathcal{R}_k$ to a provisional child. Their composition is $F_k(a, r; \xi) = \mathsf{R}_{k,r}(\Phi_k(\theta^a; \xi); \xi)$ and the completed child is $\theta_{a,r}^{\mathrm{out}} = F_k(a, r; \xi)$. The scalar G_k is the terminal controller reward.

Scores, utility and retention. The population score of policy θ on task j is $\mathsf{S}_j(\theta) \in [0, 1]$. The encountered-task utility $\mathsf{U}_k(\theta) = \sum_{j \le k} w_{k,j} \mathsf{S}_j(\theta)$ uses frozen weights $w_{k,j} \ge 0$ that sum to one. Tables report $100\mathsf{U}_k$ or the stated component metric, so a one-point table difference means 0.01 on the unit interval. For an earlier task $j < k$, $\Delta_j = \mathsf{S}_j(\theta_{a,r}^{\mathrm{out}}) - \mathsf{S}_j(\theta_k^{\mathrm{inc}})$ is the signed score change: negative values indicate forgetting. The tolerance $\epsilon_j \ge 0$ permits regression up to ϵ_j, $\kappa_j \ge 0$ weights excess regression and $[u]_+ = \max\{u, 0\}$ is the positive-part operator. Table 8 gives the registered numerical values.

Resources and feasibility. The vector $\mathsf{C}_k(a, r) \in \mathbb{R}_+^d$ records the prespecified resource coordinates for a completed branch and $B_k \in \mathbb{R}_+^d$ is the stage budget. The relation $\mathsf{C}_k(a, r) \preceq B_k$ means coordinatewise inequality. The nonnegative vector $\lambda \in \mathbb{R}_+^d$ converts measured resources into the scalar penalty $\lambda^\top \mathsf{C}_k(a, r)$. Feasible parent and recovery sets are formed before sampling from information then available; masked actions receive probability zero. Failed executed actions are outcomes, not missing data: they receive the frozen bounded failure reward and retain their consumed cost.

Information states and values. The pre-adaptation state x_k is measurable with respect to information $\mathcal{F}_k$ available before choosing a. After adaptation, the information expands to $\mathcal{G}_k$ and y_k is the post-adaptation state. We write $y(a)$ when emphasizing that the observed probe depends on the chosen parent. The pair value $Q_k(a, r \mid x_k)$ is the expected cost-penalized improvement of the completed child. In the controller appendix, lowercase $q(x, a, y, r) = \mathbb{E}[G_k \mid x, a, y, r]$ denotes the more finely conditioned post-probe value. Averaging q under a recovery rule π gives the parent value V_k^π of Equation (5); the targets Z_μ and Z_ρ are this value under the coverage recovery policy μ^R and the learned soft recovery distribution ρ^R, respectively. Deployment instead executes the greedy mode of ρ^R; its backup is treated separately.

Controllers and estimators. The learned controller has parent factor $\rho_\psi^A(a \mid x)$ and recovery factor $\rho_\psi^R(r \mid y, a)$, with parameters ψ. Superscript A denotes archive-parent selection and superscript R denotes recovery. The randomized coverage controller is ρ_ω in the policy-gradient derivation; μ denotes its recovery distribution in the supervised-backup comparison. The Seq-2H parent and recovery regressors are S_ϕ^A and S_η^R; the archive encoder is ϕ_ζ, the learned-policy-weighted parent regressor is H_χ and the fitted recovery policy is ρ_ν^R. A hat denotes a finite-sample estimate, a bar denotes an empirical average, $\mathbf{1}\{E\}$ is the indicator of event E and $\arg\max$ may return any maximizer under the fixed deterministic tie rule.

Remaining operators. $\mathrm{KL}(p\|q)$ is Kullback–Leibler divergence, Δ^{V-1} is the probability simplex over a vocabulary of size V and HV is hypervolume relative to the frozen reference point. All confidence bounds, temperatures, thresholds, tie rules, normalizers and cost weights are fitted or fixed on development histories before prospective evaluation.

A.2 Information available at a task boundary

At stage k, task distribution $\mathcal{D}_k$ arrives while θ_k^{inc} is deployed. The bounded archive $\mathcal{A}_k$ contains at most M realized weight histories. The controller first chooses parent a, the fixed operator Φ_k produces a provisional child and recovery action r indexes $\mathsf{R}_{k,r}$. The completed child is $\theta_{a,r}^{\mathrm{out}}$. Task score S_j, encountered-task utility U_k, resource vector C_k, expected pair value Q_k and sampled terminal reward G_k are defined in Equations (1), (3) and (4). A hat always denotes a finite-sample quantity available to the controller; an unmarked score denotes its population target.

At stage k, the observable sigma-field $\mathcal{F}_k$ contains training and controller-development samples from tasks $1{:}k$, archived parameters, logged training histories, resource measurements and controller randomness revealed before the first action. It excludes the disjoint incumbent-selection panel, all final-evaluation samples, future-task identities and labels and outcomes of unexecuted counterfactual branches. The pre-adaptation state x_k is $\mathcal{F}_k$-measurable. After sampling a and running Φ_k, the sigma-field expands to $\mathcal{G}_k \supset \mathcal{F}_k$ with adaptation cost and controller-development measurements; y_k is $\mathcal{G}_k$-measurable. The recovery action is sampled only after this expansion. The incumbent-selection panel is revealed only after all candidates and controller actions for the stage are fixed.

This ordering defines the prediction problem. Post-adaptation development measurements are available when recovery is chosen. Final evaluation scores and outcomes of unexecuted branches arrive afterward and never enter the controller state. Archive metadata contains measurements recorded when the checkpoint was created, together with current disposable-probe features.

A.3 Credit assignment to completed children

We begin from the probability of a complete two-stage branch. This derivation fixes the reward assigned to the parent and recovery decisions when adaptation, verification and branch completion are stochastic.

Let a trace be

$$\tau = (x, a, \widetilde{\theta}, y, r, \theta^{\mathrm{out}}, T, G),$$

where $T \in \{0, 1\}$ records successful completion and G is the terminal reward G_k of Equation (3). A failed branch receives a prespecified bounded failure reward and retains its consumed cost. Let $p^{\mathrm{ad}}(\widetilde{\theta}, y \mid x, a)$ denote the distribution induced by adaptation and measurement (its y-marginal is p_k in Equation (8)) and let $p^{\mathrm{rec}}(\theta^{\mathrm{out}}, T, G \mid x, a, \widetilde{\theta}, y, r)$ denote the distribution induced by recovery and evaluation. The controller-dependent trace density is

$$\begin{aligned} p_\psi(\tau) = p(x)\rho_\psi^A(a \mid x)p^{\mathrm{ad}}(\widetilde{\theta}, y \mid x, a) \\ \cdot \rho_\psi^R(r \mid y, a)p^{\mathrm{rec}}(\theta^{\mathrm{out}}, T, G \mid x, a, \widetilde{\theta}, y, r). \end{aligned} \tag{11}$$

The last kernel includes every stochastic operation after recovery is chosen, including donor decoding, verifier outcomes, the recovery update, terminal evaluation and branch failure. Conditioning on $\widetilde{\theta}$ makes clear that recovery acts on the realized provisional child, not merely on its low-dimensional probe y.

Proposition 1 (Valid outer credit assignment)**.** *Assume (i) $\mathbb{E}_{p_\psi}|G| < \infty$; (ii) the support of each controller distribution is fixed locally in ψ after applying a state-dependent feasibility mask; (iii) differentiation and integration can be exchanged; (iv) the environment kernels p^{ad} and p^{rec} have no direct dependence on ψ once their sampled actions are conditioned upon; and (v) baselines $b_A(x)$ and $b_R(y, a)$ do not depend on the corresponding sampled action. Then*

$$\begin{aligned} \nabla_\psi \mathbb{E}_{p_\psi}[G] = \mathbb{E}_{p_\psi}\big[\nabla_\psi \log \rho_\psi^A(a \mid x)(G - b_A(x)) \\ + \nabla_\psi \log \rho_\psi^R(r \mid y, a)(G - b_R(y, a))\big]. \end{aligned} \tag{12}$$

Proof. Define $J(\psi) = \mathbb{E}_{p_\psi}[G]$. We proceed in four steps.

Step 1: differentiate the trace expectation. Writing the expectation as an integral over continuous trace components and a sum over the finite actions, assumption (iii) gives

$$\nabla_\psi J(\psi) = \int G(\tau) \nabla_\psi p_\psi(\tau)\, d\tau.$$

Where $p_\psi(\tau) > 0$, the log-derivative identity $\nabla_\psi p_\psi = p_\psi \nabla_\psi \log p_\psi$ yields

$$\nabla_\psi J(\psi) = \mathbb{E}_{p_\psi}\left[G \nabla_\psi \log p_\psi(\tau)\right].$$

The locally fixed-support assumption avoids boundary terms from actions entering or leaving the mask while differentiating.

Step 2: identify the controller-dependent factors. Taking logs of Equation (11) turns its product into a sum. By assumption (iv), $p(x)$, p^{ad} and p^{rec} have zero direct derivative with respect to ψ. Hence

$$\nabla_\psi \log p_\psi(\tau) = \nabla_\psi \log \rho^A_\psi(a \mid x) + \nabla_\psi \log \rho^R_\psi(r \mid y, a).$$

Substitution already gives an unbiased score-function gradient in which both actions receive the same completed-child reward.

Step 3: subtract the parent baseline. Conditioning on x and summing over the feasible parent set,

$$\begin{aligned}
\mathbb{E}[b_A(x) \nabla_\psi \log \rho^A_\psi(a \mid x) \mid x] &= b_A(x) \sum_a \rho^A_\psi(a \mid x) \nabla_\psi \log \rho^A_\psi(a \mid x) \\
&= b_A(x) \sum_a \nabla_\psi \rho^A_\psi(a \mid x) \\
&= b_A(x) \nabla_\psi 1 = 0.
\end{aligned}$$

Step 4: subtract the recovery baseline. Condition on (y, a) and repeat the same calculation:

$$\begin{aligned}
&\mathbb{E}[b_R(y, a) \nabla_\psi \log \rho^R_\psi(r \mid y, a) \mid y, a] \\
&\quad = b_R(y, a) \nabla_\psi \sum_r \rho^R_\psi(r \mid y, a) = 0.
\end{aligned}$$

Subtracting both zero-expectation terms gives Equation (12). The completion indicator T and all failure outcomes already lie inside p^{rec} and G, so stochastic termination introduces no additional score term. □

Interpretation. Both controller actions receive the reward of the completed branch. Randomized branch coverage supplies comparisons among recovery actions and logged-data fitting uses support checks and importance ratios for both decisions. The gradient therefore optimizes the branch distribution defined in Equation (11); the end-to-end experiments separately measure the incumbent produced by the archive transition.

A.4 Off-policy estimation from coverage traces

Suppose traces are collected by behavior controller ρ_ω but the objective concerns target controller ρ_ψ. Define the two action ratios and their product by

$$w_A(\tau) = \frac{\rho^A_\psi(a \mid x)}{\rho^A_\omega(a \mid x)}, \qquad w_R(\tau) = \frac{\rho^R_\psi(r \mid y, a)}{\rho^R_\omega(r \mid y, a)}, \qquad w(\tau) = w_A(\tau) w_R(\tau).$$

Coverage means that $\rho^A_\omega(a \mid x) > 0$ whenever $\rho^A_\psi(a \mid x) > 0$ and likewise $\rho^R_\omega(r \mid y, a) > 0$ whenever $\rho^R_\psi(r \mid y, a) > 0$. Under this support condition and the assumptions of Proposition 1, changing measure from p_ω to p_ψ gives

$$\begin{aligned}
\nabla_\psi J(\psi) = \mathbb{E}_{p_\omega}\Big[w(\tau) \Big(&\nabla_\psi \log \rho^A_\psi(a \mid x)(G - b_A(x)) \\
&+ \nabla_\psi \log \rho^R_\psi(r \mid y, a)(G - b_R(y, a)) \Big) \Big].
\end{aligned} \tag{13}$$

To see the change of measure, divide the target trace density in Equation (11) by the behavior trace density. The environment factors $p(x)$, p^{ad} and p^{rec} cancel, leaving exactly $w_A w_R$. Therefore, for any integrable trace function h, $\mathbb{E}_{p_\psi}[h] = \mathbb{E}_{p_\omega}[wh]$; applying this identity to the on-policy score-function integrand proves Equation (13).

For n logged traces, the direct estimator replaces the expectation by $n^{-1}\sum_{i=1}^{n}$. Clipping uses $\bar{w}_i = \min\{w_i, c\}$ for fixed $c > 0$; self-normalization uses $\bar{w}_i / \sum_l \bar{w}_l$. These variants trade bias for lower variance. Every target action must remain inside behavior support, so controller development begins with randomized coverage of the feasible parent–recovery pairs.

A.5 When recovery changes parent choice

Fix a state and suppress k and x. Let $Q : \mathcal{A} \times \mathcal{R} \to \mathbb{R}$ be the expected terminal value after complete recovery.

Proposition 2 (Rectangle test for nonadditivity)**.** *If $Q(a, r) = f(a) + h(r)$ for all (a, r), then for every a_1, a_2, r_1, r_2,*

$$I = Q(a_1, r_1) - Q(a_1, r_2) - Q(a_2, r_1) + Q(a_2, r_2) = 0.$$

Conversely, $I \neq 0$ on a tested rectangle rules out an additive representation on that rectangle.

Proof. Substituting $f(a) + h(r)$ gives

$$\begin{aligned} I = {} & f(a_1) + h(r_1) - f(a_1) - h(r_2) \\ & - f(a_2) - h(r_1) + f(a_2) + h(r_2) = 0. \end{aligned}$$

The converse follows by contradiction: an additive representation on the rectangle would force $I = 0$ by the same cancellation. □

Interaction and decision relevance. Consider

$$Q = \begin{pmatrix} .90 & .88 \\ .70 & .60 \end{pmatrix}.$$

Here $I = -.08 \neq 0$, while the first parent and first recovery action remain optimal under both joint and marginal rankings. The interaction leaves the selected action unchanged. By contrast,

$$Q = \begin{pmatrix} .90 & .62 \\ .72 & .94 \end{pmatrix}$$

contains a ranking reversal: the preferred parent changes with recovery. The experiments compare joint selection with three operational restrictions: an additive predictor, a fixed-recovery parent selector and a myopic selector whose first decision precedes the recovery observation.

Regret without a ranking reversal. Consider

$$Q = \begin{pmatrix} .90 & .80 \\ .10 & .70 \end{pmatrix}.$$

The first parent is better under both recoveries. A uniformly fitted least-squares additive approximation still ranks the second recovery higher because its column mean is .75 instead of .50; it obtains .80, whereas joint maximization obtains .90. We use ranking reversals to expose the interaction and held-out selection regret to measure its decision cost.

Definition 1 (Net joint decision benefit)**.** *For a joint selector δ_J, restricted selector δ_R and state distribution ν, define*

$$\Delta_{J:R} = \mathbb{E}_{x\sim\nu}\left[Q_x(\delta_J(x)) - Q_x(\delta_R(x)) - c_J(x) + c_R(x)\right],$$

where c includes feature measurement, branch probes, fitting and inference. Joint selection is operationally useful only if a lower uncertainty bound for $\Delta_{J:R}$ exceeds a prespecified practical threshold on fresh states.

A.6 PREDICTION ERROR AND SELECTION REGRET

Proposition 3 (Finite-action greedy selection)**.** *Let $\mathcal{Z}$ be a finite feasible set, $z^* \in \arg\max_z Q(z)$ and $\widehat{z} \in \arg\max_z \widehat{Q}(z)$. If $\max_{z\in\mathcal{Z}} |\widehat{Q}(z) - Q(z)| \le \eta$, then*

$$Q(z^*) - Q(\widehat{z}) \le 2\eta.$$

Proof. The uniform-error event gives two inequalities for every action z: $Q(z) \le \widehat{Q}(z) + \eta$ and $\widehat{Q}(z) \le Q(z) + \eta$. Apply the first to the true optimizer z^*:

$$Q(z^*) \le \widehat{Q}(z^*) + \eta.$$

Because $\widehat{z}$ maximizes $\widehat{Q}$, we also have $\widehat{Q}(z^*) \le \widehat{Q}(\widehat{z})$. Finally, apply the second inequality to $\widehat{z}$: $\widehat{Q}(\widehat{z}) \le Q(\widehat{z}) + \eta$. Chaining the three statements gives

$$Q(z^*) \le \widehat{Q}(z^*) + \eta \le \widehat{Q}(\widehat{z}) + \eta \le Q(\widehat{z}) + 2\eta.$$

Subtracting $Q(\widehat{z})$ from both sides proves $Q(z^*) - Q(\widehat{z}) \le 2\eta$. The factor two reflects one possible estimation error at the true maximizer and one at the selected maximizer. □

High-probability specialization. Suppose each terminal value lies in a known interval $[L, U]$ of width $W = U - L$ and $Q(z)$ is estimated by the mean of n_z independent outcomes. In our implementation this interval follows from bounded scores, clipped cost coordinates and the fixed failure reward. Hoeffding's inequality and a union bound give

$$\Pr\Big(\max_z |\widehat{Q}(z) - Q(z)| > \eta\Big) \le 2\sum_{z\in\mathcal{Z}} \exp(-2n_z\eta^2/W^2).$$

For equal n, a sufficient condition for regret at most 2η with probability at least $1 - \delta$ is

$$n \ge \frac{W^2}{2\eta^2} \log \frac{2|\mathcal{Z}|}{\delta}.$$

Indeed, with $n_z = n$, the preceding probability is at most $2|\mathcal{Z}|e^{-2n\eta^2/W^2}$. Requiring this expression to be at most δ, taking logarithms and solving for n yields the displayed inequality. On the complementary event, Proposition 3 converts the uniform value error η into regret at most 2η. Branch outcomes share parents, roots, prompts and task orders, so the experiment treats independent training roots as the sampling unit and uses paired or hierarchical uncertainty estimates. The bound above characterizes value-estimation error rather than sample-size selection.

Cost-sensitive selection. Let $U(z)$ denote the expected completed-child improvement before resource penalization and let $V(z) = U(z) - \lambda^\top c(z)$. For estimates with $|\widehat{U}(z) - U(z)| \le \eta_U$ and $\|\widehat{c}(z) - c(z)\|_\infty \le \eta_c$, greedy selection by $\widehat{V}$ has regret at most $2(\eta_U + \|\lambda\|_1\eta_c)$. This is a direct application of Proposition 3; cost is subtracted once, rather than from the already cost-penalized Q in Equation (4). It emphasizes that uncertain cost estimates can reverse a small apparent quality advantage.

A.7 CONSTRAINTS AND RETENTION DIAGNOSTICS

The scalar reward in Equation (3) is convenient for controller training but is not the only scientific target. The constrained form is

$$\max_{a,r} \mathbb{E}[\mathsf{S}_k(\theta^{\text{out}}_{a,r})], \tag{14}$$

$$\text{s.t. } \mathbb{E}[\mathsf{S}_j(\theta^{\text{out}}_{a,r})] \ge \mathsf{S}_j(\theta^{\text{inc}}_k) - \epsilon_j, \quad j < k, \tag{15}$$

$$\mathbb{E}[\mathsf{C}_k(a,r)] \preceq B_k. \tag{16}$$

The Lagrangian is

$$\mathcal{J}(\psi,\kappa,\lambda) = \mathbb{E}_{\rho_\psi}\big[\mathsf{S}_k(\theta^{\text{out}}) - \sum_{j<k} \kappa_j(\mathsf{S}_j(\theta^{\text{inc}}_k) - \epsilon_j - \mathsf{S}_j(\theta^{\text{out}})) - \lambda^\top(\mathsf{C}_k - B_k)\big], \tag{17}$$

with $\kappa, \lambda \ge 0$. The experiments use the fixed coefficients in Table 8; the corresponding retention statistics are reported with the main component results.

B Controller and Policy Updates

B.1 Controller inputs and architecture

The initial controller is a two-tower categorical policy. Each archive member is encoded independently, pooled with attention and scored against the stage context. The repair head receives the selected parent embedding and post-adaptation measurements. This architecture is deliberately small relative to the language model.

The frozen parent input concatenates six feature groups. Per-task score is the controller-panel mean and its paired-bootstrap lower bound. Reward-bearing rollout fraction is the number of positive verified completions divided by attempted completions in the fixed probe. Short-block slope is the ordinary-least-squares slope of score against normalized update tokens over the probe checkpoints. Update norm is the LoRA parameter displacement divided by the square root of the trainable-parameter count. Age counts completed task boundaries since checkpoint creation; depth counts lineage edges; tasks seen counts distinct training tasks; adapter norm uses the same root-mean-square normalization as update norm. Remaining budget is the unused fraction of each resource coordinate; throughput is generated tokens per accelerator-second; swap cost is measured materialization-plus-reset time divided by the stage budget. Bootstrap width is the 95% percentile-interval width and root count is the number of independent development roots contributing to the estimate. An explicit mask marks every unavailable coordinate. The recovery head additionally receives the complete post-adaptation regression vector, verified donor coverage, donor age and measured adaptation cost. Normalization statistics are fitted on development histories only; final panels are never features.

Let $e_i = \phi_\zeta(f_i, m_i)$ encode archive member i and $c = \phi_c(f_k^{\mathrm{stage}})$. The parent logits are

$$\ell_i^A = u_A^\top \tanh(W_A[e_i; c; \bar{e}]) + b_A, \qquad \bar{e} = \sum_i \alpha_i e_i,$$

where f_i is the numeric feature vector, m_i is its binary missingness mask, f_k^{stage} is the stage context, brackets denote vector concatenation and α_i is a masked attention-pooling weight satisfying $\alpha_i \geq 0$ and $\sum_i \alpha_i = 1$. The recovery logits are

$$\ell_r^R = u_r^\top \tanh(W_R[e_a; c; d_a; g_r]) + b_r,$$

where d_a contains post-adaptation deltas and g_r describes the recovery action and eligible donor. If $\mathcal{A}_k^{\mathrm{feas}}(x)$ and $\mathcal{R}_k^{\mathrm{feas}}(y, a)$ are the feasible sets, the categorical probabilities are

$$\begin{aligned} \rho^A(i \mid x) &= \frac{\mathbf{1}\{i \in \mathcal{A}_k^{\mathrm{feas}}(x)\} \exp(\ell_i^A / T_A)}{\sum_j \mathbf{1}\{j \in \mathcal{A}_k^{\mathrm{feas}}(x)\} \exp(\ell_j^A / T_A)}, \\ \rho^R(r \mid y, a) &= \frac{\mathbf{1}\{r \in \mathcal{R}_k^{\mathrm{feas}}(y, a)\} \exp(\ell_r^R / T_R)}{\sum_{r'} \mathbf{1}\{r' \in \mathcal{R}_k^{\mathrm{feas}}(y, a)\} \exp(\ell_{r'}^R / T_R)}, \end{aligned} \tag{18}$$

with temperatures $T_A, T_R > 0$. Temperatures and an exploration floor are fixed on controller-development histories. Failed actions remain in the outcome log; infeasible actions are masked before sampling and receive no probability.

B.2 Seq-2H and the recovery-aware policy backup

The empirical comparator is Seq-2H, a complete sequential algorithm rather than the mathematical additive model used in the rectangle diagnostic. Before adaptation it computes

$$S_\phi^A(x, a) = u_{\mathrm{S}}^{A\top} \tanh(W_{\mathrm{S}}^A[e_a; c; \bar{e}]) + b_{\mathrm{S}}^A, \qquad \widehat{a} = \arg\max_{a \in \mathcal{A}_k^{\mathrm{feas}}} S_\phi^A(x, a). \tag{19}$$

Only $\widehat{a}$ is cloned and adapted. After observing its probe $y(\widehat{a})$, the second head computes

$$S_\eta^R(y(\widehat{a}), \widehat{a}, r) = u_{\mathrm{S}}^{R\top} \tanh(W_{\mathrm{S}}^R[e_{\widehat{a}}; c; d_{\widehat{a}}; g_r]) + b_{\mathrm{S}}^R, \qquad \widehat{r} = \arg\max_{r \in \mathcal{R}_k^{\mathrm{feas}}(y(\widehat{a}), \widehat{a})} S_\eta^R(y(\widehat{a}), \widehat{a}, r). \tag{20}$$

The recovery score uses the parent embedding and parent-dependent probe measurements available after adaptation, so Seq-2H represents conditional recovery preferences rather than an additive

function $f(x,a)+h(x,r)$. Both controllers commit to $\widehat{a}$ before observing this probe; they differ in the target used to train that first decision.

Define the post-probe value

$$q(x,a,y,r)=\mathbb{E}[G_k \mid x,a,y,r], \tag{21}$$

where the expectation is over recovery optimization, decoding, verification, terminal evaluation and any failure randomness not fixed by (x,a,y,r). Let the frozen randomized coverage policy be $\mu(a,r \mid x,y)=\mu^A(a \mid x)\mu^R(r \mid y,a)$. Seq-2H fits the behavior-marginal parent value

$$Z_\mu(x,a)=\mathbb{E}_{Y|x,a}\left[\sum_{r\in\mathcal{R}_k^{\mathrm{feas}}(Y,a)}\mu^R(r \mid Y,a)q(x,a,Y,r)\right],$$
$$\mathcal{L}_{\mathrm{S}}^A=\sum_i w_i\ \mathrm{Huber}\big(S_\phi^A(x_i,a_i)-G_i\big), \tag{22}$$

where root-balancing weights w_i also make the empirical recovery-cell distribution equal to the prespecified μ^R. Thus S_ϕ^A is a robust regression of terminal reward under the coverage-policy measure; replacing Huber loss by squared loss makes its population minimizer exactly Z_μ. Its second head minimizes

$$\mathcal{L}_{\mathrm{S}}^R=\sum_i w_i\ \mathrm{Huber}\big(S_\eta^R(y_i,a_i,r_i)-G_i\big). \tag{23}$$

Thus S_ϕ^A values a parent under the fixed coverage mixture, while S_η^R chooses a recovery only after that parent has been committed.

EvoLPO-S instead replaces the coverage-mixture target with a parent target weighted by the learned soft recovery policy:

$$Z_\rho(x,a)=\mathbb{E}_{y|x,a}\left[\sum_{r\in\mathcal{R}_k^{\mathrm{feas}}(y,a)}\rho_\nu^R(r \mid y,a)\,q(x,a,y,r)\right],$$
$$\mathcal{L}_H=\sum_i w_i\ \mathrm{Huber}\Big(H_\chi(x_i,a_i)-\widehat{Z}_{\rho,i,a_i}\Big), \tag{24}$$

where $w_i\ge 0$ is the frozen root-and-cell balancing weight and $\mathrm{Huber}(u)=\frac{1}{2}u^2$ for $|u|\le\delta_H$ and $\delta_H(|u|-\frac{1}{2}\delta_H)$ otherwise, with threshold δ_H fixed on development folds. The estimator uses complete recovery forks rather than a fitted counterfactual critic. For coverage record i, adaptation is executed once to obtain $y_i(a)$; that provisional child is cloned before recovery and every scheduled feasible recovery is executed. Let $\mathcal{E}_i(a)$ be this finite set of executed recovery cells and let $G_{i,a,r}$ be the observed terminal reward in cell r. With the recovery head frozen, define

$$\widetilde{\rho}_i^R(r)=\frac{\mathbf{1}\{r\in\mathcal{E}_i(a)\}\rho_\nu^R(r \mid y_i(a),a)}{\sum_{r'}\mathbf{1}\{r'\in\mathcal{E}_i(a)\}\rho_\nu^R(r' \mid y_i(a),a)},\qquad \widehat{Z}_{\rho,i,a}=\sum_{r\in\mathcal{E}_i(a)}\widetilde{\rho}_i^R(r)G_{i,a,r}. \tag{25}$$

Target construction is a direct policy backup over observed completed branches. It evaluates the learned soft recovery distribution and does not, in general, equal the backup of the greedy rule used at deployment. Failed recoveries contribute their bounded failure reward and incurred cost. Structurally infeasible recoveries are removed before the policy is normalized. If an infrastructure failure removes a scheduled feasible cell, that coverage record is excluded from the parent regression and its consumed resources remain in the ledger.

Matched post-transition target. The main controller assigns credit before archive admission and incumbent selection. To test this choice directly, we construct a second label from the same development branch. Let

$$\theta_{i,k+1}^{\mathrm{inc},D}(a,r)=T_k^{\mathrm{inc}}(\mathcal{A}_{i,k},\theta_{i,a,r}^{\mathrm{out}},\theta_{i,k}^{\mathrm{inc}}) \tag{26}$$

be the incumbent selected after applying the frozen archive transition to recovery cell (a,r) on a development-only selection panel. Define

$$G_{i,a,r}^D=\mathsf{U}_k(\theta_{i,k+1}^{\mathrm{inc},D}(a,r))-\mathsf{U}_k(\theta_{i,k}^{\mathrm{inc}})-\lambda^\top\mathsf{C}_{i,a,r}$$
$$-\sum_{j<k}\kappa_j[-\epsilon_j-\Delta_{i,j}^{\mathrm{inc}}(a,r)]_+,\qquad \widehat{Z}_{\rho,i,a}^D=\sum_{r\in\mathcal{E}_i(a)}\widetilde{\rho}_i^R(r)G_{i,a,r}^D, \tag{27}$$

where $\Delta_{i,j}^{\mathrm{inc}}$ measures the earlier-task score change of the selected incumbent. We fit $\widehat{Z}_{\rho}^{D}$ with the same parent network, Huber threshold, weights, folds, recovery head and optimization schedule used for $\widehat{Z}_{\rho}$. Controller inputs are unchanged and the development transition panel remains disjoint from final evaluation. This control changes only whether branch utility is measured on the completed child or after the deterministic archive transition.

Soft and greedy recovery targets. The supervised controller constructs targets with the out-of-fold soft recovery policy ρ_{ν}^{R} from Equation (18). For each provisional child, every scheduled feasible recovery is executed and the target averages terminal rewards using the normalized recovery probabilities. Deployment executes the feasible recovery with maximum probability (Algorithm 1, line 8). The soft backup therefore estimates a stochastic recovery policy, whereas deployment uses its greedy mode. Table 4 reconstructs the greedy target and Section I.8 trains and deploys the corresponding controller.

Recovery fitting and parent-target construction use four-fold cross-fitting over root–order blocks. In fold b, S_{η}^{R} and ρ_{ν}^{R} are fitted on the other three folds and frozen while Equation (25) is evaluated on fold b. Pooling the four out-of-fold targets yields the training set for H_{χ}. Feature normalization, temperature, masks, stopping rules and hyperparameters are fitted inside the corresponding training folds. Incumbent-selection and final panels remain outside this procedure. Both controllers condition recovery on the parent; their parent heads differ in whether they weight completed-branch outcomes by the fixed coverage mixture or by the learned soft recovery distribution. The latter informs parent selection but is not claimed to be the exact greedy-deployment backup.

Proposition 4 (Recovery-policy backup)**.** *Let Y be the probe generated after adapting parent a in state x. Let $\rho^{R}(\cdot \mid Y, a)$ be a frozen recovery policy supported on the feasible recovery set and assume $\mathbb{E}|G_k| < \infty$. Then*

$$Z_{\rho}(x,a) = \mathbb{E}_{Y|x,a}\left[\sum_{r} \rho^{R}(r \mid Y,a) q(x,a,Y,r)\right] = \mathbb{E}_{\rho}[G_k \mid x,a]. \tag{28}$$

If record i contains every recovery cell to which the frozen policy assigns positive probability, each $G_{i,a,r}$ is an unbiased draw from its conditional reward distribution and the recovery policy is fitted without using those held-out rewards, then

$$\mathbb{E}[\widehat{Z}_{\rho,i,a} \mid x_i, a, y_i(a)] = \sum_{r} \rho^{R}(r \mid y_i(a), a) q(x_i, a, y_i(a), r). \tag{29}$$

Averaging independent coverage records over the distribution of Y therefore estimates $Z_{\rho}(x,a)$.

Proof. We first condition on the probe $Y = y$. Recovery is a finite categorical action, so the conditional law of total expectation gives

$$\begin{aligned}\mathbb{E}_{\rho}[G_k \mid x,a,Y=y] &= \sum_{r} \Pr_{\rho}(r \mid y,a)\mathbb{E}[G_k \mid x,a,Y=y,r] \\ &= \sum_{r} \rho^{R}(r \mid y,a) q(x,a,y,r).\end{aligned}$$

Now average both sides over $Y \sim p(\cdot \mid x,a)$. This yields the first equality in Equation (28); the tower property $\mathbb{E}[\mathbb{E}[G_k \mid x,a,Y] \mid x,a] = \mathbb{E}[G_k \mid x,a]$ gives the second.

For the finite-sample statement, complete support makes the denominator in Equation (25) equal to one, so $\widetilde{\rho}_{i}^{R}(r) = \rho^{R}(r \mid y_i(a), a)$. Conditional on $(x_i, a, y_i(a))$, the weights are fixed because recovery fitting is out of fold. Linearity of expectation then gives

$$\begin{aligned}\mathbb{E}[\widehat{Z}_{\rho,i,a} \mid x_i,a,y_i(a)] &= \sum_{r} \rho^{R}(r \mid y_i(a),a)\mathbb{E}[G_{i,a,r} \mid x_i,a,y_i(a)] \\ &= \sum_{r} \rho^{R}(r \mid y_i(a),a) q(x_i,a,y_i(a),r),\end{aligned}$$

which is Equation (29). Finally, averaging over independently generated probes applies the law of large numbers to the outer expectation. A single fork record estimates the value conditional on its realized probe; it is not by itself an estimate of the full probe-marginal value. □

Proposition 4 applies directly to the soft categorical policy used by the supervised backup. The greedy deployed rule is the corresponding degenerate categorical policy, assigning probability one to the deterministically tie-broken maximizer, so the same tower-property identity holds for it. The two targets therefore differ in which policy they evaluate, not in whether the identity is available.

Table 4: Greedy-target offline sensitivity. Both rows rescore the same completed policies. The greedy row reconstructs the parent target under the deployed recovery rule while holding the parent head and archive trajectories fixed.

Parent target	Offline score	Paired Δ to registered soft backup
Registered soft backup	81.7	0.0
Greedy-mode reconstruction	81.5	$-0.2\ [-0.7, 0.3]$

Reconstructing the greedy recovery target changes the parent ranking at 8.3% of covered states and shifts the rescored contrast by -0.2 points, with root-clustered 95% CI $[-0.7, 0.3]$. This fixed-trajectory calculation uses states whose logged forks contain the greedy-preferred recovery. The prospective experiment in Section I.8 then refits the parent head and reruns deployment from the same initial conditions.

The contrast with the coverage target follows directly. For the same (x, a),

$$Z_\rho(x,a) - Z_\mu(x,a) = \mathbb{E}_{Y|x,a}\left[\sum_r \big(\rho^R(r \mid Y, a) - \mu^R(r \mid Y, a)\big) q(x, a, Y, r)\right]. \tag{30}$$

The expression is zero when the two recovery policies agree or when recovery leaves conditional value unchanged. In all other cases it measures how changing the recovery policy shifts the parent value and it can reverse the parent ranking.

Seq-2H and EvoLPO-S use the same trace pool, Huber loss family, root-balanced minibatches, action masks and tuning trials. Their networks are width-matched so that trainable parameter counts differ by less than 0.3%. Infeasible and failed executed branches retain their recorded bounded target. The full EvoLPO row instead fits the joint categorical heads by policy gradient using the terminal advantage, entropy coefficient, batches, update count and early-stopping rule stated in the run manifest. EvoLPO-S minus Seq-2H under LOFO is the primary methodological estimand: it measures the effect of replacing coverage-mixture weighting with learned-recovery-policy weighting. Its estimate is $0.70\,[0.10, 1.30]$. Full EvoLPO minus Seq-2H is the primary in-family system endpoint and includes the fitting change.

Equation (25) turns randomized recovery forks into a parent target. The probe $y_i(a)$ appears after adapting a checkpoint, feasibility depends on the resulting child and deployment follows one completed branch. Cross-fitting lets the recovery policy weight held-out branch outcomes without training its parent head on the same recovery rewards.

Common-state selector evaluation gives both methods identical checkpoint contents. End-to-end deployment instead matches the initial checkpoint, capacity, admission, eviction and feasibility rules; subsequent archives evolve from each method's earlier choices.

B.3 Inner RL and verified repair

For prompt q, GRPO (Shao et al., 2024) draws a group of G responses $o_1{:}G \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$ and verifier rewards v_i. Define group-normalized advantages

$$\widehat{A}_i = \frac{v_i - \bar{v}}{s_v + \varepsilon}.$$

Here

$$\bar{v} = \frac{1}{G}\sum_{i=1}^{G} v_i, \qquad s_v = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(v_i - \bar{v})^2},$$

and $\varepsilon > 0$ is a fixed numerical stabilizer. Response $o_i = (o_{i,1}, \ldots, o_{i,|o_i|})$ has $|o_i|$ generated tokens. At token position t, the likelihood ratio between the updated policy and the rollout policy is

$$\omega_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}. \tag{31}$$

A clipped response-level GRPO objective is

$$\mathcal{L}_{\text{RL}}(\theta) = -\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t}\min\left\{\begin{array}{l}\omega_{i,t}(\theta)\widehat{A}_i, \\ \text{clip}(\omega_{i,t}, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\widehat{A}_i\end{array}\right\}\right] + \gamma\,\mathbb{E}[\text{KL}(\pi_\theta \| \pi_{\text{ref}})]\,, \tag{32}$$

where $\text{clip}(u, l, h) = \min\{h, \max\{l, u\}\}$, $\epsilon_{\text{clip}} > 0$ is the frozen trust interval, $\gamma \geq 0$ is the reference-policy penalty and π_{ref} is the frozen reference used for that update block. The outer expectation is over prompts and response groups sampled by the rollout policy. The implemented KL term is the mean tokenwise divergence on sampled prefixes; the normalization, KL estimator, invalid-output rule and reference update are fixed across methods. The leading minus sign converts the maximization surrogate into a loss minimized by the optimizer and the KL penalty is added outside that negation so that minimizing $\mathcal{L}_{\text{RL}}$ penalizes divergence from π_{ref}.

Replay computes this objective on complete on-policy response groups. Failed responses remain in the group and receive their verifier reward; no success filter is applied before $\bar{v}$, s_v, or $\widehat{A}_i$ is computed. A group whose responses all receive the same reward has $\widehat{A}_i = 0$ for every member. Because the implemented KL penalty is evaluated on the sampled prefixes independently of the reward-normalized advantage, those prefixes still enter the KL term. Success filtering is confined to the verified donor set used by the supervised loss below.

For the TROLL robustness condition (Becker et al., 2026), each current token distribution $\widetilde{p}_t$ is projected toward the rollout distribution p_t^{old}:

$$p_t^\star = \arg\min_{p \in \Delta^{V-1}} \text{KL}(p \| \widetilde{p}_t) \quad \text{s.t.} \quad \text{KL}(p \| p_t^{\text{old}}) \leq \varepsilon_{\text{TR}}. \tag{33}$$

Here $\widetilde{p}_t \in \Delta^{V-1}$ is the unconstrained updated distribution at the sampled prefix, p_t^{old} is the rollout distribution on the same vocabulary and $\varepsilon_{\text{TR}} \geq 0$ is the frozen trust-region radius. The projected probability of the sampled token replaces the clipped probability in the policy ratio; all lineage actions, controller inputs, resource accounting and recovery choices remain unchanged. The projection radius and sparse-support rule are tuned only on development streams. This condition tests whether lineage selection survives when the inner update itself is stabilized by a principled token-level trust region.

For verified donor repair, $\mathcal{B}_d^{\text{ver}}$ contains the donor's tuples for earlier-task prompts whose outputs pass the declared verifier and the child minimizes

$$\mathcal{L}_{\text{child}} = \mathcal{L}_{\text{RL}} + \alpha_{\text{SFT}}\,\mathbb{E}_{(q,y)\sim\mathcal{B}_d^{\text{ver}}}\left[-\frac{1}{|y|}\sum_{t=1}^{|y|}\log \pi_\theta(y_t \mid q, y_{<t})\right], \tag{34}$$

whose supervised term is length normalized; α_{SFT} is fixed on development streams (Table 8). The child never receives final-test inputs. If a donor produces no verified output for a prompt, the example is absent and the shortfall is logged; it is not replaced using knowledge of final performance. Self, pooled, fixed and selected-donor variants use the same attempted-generation cap and the same prespecified number of accepted verified update tokens. Accepted sets are subsampled to the common target without reference to final outcomes; failure to reach that target makes the arm infeasible and leaves its consumed generation and verification cost in the ledger.

B.4 Archive update, candidate set and deployment

For a completed child c, define new-task gain $g_k(c)$ and old-task losses $\ell_j(c) = \mathsf{S}_j(\theta_k^{\text{inc}}) - \mathsf{S}_j(c)$. Bounds use 2,000 paired, task-stratified bootstrap resamples of controller-panel problem identifiers, generated from the manifest seed before any admission decision. Let $L_{.05}(g_k)$ be the fifth percentile of the gain replicates and $U_{.95}(\ell_j)$ the 95th percentile of the loss replicates. The child is admissible when

$$L_{.05}(g_k(c)) \geq 0, \qquad U_{.95}(\ell_j(c)) \leq \epsilon_j \quad \forall j < k.$$

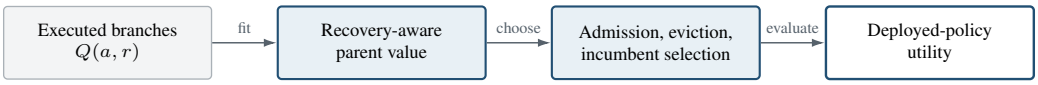


Figure 5: Analysis from branch outcomes to deployment. Executed parent–recovery branches train the parent value; the archive transition then maps the selected completed child to the next deployed incumbent. Branch regret and deployed-policy utility measure the two stages separately.

Thus gain must have a one-sided 95% lower bound at least zero and each prior-task loss must have a one-sided 95% upper bound no larger than its frozen tolerance. The same resample indices are shared between incumbent and child, preserving pairing. This fixed development rule determines archive admission; sealed panels evaluate the resulting policies.

For eviction after stage k, define the fixed capability vector $s_k(c) = (\mathsf{S}_1(c), \ldots, \mathsf{S}_k(c)) \in [0,1]^k$ on the controller panel, the reference point $z_k^{\mathrm{ref}} = \mathbf{0}_k$ and coverage contribution

$$h_k(c; \mathcal{A}) = \mathrm{HV}(\{s_k(u) : u \in \mathcal{A}\}; \mathbf{0}_k) - \mathrm{HV}(\{s_k(u) : u \in \mathcal{A} \setminus \{c\}\}; \mathbf{0}_k),$$

where all coordinates are maximized. Explicitly, for a finite score set $S \subset [0,1]^k$,

$$\mathrm{HV}(S; \mathbf{0}_k) = \mathrm{Leb}\left(\bigcup_{z \in S} \prod_{h=1}^{k} [0, z_h] \right),$$

where Leb denotes k-dimensional Lebesgue measure. Thus $h_k(c; \mathcal{A})$ is the score-space volume covered uniquely by child c. After an admissible child is inserted, if $|\mathcal{A}| > M$, remove the non-incumbent member with smallest h_k, breaking ties by the oldest creation stage and then manifest key. A rejected child is never inserted. The primary system uses this fixed rule; an oldest-first rule provides the eviction sensitivity.

The deployment candidate set. Incumbent choice occurs only after admission and capacity restoration and uses a disjoint 400-example selection panel. The controller, short probes, recovery threshold, admission rule and hypervolume coordinates never observe these examples. All methods are evaluated on the same panel and the selected incumbent is then scored once on the sealed final panel. The registered system exposes the eligible policies present after admission and eviction, which is at most the archive cap $M = 4$. The completed-child reward remains attached to the executed branch even when the child is rejected or another archive member wins deployment. Task-wise routing is used only as an oracle.

Candidate-count sensitivity is a post-execution analysis over logged completed policies. At each selection point, candidates are ordered by the frozen controller score after feasibility checks and $n \in \{1, 2, 4, 8\}$ exposes the first n policies to the same selection panel. The four-candidate row matches the online archive size. The eight-candidate row is an oracle-style exposure set that may include policies rejected online; its branch cost is charged as randomized coverage or failed execution in Table 27.

C Prospective Evaluation Protocol

C.1 Development, freezing and held-out deployment

The interaction study crosses four checkpoint histories with three recovery actions at one boundary from each of three arriving task families and four independent training roots, yielding $4 \times 3 \times 3 \times 4 = 144$ terminal branch outcomes. Each family contributes a twelve-cell opportunity matrix estimated from four roots. Recovery arms share acquisition prefixes when their treatment definitions coincide and the analysis preserves this pairing.

There are $\binom{4}{2}\binom{3}{2} = 18$ parent–recovery rectangles per matrix. A rectangle is a *replicated reversal* only if the preferred-parent contrast changes sign across the two recoveries in at least three of four diagnostic roots and the paired 95% interval for each margin excludes zero in its corresponding direction.

Table 5: Qualifying rectangles under the replicated-reversal criterion: the preferred parent changes across two recoveries in at least three of four roots and both paired 95% margins exclude zero in their respective directions. Counts are reported separately for the three family matrices.

Family matrix	Qualifying rectangles	Total rectangles
Math	4	18
Code	3	18
Mixed	7	18

The three matrices are family-specific mechanism studies and their counts describe the sampled rectangles within each family. The mixed family has the most qualifying rectangles and the weakest leave-one-stream-family-out transfer. Section G.1 displays one complete matrix and Section I.1 gives its root-level interaction uncertainty.

The additive null is fitted separately at each held-out boundary,

$$Q^{(0)}_{a,r} = \alpha_a + \beta_r, \qquad \widehat{\varepsilon}_{s,a,r} = \widehat{Q}_{s,a,r} - \widehat{\alpha}_a - \widehat{\beta}_r.$$

The additive fit shows which cell residuals generate the interaction. With four roots, its 12-dimensional residual covariance is low-rank; the analysis therefore reports the three family matrices separately. Seq-2H is a learned sequential controller and is evaluated independently of this theoretical additive fit.

For each measured state, the analysis computes the complete $Q(a, r)$ table with root-level uncertainty. It reports:

1. maximum absolute rectangle interaction $\max |I|$ with multiplicity-aware interval;
2. magnitude and count of parent ranking reversals across recovery actions;
3. regret of the theoretical additive fit, Seq-2H and joint oracle on the same executed cells;
4. net value after all branch measurements and failed trials;
5. robustness to removing age, depth and task-identity features.

Thresholds are chosen on development states and then applied to the held-out roots.

C.2 Continual streams and data partitions

The math stream is GSM8K → MATH algebra → MATH number theory → MATH counting and probability. The code stream is APPS introductory → APPS interview → PIE sequence optimization → PIE graph optimization. The mixed stream alternates GSM8K, APPS introductory, MATH algebra and PIE sequence optimization. A separate eight-stage mixed stress test appends disjoint MATH geometry and intermediate-algebra panels plus PIE string and dynamic-programming optimization families and a twelve-stage extension appends four further disjoint panels in alternating order: MATH prealgebra, APPS competition, MATH precalculus and PIE array optimization. Within each experimental condition, task panels are identifier-disjoint across stages. Math and mixed conditions may use the same named task family, but their development and evaluation identifiers are partitioned independently. The APPS difficulty tiers are used in their released order. With $M = 4$, every admission after stage 4 triggers an eviction decision in both stress tests. Revisit examples are freshly sampled and never reuse controller-development or incumbent-selection items.

Draws and distinct identifiers. We distinguish sampled instances from distinct native problem identifiers. Table 6 reports per-stage draw budgets: $6{,}000$ training draws, 800 development draws and $1{,}200$ sealed-evaluation draws. Table 7 reports the distinct native identifiers those draws represent. Identifier sets are formed before prompt formatting and are disjoint across training, controller/probe, incumbent-selection and sealed evaluation. When a draw budget exceeds the number of available identifiers, the loader revisits identifiers on the frozen sampling schedule; repeated draws are not treated as independent problems. The confirmatory deployment estimand clusters by training root on fixed panels (Section H.1); it therefore characterizes variation across training runs rather than across newly sampled problem identities.

Table 6: Per-stream sampling and replication budget for the focused 30B-total/3B-active MoE setting and its eight- and twelve-stage stress tests. The 6,000, 800 and 1,200 columns are *draw counts*; Table 7 reports the distinct native identifiers those draws represent. Smaller-scale replication counts are reported separately in Table 27.

Stream	Stages	Orders	Roots	Train/stage	Dev/stage	Test/stage
Math	4	4	8	6000	800	1200
Code	4	4	8	6000	800	1200
Mixed	4	4	8	6000	800	1200
Long mixed stress	8	3	6	6000	800	1200
Extended mixed stress	12	3	6	6000	800	1200

GSM8K uses 6,000 of the 7,473 released training identifiers. MATH and APPS training traverse every identifier in the named released training subset, with repeated draws where the budget exceeds the pool. GSM8K and each MATH subject partition the released test identifiers approximately $40/60$ into development and sealed-test sets. APPS introductory and competition partition their 1,000 released test problems as $400/600$; APPS interview uses $800/1{,}200$ of its 3,000 released test problems and leaves 1,000 unused. PIE evaluation is constructed from the 1,474 released training problems instead of the $77/41$ official validation and test problems. We first partition this pool into five disjoint optimization families of 300, 300, 295, 290 and 289 problems, then form disjoint train, development and sealed-test identifier sets within each family.

Table 7: Per-stage identifier partitions from the run manifests. Training draws are sampled from the training-identifier pool on the frozen schedule; development and sealed panels are disjoint identifier sets. Multiplicity is training draws divided by unique training identifiers. GSM8K is the only stage whose released training pool exceeds the draw budget, so it uses 6,000 of 7,473 identifiers; every other stage traverses its full pool. GSM8K and MATH development/test columns sum to the released test-subject size; the five PIE family pools sum to the 1,474 released training problems.

Stage	Source train IDs	Unique train IDs	Train draws	Mult.	Unique dev IDs	Unique test IDs
GSM8K	7473	6000	6000	1.00	528	791
MATH algebra	1744	1744	6000	3.44	475	712
MATH number theory	869	869	6000	6.90	216	324
MATH counting/probability	771	771	6000	7.78	190	284
MATH geometry	870	870	6000	6.90	192	287
MATH intermediate algebra	1295	1295	6000	4.63	361	542
MATH prealgebra	1205	1205	6000	4.98	348	523
MATH precalculus	746	746	6000	8.04	218	328
APPS introductory	2639	2639	6000	2.27	400	600
APPS interview	2000	2000	6000	3.00	800	1200
APPS competition	361	361	6000	16.62	400	600
PIE sequence optimization	300	150	6000	40.00	60	90
PIE graph optimization	300	150	6000	40.00	60	90
PIE string optimization	295	148	6000	40.54	59	88
PIE dynamic programming	290	145	6000	41.38	58	87
PIE array optimization	289	145	6000	41.38	58	86

The 800 development and 1,200 sealed PIE observations are repeated draws from 58–60 and 86–90 distinct problems per family, respectively. Repeated prompts, candidate programs and runtime trials improve measurement precision conditional on those problems; the independent task-instance count remains the number of distinct problems. The PIE results therefore describe the bounded problem panels rather than 1,200 unique problems. Training multiplicity is correspondingly high for the smallest pools, reaching $41\times$ for the PIE families and $16.6\times$ for APPS competition.

Every reported score, including the completed-child utilities in Section D.3, uses this registered weight rule; with uniform weights the stage-k utility is the unweighted mean of encountered-task scores and Section D.3 reproduces $U_4 = 0.7825$ from $\mathbf{s}^{\mathrm{out}} = (0.90, 0.87, 0.82, 0.54)$. The alternatives in Table 31 are rescorings of the same completed policies and are never used to produce a reported result.

For PIE example i among the N problems of the evaluated panel, let $v_i = 1$ only when the candidate passes every functional test and let s_i be the verified median speedup (source runtime divided by

Table 8: Registered utility parameters, frozen before any held-out deployment. The weight rule is the one used for every reported score; Table 31 rescores completed policies under alternatives to it.

Symbol	Quantity	Value
$w_{k,j}$	Task weight at stage k	$1/k \quad \forall j \le k$
ϵ_j	Retention tolerance	$0.02\ \forall j$
κ_j	Excess-regression price	$1.00\ \forall j$
λ	Cost coefficients, ordered as c_m in Section H.3	see note
α_{SFT}	Donor-repair SFT coefficient	0.20

$\lambda = (0.006, 0.006, 0.008, 0.010, 0.002, 0.003, 0.005)$ prices, in order, generated tokens, update tokens, training FLOPs, GPU-hours, CPU-hours, storage and wall time, each per normalized stream-equivalent of that coordinate.

candidate runtime). The task score is

$$\mathsf{S}_{\mathrm{PIE}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} v_i \min\left\{1, \frac{[\log_2 s_i]_+}{\log_2 16}\right\}.$$

Thus failing programs score zero, correct programs that are no faster than the source also score zero, verified speedups receive logarithmic credit and credit saturates at $16\times$. This $[0, 1]$ score is distinct from the useful-optimization rate in Table 9, which is the fraction of attempted programs passing every test and satisfying the prespecified speedup and bootstrap-LCB thresholds; only the former enters Equation (1).

Each source is split by original problem identifier before prompt formatting. The 800 development examples are deterministically partitioned by problem identifier into 400 controller/probe and 400 incumbent-selection examples. Controller traces enter four root–order cross-fitting folds before target construction; a problem identifier and its provisional child remain in one fold. Evaluation uses the panels in Table 7. MATH-500 identifiers are removed from every controller-accessible MATH pool, while APPS and PIE are deduplicated with normalized statement hashes, token shingles and available provenance metadata.

Math output extraction removes formatting wrappers, parses numeric or symbolic answers, applies declared equivalence rules and treats malformed or timed-out outputs as incorrect. Code is compiled and executed in an isolated environment with fixed CPU, memory, wall-time and output limits. A PIE candidate is a useful optimization only if it passes every functional test, has median source-time/candidate-time strictly above 1.00 and the paired 95% bootstrap lower bound over the repeated gem5 measurements also exceeds 1.00. Runtime uses the frozen warmup and repeated-trial counts in the run manifest; suspicious speedups are rerun and all failed tests remain in the denominator.

D Dataset Examples and Branch Traces

This section follows six reproducible checkpoint decisions from task input through adaptation, recovery and archive update. The cases are outcome-stratified to cover ordinary deployments, historical-parent selections and adverse outcomes.

Table 9: Dataset-level branch outcomes. For PIE, Initial/Acquire/Complete are *useful-optimization rates*: the fraction of programs passing every functional test and exceeding the frozen speed threshold. They are distinct from the normalized PIE task-utility component in the trace vectors. Cost is in normalized stream-equivalents.

Dataset	Initial	Acquire	Complete	Retention	Regret	Cost
GSM8K	.68	.86	.85	.93	.004	.007
MATH algebra	.42	.74	.72	.90	.009	.010
APPS interview	.31	.53	.52	.89	.011	.012
PIE	.18	.39	.36	.92	.007	.008

GSM8K: the incumbent is already the right parent. Consider a word problem requiring multiplication, a percentage discount and tax. The incumbent begins at 0.68 exact-match accuracy and adapts

to 0.86. The regression probe shows a 4.1-point loss on earlier algebra examples, so the controller chooses on-policy replay. Final GSM8K accuracy is 0.85 and old-task retention is 0.93. Historical alternatives learn no faster after their lower initial competence is accounted for, so the controller retains the incumbent and applies recovery.

MATH algebra: repair changes the best parent. For symbolic factorization, history $k-2$ adapts from 0.42 to 0.74 but loses number-theory performance. Under continued learning, the incumbent's completed utility is higher; under verified donor repair, history $k-2$ recovers the missing task while retaining 0.72 algebra accuracy. The controller therefore selects the older parent only together with donor repair. Equivalent expressions are judged by the fixed symbolic checker and malformed derivations remain incorrect even if their surface form resembles the reference.

APPS: hidden tests expose a misleading acquisition signal. An interval-scheduling problem initially appears solved by a history with high public-example reward. Hidden cases containing duplicate endpoints reduce its pass rate. History $k-1$ instead reaches 0.53 pass@1 and finishes at 0.52 after replay, with 0.89 retention. The branch that fails hidden tests receives zero correctness-gated performance reward regardless of runtime. This example shows why controller inputs use verifier statistics rather than self-reported correctness.

PIE: optimization is gated by semantics. The input program recomputes prefix sums inside a loop. History $k-3$ more often proposes a linear-time cumulative update, improving the useful-optimization rate from 0.18 to 0.39 after adaptation. Donor repair restores earlier code-generation behavior, leaving a final rate of 0.36 and retention of 0.92. Only programs passing every functional test enter timing; speedups from incorrect programs are assigned zero.

D.1 Additional benchmark-format examples

Each example specifies its public Hugging Face repository, configuration, split and native identifier. Prompt text is abbreviated in the table; evaluation uses the complete record and verifier payload from the pinned revision. The examples illustrate the task and verifier contracts used by the branch traces.

GSM8K, official OpenAI release. Source: `openai/gsm8k`, configuration `main`, split `train`, row 0 (repository revision `e53f048`). The record asks how many clips Natalia sold after selling 48 in April and half as many in May; its normalized answer is 72. The full solution contains calculator annotations and ends in `#### 72`. EvoLPO receives verifier reward only from the extracted terminal number, not from overlap with the written derivation. The exact record is available at `https://huggingface.co/datasets/openai/gsm8k`.

MATH algebra, released-benchmark mirror. Source: `EleutherAI/hendrycks_math`, configuration `algebra`, split `train`, row 0. The Level-5 record defines a three-branch piecewise function and asks for $a + b$ under continuity at $x = 2$ and $x = -2$; matching the boundary values yields the boxed answer 0. Evaluation parses the boxed result and applies the frozen symbolic-equivalence rules. The dataset card links the original MATH release and citation: `https://huggingface.co/datasets/EleutherAI/hendrycks_math`.

APPS interview, public HF-hosted benchmark record. Source: `codeparrot/apps`, split `train`, native `id=0`, difficulty `interview`, linked to Codeforces problem `1259/D`. The task asks for the minimum set of binary words to reverse so that all words can be chained by matching adjacent boundary characters while remaining unique. The repository stores the complete statement, candidate solutions and JSON input/output tests. Generated programs are executed against the full frozen tests under resource limits; prose similarity receives no credit. Record provenance and loading instructions are at `https://huggingface.co/datasets/codeparrot/apps`.

PIE optimization, measured HF record. Source: `stablegradients/pie-gem5-bysrc`, configuration `default`, split `test`, row 0, with `problem_id=p02676` and `src_id=s000084108`. The source is a C++ program that reads an integer and a string, truncates the string to the requested length and appends an ellipsis when required. The record includes per-test-case gem5 tick measurements rather than a textual speed label. An optimized child must preserve outputs

before its measured execution cost is compared with the frozen source. The exact row schema is visible at `https://huggingface.co/datasets/stablegradients/pie-gem5-bysrc`.

All four benchmark records are pinned in the run manifest by repository revision and row identity. The materialized records are hashed after decoding, so a changed upstream viewer, schema, or conversion cannot silently alter the evaluation set.

D.2 What the reported experiments establish by dataset

Across the four measured benchmark families, EvoLPO finishes at 82.8 aggregate final score. It exceeds Seq-2H by 1.79 points, with root-clustered 95% interval $[1.17, 2.42]$ and latest + matched repair by 3.01 points, with interval $[2.15, 3.88]$. The first contrast compares complete algorithms and includes controller fitting. The supervised controller improves over supervised Seq-2H by 0.70 points under leave-one-stream-family-out evaluation, $[0.10, 1.30]$. The mixed stream overlaps both task domains, so this experiment measures composition transfer. The traces below explain individual decisions within these aggregate results.

On GSM8K, the method usually keeps the incumbent: acquisition rises from .68 to .86 and recovery preserves nearly all of it at .85, with .93 retention. On MATH algebra, an older history can acquire more effectively ($.42 \to .74$), but its value becomes visible only after matched repair leaves .72 task performance and .90 retention. APPS begins from the weakest general code competence after PIE; its useful pass rate moves from .31 to .53 and completes at .52, while hidden tests prevent superficial branches from being selected. PIE has the largest proportional acquisition gain ($.18 \to .39$); verified donor repair trades three points of the new optimization rate for substantially stronger prior-task retention, finishing at .36 and .92.

The complete opportunity matrix in Section G.1 exhibits a reversal and Table 5 reports qualifying rectangles in the three family matrices. In the eight-stage archive intervention, hypervolume eviction with $M = 4$ reaches 81.9, close to 82.1 with $M = 8$ and above 81.2 for oldest-first eviction. Together, these results connect recovery-dependent checkpoint rankings to the archive decisions made during deployment.

D.3 Selected branch traces

We report controller decisions. In every case the archive is ordered as $(\theta_k^{\mathrm{inc}}, \theta_{k-1}, \theta_{k-2}, \theta_{k-3})$ and recovery actions as (continue, replay, verified donor). Each trace contains the item, the frozen parent posterior, the sampled parent, the post-update probe, the frozen recovery posterior and the completed encountered-task vector. Probabilities are recorded before sampling.

The trace set is a reproducible, outcome-stratified case study. It contains two ordinary deployments (Traces 1 and 3), two historical-parent deployments (Traces 2 and 4) and two adverse outcomes (Traces 5 and 6). Within each stratum, the two smallest SHA-256 manifest keys determine the examples. The manifest identifiers and sampling seeds below join directly to the recorded posterior, probe, action and score records.

Table 10: Trace provenance. A run is identified by model, root, stream, held-out order and boundary. Seeds are decimal forms of the frozen run-manifest seeds.

Trace	Run identifier	Seed	Stage	Vector order
1	`moe-r06-mix-o09-k4`	4106094	4	APPS/MATH/PIE/GSM8K
2	`moe-r02-math-o03-k4`	4102034	4	GSM8K/number theory/counting/algebra
3	`moe-r05-code-o07-k4`	4105074	4	APPS-intro/PIE-seq/PIE-graph/APPS-interview
4	`moe-r04-mix-o01-k4`	4104014	4	GSM8K/APPS/MATH-algebra/PIE-seq
5	`moe-r07-mix-o12-k4`	4107124	4	GSM8K/APPS/PIE/MATH-algebra
6	`moe-r01-code-o11-k4`	4101114	4	APPS-intro/APPS-interview/PIE-seq/PIE-graph

D.3.1 Trace 1: GSM8K—retain the incumbent

Item. `openai/gsm8k`, `main/train`, row 0, asks for total April–May clip sales. The exact verifier target is 72.

Parent decision. The frozen parent head returns

$$\rho^A(\cdot \mid x_k) = (\mathbf{0.61}, 0.18, 0.13, 0.08), \qquad a = \theta_k^{\text{inc}}.$$

The fixed update raises current-task exact match from .68 to .86. The regression probe changes the mean of previously encountered panels from .884 to .843, a drop of .041, with no feasibility-mask violation.

Recovery and completion. Conditioned on the incumbent and probe result,

$$\rho^R(\cdot \mid y_k, a) = (0.08, \mathbf{0.76}, 0.16), \qquad r = \text{REPLAY}.$$

The completed task vector is

$$\mathbf{s}^{\text{out}} = (0.85, 0.91, 0.54, 0.85),$$

ordered as APPS/MATH/PIE/GSM8K. The encountered GSM8K completion is .85 and retention relative to the prior-task reference maxima is .93, matching Table 9. The child is admitted and deployed; the incumbent remains the selected parent because none of the historical checkpoints offers higher completed-child value.

D.3.2 TRACE 2: MATH ALGEBRA—AN OLDER PARENT NEEDS DONOR REPAIR

Item. `EleutherAI/hendrycks_math`, `algebra/train`, row 0, asks for $a + b$ under continuity of a three-branch piecewise function. The symbolic target is 0.

Parent decision. The posterior is

$$\rho^A(\cdot \mid x_k) = (0.18, 0.16, \mathbf{0.51}, 0.15), \qquad a = \theta_{k-2}.$$

The provisional child improves algebra accuracy from .42 to .74, while prior-task mean falls from .812 to .754. The largest component drop is .083 on number theory; the current item passes symbolic verification.

Recovery and completion. The recovery head assigns

$$\rho^R(\cdot \mid y_k, a) = (0.07, 0.21, \mathbf{0.72}), \qquad r = \text{DONOR}.$$

After using only verifier-accepted donor outputs, the completed vector is

$$\mathbf{s}^{\text{out}} = (0.90, 0.87, 0.82, 0.72).$$

Its order is GSM8K/number theory/counting/algebra, so the arriving algebra task is the final component. Algebra completion remains .72 and normalized retention is .90. Under continued adaptation the same parent is infeasible; donor repair reverses its ranking relative to the incumbent.

D.3.3 TRACE 3: APPS INTERVIEW—HIDDEN TESTS OVERTURN PUBLIC REWARD

Item. `codeparrot/apps`, `train`, native `id=0`, asks for the minimum collision-free set of binary-word reversals. Correctness is evaluated on the complete stored input/output bundle.

Parent decision. The controller produces

$$\rho^A(\cdot \mid x_k) = (0.24, \mathbf{0.48}, 0.17, 0.11), \qquad a = \theta_{k-1}.$$

The provisional child reaches .53 pass@1. Its public-example pass rate is .78, but the hidden collision subset is .49; prior-task mean drops by .026. The discrepancy, rather than the public score alone, enters y_k.

Recovery and completion. The recovery posterior is

$$\rho^R(\cdot \mid y_k, a) = (0.12, \mathbf{0.71}, 0.17), \qquad r = \text{REPLAY}.$$

The completed vector is

$$\mathbf{s}^{\text{out}} = (0.86, 0.79, 0.84, 0.52),$$

ordered as APPS-intro/PIE-seq/PIE-graph/APPS-interview, with the arriving APPS task last. Final APPS pass@1 is .52 and retention is .89. A competing branch with higher public reward fails duplicate-reversal tests and receives zero correctness-gated utility.

D.3.4 TRACE 4: PIE SEQUENCE OPTIMIZATION—LARGE ACQUISITION, STRUCTURED RECOVERY

Item. The PIE task replaces repeated prefix summation by a semantically equivalent cumulative update. Functional tests precede timing, so an incorrect speedup receives zero reward.

Parent decision. The recorded posterior is

$$\rho^A(\cdot \mid x_k) = (0.12, 0.18, 0.21, \mathbf{0.49}), \qquad a = \theta_{k-3}.$$

After the fixed update, hidden pass rate is .88, median speedup is $5.4\times$ and the prior-task mean changes from .836 to .797. The encountered-task component rises from .31 to .56.

Here .31 $\rightarrow$ .56 is the normalized correctness-and-speed task utility used in $\mathbf{s}$, not the useful-optimization rate in Table 9. The latter is a thresholded population proportion (.18 $\rightarrow$ .39); recovery completes these two quantities at .54 and .36, respectively.

Recovery and completion. The recovery head returns

$$\rho^R(\cdot \mid y_k, a) = (0.09, 0.27, \mathbf{0.64}), \qquad r = \text{DONOR}.$$

The completed vector is

$$\mathbf{s}^{\text{out}} = (0.90, 0.87, 0.82, 0.54),$$

with $5.1\times$ verified speedup. Its utility rises from 0.7350 at entry to 0.7825 after recovery; after 0.008 normalized branch cost, terminal credit is $G_4 = 0.0395$.

D.3.5 TRACE 5: AN ORDER WHERE SEQ-2H WINS

Item and decisions. This is one of the two negative stream–order comparisons retained in Figure 6a. On a mixed order ending in MATH algebra, EVOLPO assigns

$$\rho^A_J = (0.22, 0.19, \mathbf{0.44}, 0.15), \qquad a_J = \theta_{k-2},$$

while Seq-2H chooses the incumbent. EVOLPO's provisional child reaches .76 current-task accuracy but loses .071 prior-task mean. Its recovery posterior is

$$\rho^R_J = (0.11, 0.25, \mathbf{0.64}), \qquad r_J = \text{DONOR}.$$

Completed comparison. The joint child completes at

$$\mathbf{s}^{\text{out}}_J = (0.88, 0.82, 0.75, 0.79), \qquad \text{Final}_J = 81.0,$$

whereas the Seq-2H incumbent–replay child completes at

$$\mathbf{s}^{\text{out}}_S = (0.87, 0.83, 0.76, 0.80), \qquad \text{Final}_S = 81.5.$$

Displayed components are rounded to two decimals; the final scores are computed from unrounded panel values. Thus $J - S = -0.5$ on this order. The sampled joint action is feasible and admitted, but Seq-2H produces the stronger final policy.

D.3.6 TRACE 6: A SAMPLED BRANCH THAT FAILS ADMISSION

Item and probe. On a PIE graph-optimization boundary, the parent posterior is

$$\rho^A(\cdot \mid x_k) = (0.14, 0.20, 0.25, \mathbf{0.41}), \qquad a = \theta_{k-3}.$$

The provisional child raises useful optimization from .22 to .61, but prior-task mean falls from .843 to .721 and the worst protected panel falls by .164.

Attempted recovery. The controller chooses donor repair from

$$\rho^R(\cdot \mid y_k, a) = (0.05, 0.18, \mathbf{0.77}).$$

The completed vector is

$$\mathbf{s}^{\text{out}} = (0.88, 0.79, 0.74, 0.61).$$

Although the new-task component remains strong, the one-sided 95% upper bound on the worst protected-panel loss exceeds the registered 0.02 tolerance. The child is rejected, receives its measured negative terminal utility and never becomes the incumbent or an archive member. The deployed policy remains the pre-boundary incumbent. Thus controller exploration and archive admission remain separate: a failed admission leaves the incumbent unchanged while retaining the branch's reward and cost.

E SCOPE OF THE EVIDENCE

The empirical scope comprises the Qwen scale ladder, three verifiable stream families, four-stage main deployments, eight- and twelve-stage archive stress tests and frozen deployment on five additional model families. Within the Qwen family we also report leave-one-stream-family-out transfer. All deployments use low-rank adaptation, visible task boundaries and machine-checkable verifiers.

E.1 FROZEN CROSS-MODEL-FAMILY TRANSFER PROTOCOL

The cross-family test fits EVOLPO-S once on Qwen3-8B development histories and freezes its weights, feature definitions, missingness masks, normalizers, archive rules, feasibility thresholds, recovery actions and cost weights. Raw model identifiers, tokenizer identifiers, vocabulary statistics and architecture labels are excluded from controller inputs. Each target receives its native tokenizer and chat template from a pinned Hugging Face repository revision; scores reach the controller only through the same dimensionless acquisition, regression, uncertainty, age, depth and cost features used for Qwen. No target-family trajectory updates the controller or any selection threshold.

The five target checkpoints span independent model families in a matched 4–8B parameter band:

1. Google `google/gemma-3-4b-it` (4B);
2. Meta `meta-llama/Llama-3.1-8B-Instruct` (8B);
3. Mistral AI `mistralai/Mistral-7B-Instruct-v0.3` (7B);
4. Microsoft `microsoft/Phi-4-mini-instruct` (4B); and
5. Ai2 `allenai/OLMo-2-1124-7B-Instruct` (7B).

Gemma is evaluated in text-only mode. For every target, the three arms are latest + matched repair, Seq-2H and EVOLPO-S. They share roots, orders, prompts, LoRA rank, update tokens, reset behavior, attempted generations, accepted verified tokens, donor repair, verifier calls and the disjoint selection panel. Each target family receives its own paired effect and confidence interval, which preserves architecture-specific throughput and variance.

For target family m and root s, let

$$d_{m,s} = 100\big[\mathsf{U}_K(\theta_{m,s}^{\text{EvoLPO-S}}) - \mathsf{U}_K(\theta_{m,s}^{\text{Seq2H}})\big] .$$

The family-specific estimand is $\Delta_m = S^{-1}\sum_{s=1}^{S} d_{m,s}$ with a root-clustered t_{S-1} interval. We report each family separately and summarize the range of effects across architectures. Cost-adjusted transfer uses

$$\Delta_m^{\text{net}}(n) = \Delta_m - \lambda^\top(\bar{c}_{J,m} - \bar{c}_{A,m}) - \frac{\lambda^\top c_{\text{fit}}}{n},$$

where n is the number of deployments reusing the frozen Qwen-trained controller. Reversal diagnostics are recomputed from each target archive and are not transferred from Qwen. The measured family-specific results appear in Table 36.

F BASELINES AND METRICS

F.1 BASELINE IMPLEMENTATIONS

Every method receives the same model architecture, trainable parameterization, prompt access, verifier and total branch budget within a comparison. Sequential and replay baselines share the GRPO variant (Shao et al., 2024), group size, KL term, prompt pool and token budget. Replay draws earlier-task prompts uniformly by task, generates completions on policy, assigns the frozen task verifier reward and mixes those groups with current-task groups under the same GRPO loss; it has no auxiliary supervised term. Thus it is on-policy prompt replay, not stored-rollout replay: every completion is regenerated by the current child, avoiding the rollout-staleness problem studied by Yoo et al. (2026). Replay ratio and buffer size are tuned on development streams. Replay-GRPO+reset discards first- and second-moment optimizer states at every task boundary but changes nothing else. Latest + matched repair uses that same reset and the same attempted generations, accepted verified tokens and recovery updates as EVOLPO; its only restricted axis is the forced incumbent parent.

CPO-adapted follows the source preservation objective (Luo et al., 2026b) and tunes only its declared coefficient. PBT-GRPO follows population-based training (Jaderberg et al., 2017); QD-GRPO follows MAP-Elites (Mouret & Clune, 2015). Both fix their population capacity and selection rules before evaluation.

Historical short trial adapts checkpoint-readiness selection (Li et al.; Nguyen et al., 2025) to the same archive, reset rule and probe budget charged to EvoLPO. Each archive member runs the fixed short update with recovery set to continue; the parent with the largest lower confidence bound on terminal reward initializes the full boundary update. Uniform soup (Wortsman et al., 2022) averages eligible adapter deltas; task arithmetic (Ilharco et al., 2022) sums centered deltas with a development-fitted scale; TIES (Yadav et al., 2023) trims low-magnitude coordinates, resolves signs and merges aligned updates. Every merged parent then receives the common reset, inner update and recovery budget. Repair controls match donor attempts, verifier calls, accepted verified tokens and optimization steps across self, pooled, fixed and selected donors. The exhaustive branch oracle supplies decision-state regret. The controller comparisons share features and feasibility masks and vary one element at a time: the parent target, the timing of credit, or the fitting objective.

TeLAPA-style origin selection. TeLAPA is an archive-construction and origin-retrieval method rather than an RLVR recovery method (Lillo & Cheney, 2026). Its source algorithm trains a competent PPO policy, illuminates a local parameter-space neighborhood with MAP-Elites, embeds policy trajectories in a shared latent space and forms a compact transfer pool by greedy farthest-point selection. Each candidate is evaluated zero-shot and by a short adaptation probe after resetting optimizer and rollout state. Candidates within ϵ of the best final probe score are retained and the source score then ranks their final success, recovery slope and probe AUC. The probe weights are discarded; the selected archived weights initialize the full update. TeLAPA further maintains the latent coordinate system with anchors, replay-based alignment and periodic archive re-embedding.

Our matched adaptation, denoted *TeLAPA-style + matched repair*, preserves that decision rule while replacing MiniGrid-specific success with the task-normalized score in Equation (1). It receives the same four-entry archive cap, candidate count, reset, probe tokens, verifier calls, inner update, selection panel and completed-child recovery budget as EvoLPO. Farthest-point selection operates on the normalized controller features and the chosen origin receives the same recovery implementation as latest + matched repair. Table 32 compares this latent-diverse short-probe retrieval rule with recovery-policy-aware origin valuation on the common RLVR protocol.

F.2 Metrics

Let $b = 0, \ldots, B$ index equally spaced cumulative update-token checkpoints within stage k and let $s_{k,b}$ be current-task score. The normalized acquisition AUC is

$$\mathrm{AUC}_k = \frac{1}{B} \sum_{b=1}^{B} \frac{s_{k,b-1} + s_{k,b}}{2}.$$

We also report baseline-adjusted progress $s_{k,b} - s_{k,0}$ because low initial competence can reduce raw AUC without implying a smaller learning slope.

For prior task $j < k$, retention is

$$\mathrm{Ret}_{j,k} = \mathsf{S}_j(\theta_k^{\mathrm{inc}}) / \max_{t \le k} \mathsf{S}_j(\theta_t^{\mathrm{inc}}).$$

Backward transfer is $\mathsf{S}_j(\theta_k^{\mathrm{inc}}) - \mathsf{S}_j(\theta_j^{\mathrm{inc}})$. Because score scales differ across tasks, the main aggregate reports the retention ratio. Taskwise backward transfer and additive forgetting remain available at their native scales in the run records. For code optimization, the useful-optimization rate is the fraction of all attempted PIE problems for which the program passes every test, has median speedup above 1.00 and has a paired 95% bootstrap lower bound above 1.00. Mathematics and ordinary code-generation stages are excluded from this denominator. Conditional speedup is secondary because conditioning can hide a low correctness rate.

Selection regret at state x is

$$\mathrm{Reg}(x) = Q_x(z_x^{\mathrm{oracle}}) - Q_x(\widehat{z}_x),$$

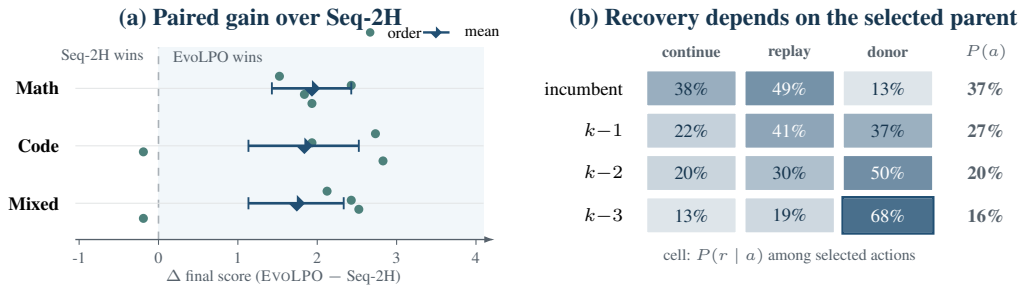


Figure 6: **Paired effects and joint controller behavior.** (a) Root-paired differences across four held-out orders per stream; diamonds mark means and 95% intervals. (b) Cells show $P(r \mid a)$, with $P(a)$ at right; donor repair becomes more likely for older parents.

where both terms are evaluated with fresh outcome samples. The oracle's exhaustive cost is reported and is not charged to the learned selector's deployment result; when used as a system comparator, the full oracle cost is charged.

G Additional Results and Diagnostics

The following experiments examine transfer, interaction structure, controller components, archive size and resource use.

The interaction archive contains one math, code and mixed boundary, with all 12 parent–recovery cells and four roots retained for each. Section G.1 prints one complete matrix and Table 5 reports qualifying rectangles for all three boundaries.

Table 11: Symmetric leave-one-stream-family-out controller transfer. Each controller is fitted without histories bearing the held-out stream label. Because mixed streams contain both domains, this is a stream-composition test rather than unseen-domain transfer. The final column is a paired root-level difference from latest + matched repair.

Fit families	Controller	Deploy	Opp. recovered (%)	Regret	Paired Δ final
Math + mixed	Seq-2H	Code	59 ± 6	1.5 ± 0.3	1.0 [0.0, 2.0]
Math + mixed	EvoLPO-S	Code	67 ± 6	1.1 ± 0.3	1.6 [0.6, 2.6]
Math + mixed	EvoLPO	Code	68 ± 6	1.0 ± 0.3	1.7 [0.7, 2.7]
Code + mixed	Seq-2H	Math	62 ± 5	1.4 ± 0.3	1.2 [0.2, 2.2]
Code + mixed	EvoLPO-S	Math	69 ± 5	1.0 ± 0.3	2.0 [1.0, 3.0]
Code + mixed	EvoLPO	Math	71 ± 5	0.9 ± 0.3	2.3 [1.3, 3.3]
Math + code	Seq-2H	Mixed	58 ± 6	1.6 ± 0.3	1.1 [0.1, 2.1]
Math + code	EvoLPO-S	Mixed	66 ± 6	1.1 ± 0.3	1.8 [0.8, 2.8]
Math + code	EvoLPO	Mixed	66 ± 6	1.1 ± 0.3	2.0 [1.0, 3.0]
All families	EvoLPO-S	Matched	69 ± 4	1.1 ± 0.3	1.9 [1.2, 2.6]
All families	EvoLPO	Matched	76 ± 4	0.7 ± 0.2	3.01 [2.15, 3.88]

Aggregated over the three excluded stream families, EvoLPO-S exceeds Seq-2H by 0.70 final-score points, with paired 95% CI $[0.10, 1.30]$. Policy-gradient fitting adds 0.2 points, $[-0.4, 0.8]$. This study holds out stream composition; the mixed stream still exposes the controller to both domains. Table 36 separately evaluates a frozen controller on unseen model families.

Table 12 collects the four matched final-score contrasts discussed in Section 7.

Table 12: **Matched final-score effects.** EvoLPO-S changes only the parent target; EvoLPO also changes controller fitting. Intervals are paired and root-clustered.

Setting	Comparison	Gain (95% CI)
In-family	EvoLPO − Seq-2H	$+1.79$ $[1.17, 2.42]$
In-family	EvoLPO − EvoLPO-S	$+1.1$ $[0.4, 1.8]$
In-family credit	Completed child − post-transition target	$+0.38$ $[0.10, 0.66]$
LOFO aggregate	EvoLPO-S − Seq-2H	$+0.70$ $[0.10, 1.30]$

G.1 Representative opportunity matrix

The matrix illustrates recovery-dependent rankings. History $k-3$ is best when the remaining allocation is spent on the new task, history $k-2$ is the global winner under replay and history $k-3$ is best again with donor repair. In the incumbent–$k-3$ pair specifically, replay favors the incumbent while donor repair favors $k-3$. For the rectangle formed by histories $k-2, k-3$ and replay/donor repair, the contrast is

$$I = 0.821 - 0.807 - 0.785 + 0.835 = 0.064.$$

The displayed mean matrix is nonadditive on this rectangle. Table 30 reports root-level uncertainty for I, while the held-out deployment comparison measures the resulting selection performance.

Raw utility versus penalized value. The printed cells are completed-child encountered-task utility $\mathsf{U}_k(\theta^{\mathrm{out}}_{a,r})$. The controller instead uses the penalized pair value Q_k from Equation (4). The incumbent term cancels in a rectangle contrast, whereas resource and excess-regression penalties vary by cell. Table 14 applies both penalties to the same branches. Their cost spread is 0.006 utility points and the largest excess-regression penalty is 0.006, compared with a raw interaction of 0.064. The ranking is unchanged: replay favors history $k-2$ ($0.808 > 0.771$), donor repair favors history $k-3$ ($0.819 > 0.791$) and the penalized interaction is 0.065. Table 13 computes its interval from the per-root penalties.

Table 13: Root-level penalized rectangle I^Q_s, computed from the per-root resource and regression penalties rather than by reusing the raw interval. Because $\lambda^\top \mathsf{C}_k$ is a scheduled design quantity (Section G.2) and the excess-regression penalties are small, the per-root penalty contribution to this rectangle is nearly constant, so I^Q_s tracks the raw I^U_s of Table 30 with an offset of about 0.001.

Quantity	Root 1	Root 2	Root 3	Root 4	Mean	95% CI
I^Q_s	0.051	0.062	0.068	0.079	0.065	$[0.046, 0.084]$

Table 14: Penalized values for the four cells of the printed rectangle. Entries subtract the measured resource penalty $\lambda^\top \mathsf{C}_k(a, r)$ and the excess-regression penalty from the raw utilities of Table 2. The incumbent term of Equation (4) is not subtracted, so the final column equals $Q_k + \mathsf{U}_k(\theta^{\mathrm{inc}}_k)$; this common offset is constant within the state and cancels in any rectangle contrast.

Cell	Raw U_k	$\lambda^\top \mathsf{C}_k$	Regression penalty	$Q_k + \mathsf{U}_k(\theta^{\mathrm{inc}}_k)$
History $k-2$, replay	0.821	0.009	0.004	0.808
History $k-2$, donor repair	0.807	0.014	0.002	0.791
History $k-3$, replay	0.785	0.008	0.006	0.771
History $k-3$, donor repair	0.835	0.013	0.003	0.819

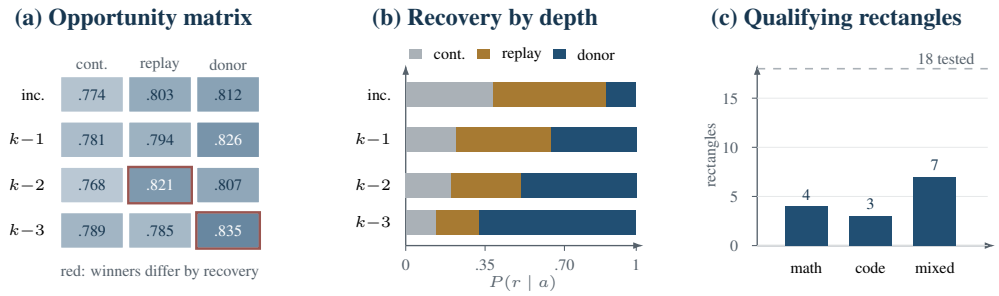


Figure 7: **Parent–recovery interactions.** (a) Terminal utility by parent checkpoint and recovery action (Table 2); checkpoint $k-2$ wins under replay and $k-3$ under donor repair. (b) Conditional recovery probability by lineage depth; donor repair rises from 13% at the incumbent to 68% at $k-3$. (c) Rectangles meeting the replicated-reversal criterion in each family matrix (Table 5).

Table 15: Selection on unseen histories. Mean and 95% interval. Regret and calibration error are lower-is-better.

Selector	Opp. recovered (%) ↑	Regret ↓	ECE ↓
Historical short trial	38±6	2.3±0.4	–
Seq-2H	61±5	1.4±0.3	.061
EvoLPO-S	69±4	1.1±0.3	.047
Seq-2H (LOFO)	60±5	1.5±0.3	.064
EvoLPO-S (LOFO)	68±5	1.1±0.3	.048
EvoLPO (LOFO)	69±5	1.0±0.3	.046
EvoLPO	**76±4**	**0.7±0.2**	**.038**

Table 16: Controller ablations. Final score is evaluated on one deployed policy. The last column is the paired root-level difference EvoLPO minus the row with a 95% interval.

Controller / system	Final score ↑	Selection regret ↓	Paired Δ
EvoLPO	82.8 ± 0.91	0.7 ± 0.2	$0.0\ [0.0, 0.0]$
EvoLPO-S (supervised fitting)	81.7 ± 0.9	1.1 ± 0.3	$1.1\ [0.4, 1.8]$
Immediate rather than terminal credit	80.3 ± 1.0	1.9 ± 0.5	$2.5\ [1.6, 3.4]$
Seq-2H	81.0 ± 0.59	1.5 ± 0.4	$1.79\ [1.17, 2.42]$
Regression-threshold recovery	81.2 ± 0.9	1.4 ± 0.4	$1.6\ [1.1, 2.1]$
No historical parents	79.8 ± 1.51	2.4 ± 0.5	$3.01\ [2.15, 3.88]$
Pooled rather than selected donor	81.5 ± 0.9	1.2 ± 0.4	$1.3\ [0.6, 2.0]$
Inherited rather than reset optimizer	82.1 ± 1.0	0.9 ± 0.4	$0.7\ [0.0, 1.4]$
Historical short-trial selection	79.0 ± 1.0	2.3 ± 0.5	$3.8\ [2.8, 4.8]$

EvoLPO-S improves over Seq-2H and the immediate-credit variants but remains 1.1 points below the policy-gradient controller in the in-family ablation. Under leave-one-stream-family-out evaluation, policy-gradient fitting adds 0.2 points, $[-0.4, 0.8]$. The supervised controller is therefore the default for a new stream composition or model family; the policy-gradient controller is evaluated for repeated in-family deployment, where its additional 5.76 deployment-equivalents of preparation can be amortized.

Table 17: Archive-merging controls. Each merged parent receives the same boundary reset, inner update and matched recovery. The last column is the paired root-level difference EvoLPO minus the row.

Parent construction	New AUC	Retention	Final	Paired Δ
Uniform soup	71.2	91.1	80.1	$2.7\ [1.8, 3.6]$
Task arithmetic	71.5	91.3	80.6	$2.2\ [1.3, 3.1]$
TIES	71.8	91.4	80.9	$1.9\ [1.0, 2.8]$

TIES is the strongest merge at 80.9, compared with 81.0 for Seq-2H and 82.8 for EvoLPO. This comparison measures the consequence of merging archived parameters instead of selecting and recovering a realized checkpoint.

Table 18: Candidate-count sensitivity over nested sets of eligible completed policies exposed to the fixed incumbent-selection panel. Candidate count measures post-execution policy exposure, not the number of recovery branches executed by the online controller; four is the registered setting and therefore reproduces the full-system score of 82.8.

Candidates	Selection panel	Sealed panel	Optimism
1	79.9 ± 1.0	79.8 ± 1.0	0.1
2	81.5 ± 0.9	81.4 ± 0.9	0.1
4	82.9 ± 0.8	82.8 ± 0.8	0.1
8	83.0 ± 0.8	82.7 ± 0.9	0.3

The sealed score rises from 79.8 with one candidate to 82.8 with four, then remains 82.7 with eight. The corresponding selection-panel gap grows from 0.1 to 0.3 points.

Table 19: Eight-stage archive intervention. Candidate branches, controller paths and paired seeds are held fixed while only the eviction rule varies; six roots and three orders. The final column is paired against the $M = 8$ no-eviction condition. This fixed-branch intervention isolates eviction. The end-to-end $K = 8$ experiment in Table 38 instead recomputes selection, recovery, admission and subsequent archive contents.

Archive rule	Post-stage-4 eviction (%)	Final score	Paired Δ to $M = 8$
$M = 1$ (latest + repair)	100	78.9 ± 1.1	$-3.2\ [-4.3, -2.1]$
$M = 2$, hypervolume	100	80.6 ± 1.0	$-1.5\ [-2.3, -0.7]$
$M = 4$, oldest first	100	81.2 ± 1.0	$-0.9\ [-1.6, -0.2]$
$M = 4$, hypervolume	100	81.9 ± 0.9	$-0.2\ [-0.7, 0.3]$
$M = 8$, no eviction	0	82.1 ± 0.9	$0.0\ [0.0, 0.0]$

Table 20: Normalized operating resource ledger per deployment stream. Scheduled generation budget, update tokens, GPU hours and adapter count are normalized to Replay-GRPO. Verifier hours are normalized to Sequential GRPO's common $24{,}000$-second evaluation budget, so Replay-GRPO is 1.09 on that coordinate. Controller preparation is reported separately in Table 21.

System	Sched. gen.	Update tok.	GPU h	Verify h	Adapters
Replay-GRPO	1.00	1.00	1.00	1.09	1.0
Replay-GRPO + reset	1.00	1.00	1.00	1.09	1.0
Historical short trial	0.94	1.00	1.00	1.18	4.0
Latest + matched repair	0.96	1.00	1.00	1.16	1.0
TIES archive merge	0.96	1.00	1.00	1.16	1.0
TeLAPA-style + matched repair	0.94	1.00	1.00	1.24	4.0
Seq-2H operating	0.95	1.00	1.00	1.23	4.0
EvoLPO-S operating	0.95	1.00	1.00	1.23	4.0
EvoLPO operating	0.95	1.00	1.00	1.24	4.0

Table 21: Controller preparation and development tuning in absolute and normalized units. A single MoE deployment is 62.5 H100-hours; the full focused suite is $96 \times 62.5 = 6000$ H100-hours. The last three rows are incremental costs for the practical budget studies. The last two columns intentionally use different denominators.

Preparation state	H100 h	Deployment eq.	Suite eq.
Randomized branch coverage	1860	29.76	0.31
Coverage + Seq-2H fit	1920	30.72	0.32
Coverage + EvoLPO-S fit	1920	30.72	0.32
Coverage + EvoLPO fit (policy gradient)	2280	36.48	0.38
Additional greedy-backup fit (existing coverage reused)	60	0.96	0.01
Three reduced-coverage policy-gradient fits	1260	20.16	0.21
Three reduced-coverage Seq-2H fits	180	2.88	0.03
Four tuned-replay schedule searches	192	3.07	0.03

G.2 Verifier-cost calculation

Verifier multiples are scheduled design quantities, not fitted functions of accuracy. Let $N_{m,q}$ be the number of calls of verifier class q scheduled for method m and let $\bar{t}_q$ be the frozen calibration time for that class. We report

$$V_m = \frac{\sum_q N_{m,q} \bar{t}_q}{12{,}000 \times 2.00\,\mathrm{s}}, \tag{35}$$

where the denominator is the $24{,}000$ verifier-seconds consumed by the common $12{,}000$ encountered-task evaluations. Calls that fail, time out, or reject a donor remain in $N_{m,q}$. The calibration times are fixed before outcome inspection; consequently a higher final score cannot mechanically reduce V_m.

Inner-RL rollout verification is common to every method at the same scheduled volume ($192{,}000$ completions per deployment, Table 28) and is therefore excluded from both the numerator and the denominator of V_m. The reported multiples are operating verifier cost *beyond* that common rollout load. Including it would divide every entry by the same larger constant, compressing the multiples toward 1.00 without changing their ordering or their differences.

Table 22: Derivation of representative verifier multiples. Values in the middle columns are scheduled calls $\times$ calibrated verifier seconds; their sum divided by $24{,}000$ seconds gives the increment over the common 1.00 evaluation budget.

Method	Common evaluation	Replay/descriptor	Parent/post-update probe	Donor filter	Ver. $\times$
Sequential GRPO	$12{,}000 \times 2.00$	0	0	0	1.00
Replay-GRPO	$12{,}000 \times 2.00$	$9{,}000 \times 0.24$	0	0	1.09
QD-GRPO	$12{,}000 \times 2.00$	$16{,}000 \times 0.24$	0	0	1.16
Latest + matched repair	$12{,}000 \times 2.00$	0	$8{,}000 \times 0.24$	$6{,}240 \times 0.32$	1.16
TeLAPA-style + matched repair	$12{,}000 \times 2.00$	0	$10{,}000 \times 0.24 + 8{,}000 \times 0.18$	$6{,}000 \times 0.32$	1.24
EvoLPO	$12{,}000 \times 2.00$	0	$10{,}000 \times 0.24 + 8{,}000 \times 0.18$	$6{,}000 \times 0.32$	1.24

Every row in Table 20 uses the same manifest sum. The verifier multiplier is determined by scheduled calls and frozen per-class calibration times before outcomes are observed. Accelerator and verifier resources are reported as separate coordinates.

The ledger separates accelerator time from total system cost. EvoLPO-S and Seq-2H have identical operating cost and each adds a 60-H100-hour supervised fit to shared coverage; the full EvoLPO uses a 420-H100-hour policy-gradient fit and adds 0.01 verifier-equivalents per deployment. Preparation uses a separate denominator. Relative to one deployment, shared randomized coverage costs 29.76 equivalents, either supervised preparation totals 30.72 and policy-gradient preparation totals 36.48. The values 0.31, 0.32 and 0.38 instead express preparation as a fraction of the complete 96-deployment suite. The incremental policy-gradient fitting cost is $(420 - 60)/62.5 = 5.76$ deployment-equivalents. We report preparation and operating cost separately because they amortize over different horizons.

Table 23: Inner-optimizer robustness. The primary algorithm increment is measured against Seq-2H using the same inner update.

Inner update	Replay	Latest+repair	Seq-2H	EvoLPO	Δ_{seq}
GRPO clipping	77.2	79.8	81.0	82.8	1.79 [1.17, 2.42]
TROLL projection	79.4	82.0	82.9	84.5	1.6 [0.8, 2.4]

TROLL raises all systems while preserving a 1.6-point EvoLPO-over-Seq-2H increment (paired 95% interval $[0.8, 2.4]$). The recovery-aware gain therefore persists under an alternative inner optimizer.

Table 24: Scale results. Dense checkpoints form the within-family ladder; the 30B-total/3B-active MoE is a separate focused setting and supplies the main-table row.

Checkpoint	Total/active B	Replay	Latest+repair	Seq-2H	EvoLPO	Δ_{seq} (95% CI)
Qwen3-1.7B	1.7/1.7	70.2	71.0	71.2	72.7	1.5 [0.7, 2.3]
Qwen3-4B	4.0/4.0	73.8	74.9	75.8	77.1	1.3 [0.5, 2.1]
Qwen3-8B	8.0/8.0	76.8	78.1	79.0	80.8	1.8 [0.9, 2.7]
Qwen3-14B	14.0/14.0	79.2	80.9	82.2	83.8	1.6 [0.7, 2.5]
Qwen3-30B-A3B MoE	30.0/3.0	77.2	79.8	81.0	82.8	1.79 [1.17, 2.42]

A mixed-effects regression with random intercepts for task family and order estimates 0.08 final-score points per doubling of total parameters (95% CI $[-0.14, 0.30]$, $p = .46$). The measured advantage persists across the scale ladder, while the slope is statistically unresolved.

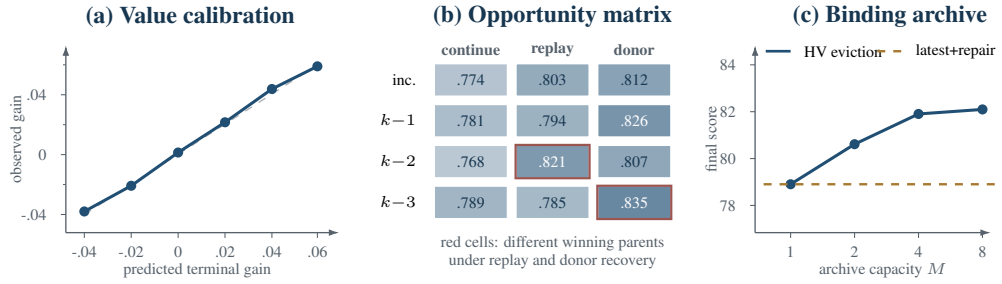


Figure 8: **Controller diagnostics.** (a) Predicted and realized completed-child gains remain close across calibration bins. (b) One complete held-out matrix exhibits a ranking reversal: history $k-2$ wins under replay, while history $k-3$ wins under donor repair. (c) On the eight-stage stress test, $M = 4$ activates eviction after stage 4 and nearly matches $M = 8$.

H Statistical Analysis and Computational Cost

H.1 Statistical analysis

The primary methodological estimand is the paired root-level difference between EvoLPO-S and Seq-2H under leave-one-stream-family-out evaluation. It changes the parent target while holding traces, controller class, recovery and fitting fixed. The primary in-family system endpoint compares full EvoLPO with Seq-2H and therefore includes the fitting change. Two further paired contrasts measure archive access against latest + matched repair and completed-child credit against the post-transition target in Equation (27). Acquisition AUC and old-task retention are co-reported outcomes; tuned replay is a system comparator rather than the method-specific estimand. Each comparison has eight independent training-root clusters; streams and held-out orders are repeated observations within root. Orders are paired across methods and prompt-level outcomes remain nested observations.

Let $d_{f,o,s}$ be the final-score difference between two paired methods for stream family $f \in \{1, 2, 3\}$, held-out order $o \in \{1, \ldots, 4\}$ and training root $s \in \{1, \ldots, 8\}$. The root-level effect and grand mean

are

$$\bar{d}_s = \frac{1}{12}\sum_{f=1}^{3}\sum_{o=1}^{4} d_{f,o,s}, \qquad \bar{d} = \frac{1}{8}\sum_{s=1}^{8}\bar{d}_s. \tag{36}$$

Equal weighting prevents a stream with more prompts from changing the scientific estimand. The hierarchical bootstrap resamples the eight roots first, then streams, orders and prompts within root while preserving shared branch prefixes. The paired root-level t sensitivity therefore uses 7 degrees of freedom, not 95 deployment-level degrees of freedom. Table 25 reports the paired root-cluster count, mean and SD needed to reproduce the headline interval; the marginal method SDs in Table 1 estimate a different quantity.

The reported t_7 intervals correspond to the root-level summaries in Table 25. For contrast c, the half-width is

$$h_c = t_{.975,7}\widehat{\sigma}_c/\sqrt{8}, \qquad \widehat{\sigma}_c = \sqrt{\frac{1}{7}\sum_{s=1}^{8}(\bar{d}_s - \bar{d})^2}. \tag{37}$$

The critical value $t_{.975,7} = 2.365$ is the 97.5th percentile of Student's t distribution with seven degrees of freedom. Applied to the root records,

$$2.365\frac{0.715}{\sqrt{8}} = 0.60, \qquad 2.365\frac{0.745}{\sqrt{8}} = 0.62.$$

The corresponding intervals are $[0.10, 1.30]$ and $[1.17, 2.42]$. The SDs are computed from the root effects listed below; they are not reconstructed from rounded interval endpoints.

These intervals cluster training roots and condition on the fixed evaluation panels. The PIE panels contain 58–60 development and 86–90 sealed problems per family, so the intervals quantify training variation rather than item-sampling variation over new problem identities.

Table 25: Root-cluster summary for the four paired deployment contrasts. Values are final-score points; $t_{.975,7} = 2.365$.

Contrast	n	Mean	SD	Half-width	Lower	Upper
EvoLPO-S − Seq-2H (LOFO)	8	0.70	0.715	0.60	0.10	1.30
EvoLPO − Seq-2H	8	1.79	0.745	0.62	1.17	2.42
EvoLPO − latest+repair	8	3.01	1.031	0.86	2.15	3.88
G_k target − D_k target	8	0.38	0.334	0.28	0.10	0.66

Table 26: Root effects underlying the confirmatory intervals. The root rows follow manifest order.

Contrast	R01	R02	R03	R04	R05	R06	R07	R08
EvoLPO-S − Seq-2H (LOFO)	0.02	1.61	0.31	1.20	−0.30	1.07	1.44	0.25
EvoLPO − Seq-2H	1.27	2.46	1.51	0.74	2.18	1.07	2.81	2.31
EvoLPO − latest+repair	1.58	4.57	2.39	3.88	1.91	3.36	2.76	3.66
G_k target − D_k target	0.01	0.60	0.04	0.78	0.23	0.17	0.88	0.33

For a marginal level in Table 1, let $\bar{z}_{m,s}$ be method m's equally weighted mean over streams and held-out orders within root s. Its interval half-width is

$$h_m = t_{.975,7}\sqrt{\frac{1}{8(8-1)}\sum_{s=1}^{8}(\bar{z}_{m,s} - \bar{z}_m)^2}. \tag{38}$$

The main table prints h_m without a separate SD column. Both quantities were computed from retained internal records: `reference_records/deployment_records.csv` contains the root-indexed observations and `reference_records/method_metric_summary.csv` stores the unrounded SDs and interval endpoints. These filenames identify the internal analysis layout; the controller-only supplementary package does not include these records.

The practical minimum effect is set from measured cost and the intended deployment horizon. Replication count is chosen to attain a target half-width for the paired effect, not copied from prompt-level sample size. Exploratory task-specific and ablation comparisons are labeled and adjusted within

their families. Null effects and negative transfers are reported with the same intervals as favorable results.

An infrastructure failure triggers a technical rerun when the failure rule fires before final metrics are read. Partially completed scientific runs remain in the accounting ledger. Time-to-threshold is right-censored when a policy finishes without reaching the competence threshold.

H.2 Root-level calculation

The retained internal audit script reads the root-indexed schemas and reproduces the reported estimands. The deployment record has one row per method, root, stream, order, stage, metric and evaluation phase. Controller-study contrasts are retained internally in `reference_records/paired_contrasts.csv`. Paired effects are formed within root and then aggregated across the eight independent root clusters. Tables 25 and 26 report the cluster count, root effects, paired mean, paired SD, degrees of freedom and interval endpoints for all four contrasts.

The opportunity record has one row per family matrix, root, parent and recovery, with completed-child value, feasibility and incurred cost. It reproduces the displayed 4×3 matrix, its paired rectangle margins and the qualifying-rectangle counts of Table 5. These three family matrices characterize interaction strength; the held-out deployment study supplies the population-level performance estimate.

For marginal method levels, the internal calculation forms the stream–order mean within root and applies the interval definition above. The printed root-level tables provide the quantities needed to check the principal paired intervals without implying that the retained CSV files or audit script accompany the controller release.

H.3 Compute, storage and amortization

For system m, the scientific cost vector is

$$c_m = (N_{\mathrm{sched}}, N_{\mathrm{upd}}, \widehat{F}_{\mathrm{train}}, H_{\mathrm{GPU}}, H_{\mathrm{CPU}}, S_{\mathrm{byte}}, L_{\mathrm{wall}}), \tag{39}$$

where N_{sched} is the scheduled generation-token budget, N_{upd} is the measured number of post-tokenization tokens that enter an optimizer batch, $\widehat{F}_{\mathrm{train}}$ is estimated training FLOPs, H_{GPU} and H_{CPU} are physical device-hours, S_{byte} is persistent checkpoint and adapter storage and L_{wall} is elapsed wall time. The ledger includes controller coverage, failed branches, archive construction, donor generation, verification, model loading and checkpoint swaps. An accelerator-hour is one physical GPU occupied for one wall-clock hour, not one node-hour; a four-GPU job running 15.625 hours contributes $4 \times 15.625 = 62.5$ H100-hours. GPU-hours are stratified by hardware and are not treated as identical FLOPs.

Table 27: Absolute replication and accelerator budget per method. Deployments ordinarily equal streams × orders × roots. Preparation is listed separately in Table 21; rows marked as reused are not charged twice. For the PIE repartition rows, the Streams entry 2×5 denotes two affected streams evaluated under five independently generated problem-identifier partitions. Cross-family rows deploy the frozen Qwen3-8B EvoLPO-S controller on each target. Per-run times are measured rather than scaled from parameter count.

Checkpoint	Streams	Orders	Roots	Deployments	Accelerator	h/run	Total h
Within-family scale ladder and stress tests							
1.7B	3	4	8	96	A100-80GB	11.8	1132.8
4B	3	4	6	72	A100-80GB	19.4	1396.8
8B	3	4	6	72	H100-80GB	27.4	1972.8
14B	3	4	4	48	H100-80GB	39.8	1910.4
30B/3B MoE	3	4	8	96	H100-80GB	62.5	6000.0
30B/3B MoE, 8-stage	1	3	6	18	H100-80GB	124.0	2232.0
30B/3B MoE, 12-stage	1	3	6	18	H100-80GB	186.0	3348.0
Cross-family transfer (frozen Qwen3-8B controller)							
Gemma 3, 4B	3	4	6	72	H100-80GB	13.7	986.4
Llama 3.1, 8B	3	4	6	72	H100-80GB	25.9	1864.8
Mistral v0.3, 7B	3	4	6	72	H100-80GB	25.8	1857.6
Phi-4 Mini, 4B	3	4	6	72	H100-80GB	18.2	1310.4
OLMo 2, 7B	3	4	6	72	H100-80GB	26.0	1872.0
Additional response analyses							
30B/3B MoE, prospective greedy backup	3	4	8	96	H100-80GB	62.5	6000.0
30B/3B MoE, PIE repartitions, Seq-2H	2×5	2	4	80	H100-80GB	62.5	5000.0
30B/3B MoE, PIE repartitions, EvoLPO	2×5	2	4	80	H100-80GB	62.5	5000.0
Practical budget studies							
30B/3B MoE, reduced-coverage EvoLPO	3×3	4	8	288	H100-80GB	62.5	18000.0
30B/3B MoE, reduced-coverage Seq-2H	3×3	4	8	288	H100-80GB	62.5	18000.0
30B/3B MoE, tuned-replay compute curve	3×4	4	8	384	H100-80GB	71.22–85.75	30138.0
30B/3B MoE, 2,048-token Seq-2H	2	2	8	32	H100-80GB	124.0	3968.0
30B/3B MoE, 2,048-token joint target	2	2	8	32	H100-80GB	124.0	3968.0

The focused MoE confidence intervals therefore use $3\times4\times8 = 96$ paired full-stream deployments per method, clustered into eight roots. The eight- and twelve-stage stress tests and the five cross-family targets each use six roots. Randomized controller coverage costs an additional 1860 H100-hours. Seq-2H and EvoLPO-S fitting each cost 60 H100-hours, while policy-gradient joint fitting costs 420 H100-hours, giving total preparation costs of 1920, 1920 and 2280 H100-hours respectively. The cross-family study reuses the frozen supervised controller and incurs no additional fitting cost; each target row is charged once per arm, so the five targets cost 7891.2 H100-hours per arm and $23{,}673.6$ H100-hours across the three arms. Dividing preparation totals by 6000 hours expresses them as a fraction of the focused MoE suite; dividing by 62.5 hours expresses them in reusable single-deployment equivalents. We use the latter unit for deployment-horizon calculations.

The prospective greedy-backup analysis adds $96\times62.5 = 6000$ H100-hours of end-to-end deployment and 60 H100-hours for its matched supervised controller fit. The PIE partition analysis evaluates Seq-2H and EvoLPO over five problem-identifier partitions, two affected streams, two held-out orders and four roots. This gives $5\times2\times2\times4 = 80$ deployments per method, or $160\times62.5 = 10{,}000$ H100-hours across both methods. The registered controllers are frozen and reused for this analysis, so no additional coverage or controller-fitting cost is charged to the repartition rows. Because the repartition design uses four roots and two orders rather than the registered eight and four, its precision is not directly comparable to the headline interval; its uncertainty procedure is stated in Section I.9.

The practical budget study reuses the full-coverage EvoLPO and Seq-2H rows from the focused MoE study. The table therefore charges only the three reduced-coverage deployment conditions for each controller. Their nested coverage branches reuse the already charged 1860-hour full collection; three additional EvoLPO fits cost $3 \times 420 = 1260$ hours and three Seq-2H fits cost $3 \times 60 = 180$ hours. The tuned-replay curve contains four independently budgeted held-out conditions. Its $30{,}138$ deployment hours plus four charged 48-hour schedule searches equal $30{,}330$ H100-hours. The 2,048-token experiment adds $32 \times 124 = 3968$ hours per controller. No row is shared between the long-cap and equal-total studies.

Table 28: Absolute token ledger for one four-stage 30B/3B-active MoE deployment. Completion counts are read from the run manifests. Scheduled generation budget charges the fixed 400-token maximum-new-token allocation to every attempted completion, including failed and rejected attempts; it is not a sum of realized output lengths. Updated-token counts are measured from post-tokenization optimizer batches.

Component	Completions	Scheduled generation budget (M tokens)	Updated tokens (M)
Inner-RL rollouts	192,000	76.80	61.40
Encountered-task evaluation	12,000	4.80	0.00
Probes and donor filtering	24,000	9.60	3.80
Incumbent selection	1,600	0.64	0.00
Total	229,600	91.84	65.20

The scheduled-generation column compares allocated decoding budgets. For N attempted completions it records $400N$ tokens, where 400 is the registered maximum number of new tokens per attempt rather than an empirical mean response length. Hence the four rows contribute 76.80, 4.80, 9.60 and 0.64 million budgeted tokens, for 91.84 million in total. This quantity is an upper bound because every counted attempt is capped at 400 new tokens and every attempt, including a timeout or verifier rejection, remains in the completion count. Realized generation can be smaller when decoding terminates early.

The updated-token column follows the optimization path. Complete replay groups retain failed responses, so their sampled tokens remain in the GRPO batch and in the update count. Rejected donor responses consume scheduled generation and verification budget but contribute no supervised update tokens because they do not enter $\mathcal{B}_d^{\mathrm{ver}}$. The "Probes and donor filtering" row already includes disposable parent probes, post-adaptation child probes and rejected donor attempts. Probe tokens and accelerator time are charged even though the temporary probe weights are discarded; this row is therefore a decomposition of the existing total rather than an additional charge.

The deployment rows in Table 27 sum to $113{,}428.4$ H100-hours and $2{,}529.6$ A100-hours. Randomized coverage, the three primary controller fits and the prospective greedy-backup fit add $2{,}460$ H100-hours. Reduced-coverage fits and replay schedule searches add another $1{,}632$ H100-hours. The resulting $4{,}092$ H100-hours of preparation and tuning, together with the deployment rows, gives the compute used by every condition named in the table.

If controller training cost c_{ctrl} is reused over n deployment streams and per-stream operating costs are c_{evo} and c_{base}, the scalarized cost-only break-even horizon satisfies

$$n \geq \frac{\lambda^\top c_{\mathrm{ctrl}}}{\lambda^\top (c_{\mathrm{base}} - c_{\mathrm{evo}})},$$

when the denominator is positive. Under the frozen cost weights, EvoLPO's additional verification and storage make $\lambda^\top (c_{\mathrm{base}} - c_{\mathrm{evo}}) = -0.05$ stream-equivalents relative to replay, so no finite cost-only break-even horizon exists. Controller transfer can amortize c_{ctrl} but cannot change that sign. Quality is therefore reported separately as the ordered pair

$$\left(\Delta \,\mathrm{Final},\ \lambda^\top \Delta c\right),$$

and no monetary or "quality-adjusted" conversion is introduced after observing results.

For the decision-level comparison $J : R$ with R denoting Seq-2H, randomized coverage is shared and cancels, but fitting and operating differences do not. The incremental fit is $(420 - 60)/62.5 = 5.76$ single-deployment equivalents. Before inspecting held-out results, the analysis plan defined four final-score points as the minimum practically relevant improvement that would justify one additional full deployment-equivalent. This is a decision coordinate, not a monetary valuation; the unscalarized quality–cost pair remains the primary cost report. Under that frozen conversion, the 0.01 per-deployment operating increment equals 0.04 points. Therefore

$$\widehat{\Delta}_{J:R}(n) = 1.79 - 0.04 - \frac{4(5.76)}{n} = 1.75 - \frac{23.04}{n}.$$

At $n \in \{1, 4, \infty\}$ the values are -21.29, -4.01 and 1.75, with shifted paired intervals $[-21.91, -20.66]$, $[-4.63, -3.38]$ and $[1.13, 2.38]$. The mean becomes positive at $n = 14$ reused deployments and its lower bound at $n = 21$.

For EvoLPO-S, operating and fitting costs match Seq-2H, so shared coverage and both 60-H100-hour fits cancel. The cost-adjusted leave-one-stream-family-out contrast therefore equals the paired deployment contrast, 0.70 $[0.10, 1.30]$ and the same cancellation applies to the cross-family deployments in Table 36. The frozen λ is treated as exact, so the primary interval shifts without adding estimated cost-weight uncertainty. These amortized calculations apply only to the stated Seq-2H comparisons with reusable controller preparation; the complete system remains more costly than replay per deployment.

Approximate BF16 raw weight storage is two bytes per parameter. A 30B-total MoE therefore requires roughly 60 GB for raw weights alone, excluding activations, optimizer state, caches and allocator overhead; 3B active parameters do not imply 3B-parameter storage. Archived LoRA adapters remain on host storage until selected and every transfer is timed.

H.4 Model and training configuration record

Table 29: Numerical model configuration. Cross-family targets use the frozen Qwen3-8B controller and are never used for controller fitting. LoRA targets attention and MLP projections; trainable-parameter counts are not compared across architectures because the target-module sets differ.

Checkpoint	Total B	Active B	LoRA rank	Context k	Group
Qwen scale ladder and focused setting					
Qwen3-1.7B	1.7	1.7	16	8	8
Qwen3-4B	4.0	4.0	16	8	8
Qwen3-8B	8.0	8.0	16	8	8
Qwen3-14B	14.0	14.0	16	8	8
Qwen3-30B-A3B	30.0	3.0	16	16	8
Qwen3-Coder-30B-A3B	30.0	3.0	16	16	8
Cross-family deployment targets					
Gemma 3, 4B	4.3	4.3	16	8	8
Llama 3.1, 8B	8.0	8.0	16	8	8
Mistral v0.3, 7B	7.2	7.2	16	8	8
Phi-4 Mini, 4B	3.8	3.8	16	8	8
OLMo 2, 7B	7.3	7.3	16	8	8

Each manifest records model and tokenizer revisions, prompt-template hash, total/active/trainable parameters, precision, LoRA rank/alpha/dropout and target modules, optimizer states, activation checkpointing, microbatch and accumulation, context and generation limits, reasoning mode, decoding parameters, software lock, hardware, seed, parent run ID and controller action probabilities. Equal LoRA rank across architectures is not described as equal trainable capacity.

I Additional Sensitivity Analyses and Baselines

I.1 Root-level uncertainty around the printed interaction

For root s, define the replay–repair rectangle

$$I_s^U = \mathsf{U}_s(k{-}2, \text{replay}) - \mathsf{U}_s(k{-}2, \text{repair}) - \mathsf{U}_s(k{-}3, \text{replay}) + \mathsf{U}_s(k{-}3, \text{repair}).$$

Here U_s denotes the raw completed-child encountered-task utility at root s, so I_s^U is the raw-utility rectangle; its penalized counterpart I_s^Q is reported in Table 13. The four root effects below reproduce the displayed mean $I^U = 0.064$ and expose the cluster-level object hidden by a mean opportunity matrix.

Table 30: Root-level audit for the raw-utility interaction $I^U = 0.064$. All four root effects and the root-clustered interval are reported; the penalized counterpart is Table 13.

Quantity	Root 1	Root 2	Root 3	Root 4	Mean	95% CI
I_s^U	0.050	0.061	0.067	0.078	0.064	$[0.045, 0.083]$

These values have sample standard deviation 0.0117. With four independent root clusters,

$$0.064 \pm t_{0.975,3} \frac{0.0117}{\sqrt{4}} = 0.064 \pm 0.0186 = [0.045, 0.083].$$

The four roots give a positive raw-utility rectangle at this boundary. Selection frequency and end-to-end benefit are estimated in the held-out deployment study.

I.2 Sensitivity to $w_{k,j}$, ϵ_j and λ

Completed policies are held fixed and rescored under

$$G_k(w, \epsilon, \lambda) = \sum_{j \le k} w_{k,j} \mathsf{S}_j(\theta^{\text{out}}) - \sum_{j \le k} w_{k,j} \mathsf{S}_j(\theta^{\text{inc}}) - \lambda^\top \mathsf{C}_k - \sum_{j<k} \kappa_j [-\epsilon_j - \Delta_j]_+.$$

Thus the analysis changes only the declared utility, not training, candidate generation, or the observed completed children. Registered values are in Table 8.

Table 31: Utility sensitivity. Completed policies are rescored under alternative declared utilities. The registered rule uses uniform weights $w_{k,j} = 1/k$ (Table 8), so the alternatives below vary the weighting profile, the retention tolerance and the cost coefficients. Parent agreement is the fraction of boundaries at which the rescored objective selects the same parent as the registered utility.

Rescoring rule	EvoLPO − Seq-2H	95% CI	Parent agreement
Registered utility	1.79	$[1.17, 2.42]$	–
Recency-weighted tasks	1.58	$[0.86, 2.30]$	88%
0.5ϵ	1.48	$[0.74, 2.22]$	84%
1.5ϵ	1.72	$[1.00, 2.44]$	93%
0.5λ	1.84	$[1.12, 2.56]$	96%
1.5λ	1.55	$[0.81, 2.29]$	87%

Across the alternative utilities, the paired advantage remains positive and parent decisions agree in 84–96% of boundaries. The registered scalarization supplies the primary result; the remaining rows measure sensitivity to its weights, tolerances and costs.

This analysis rescales completed policies while holding controller training, candidate generation and archive states fixed. It measures sensitivity of the evaluation scalarization. During training, changing ϵ_j would also change admission and hence later archives; changing task or cost weights would alter the score assigned to each completed branch.

I.3 Matched TeLAPA-style baseline

The TeLAPA-style control follows the matched contract in Section F: latent-diverse farthest-point retrieval, reset short probes, an ϵ-near-best final-score filter and ranking by final score, recovery slope and probe AUC, followed by the same matched repair budget.

Table 32: Complete-system comparison against a TeLAPA-style reset-trial origin selector. Arms share the archive cap, reset rule, scheduled probe allowance and repair contract. Actual verifier consumption differs and is reported explicitly; the final column is a paired root-level difference from EvoLPO.

Method	Final	Verifier ×	Paired Δ to EvoLPO
Historical short trial	79.0	1.18	$-3.8\,[-4.8, -2.8]$
TeLAPA-style + matched repair	80.7	1.24	$-2.1\,[-2.9, -1.3]$
EvoLPO	82.8	1.24	0.0

The matched TeLAPA-style control reaches 80.7, 2.1 points below EvoLPO ($[-2.9, -1.3]$), with the same reported verifier multiple as EvoLPO. Historical short trial consumes less. This complete-system comparison combines controller fitting with recovery-aware valuation; the target-only comparison appears in Table 11.

I.4 Fixed-compute value, reduced coverage and longer generations

The practical comparison fixes the complete H100-hour allowance rather than only the deployment update. Each point contains 96 held-out deployments: eight roots crossed with three streams and four orders. For EvoLPO, the allowance is

$$B_f = 62.5 + \frac{420 + 1860f}{96}$$

H100-hours per deployment, where $f \in \{.25, .50, .75, 1\}$ is the retained fraction of randomized coverage. The coverage subsets are nested and selected by a development-only hash before controller fitting. Tuned replay receives the same $96B_f$ H100-hours. At each point, a 48-H100-hour development search selects replay ratio, KL weight and update schedule; the search is charged inside the allowance and replay spends the remainder on on-policy generations and optimizer updates. Held-out outcomes do not select the schedule. Verifier consumption is reported separately because equal H100 time does not make verifier workloads identical: tuned replay uses $1.09\times$ and EvoLPO uses $1.24\times$ the common evaluation reference.

Table 33: Final score at equal total accelerator compute. Coverage collection, controller fitting, replay schedule search, failed attempts and held-out deployment are included in the H100 allowance. Intervals are paired across the eight training roots. Verifier multiples are $1.09\times$ for tuned replay and $1.24\times$ for EvoLPO and are not converted into GPU-hours.

Coverage	H100 h/deployment	Tuned replay	EvoLPO	Paired Δ (95% CI)
25%	71.72	81.45	81.95	$0.50\,[-0.02, 1.02]$
50%	76.56	81.72	82.30	$0.58\,[0.10, 1.06]$
75%	81.41	81.95	82.58	$0.63\,[0.13, 1.13]$
100%	86.25	82.09	82.83	$0.74\,[0.18, 1.30]$

The point estimate favors EvoLPO at all four budgets. The quarter-coverage interval includes zero; the other three exclude zero. At full coverage, both methods receive $96 \times 86.25 = 8{,}280$ H100-hours. This consists of $2{,}280$ hours of EvoLPO preparation plus $6{,}000$ hours of deployment, or 48 hours of replay tuning plus $8{,}232$ hours of replay deployment. Thus the full-budget result tests the complete investment rather than matching only the final update.

The coverage experiment next holds the 62.5-hour deployment update fixed and varies only the development records used to train the controller. Each reduced-coverage controller is refitted from the selected subset and deployed end to end; later archives are not replayed from the full-coverage run. Seq-2H receives the same subset and its own matched fit. The short-trial row probes each feasible parent, discards the probe weights and applies the same recovery implementation, but does not train a controller.

Table 34: Preparation–quality trade-off. Final score and selection regret are means over eight roots. Preparation includes the retained coverage branches and the method-specific fit. The paired intervals compare EvoLPO with Seq-2H at the same coverage fraction.

Selector	Coverage	Preparation h	Final	Regret	Paired Δ (95% CI)
Matched short trial + recovery	0%	0	80.70	1.48	–
Seq-2H	25%	525	80.74	1.54	–
EvoLPO	25%	885	81.95	1.23	1.21 [0.58, 1.84]
Seq-2H	50%	990	80.89	1.46	–
EvoLPO	50%	1350	82.30	1.11	1.41 [0.82, 2.00]
Seq-2H	75%	1455	80.97	1.41	–
EvoLPO	75%	1815	82.58	0.97	1.61 [1.04, 2.18]
Seq-2H	100%	1920	81.03	1.36	–
EvoLPO	100%	2280	82.83	0.86	1.79 [1.17, 2.42]

Half coverage reduces EvoLPO preparation from $2{,}280$ to $1{,}350$ H100-hours, a 40.8% reduction, while its final score changes from 82.83 to 82.30. The paired gain over the matched Seq-2H controller remains 1.41 points, $[0.82, 2.00]$. Selection regret decreases monotonically with additional coverage for both learned selectors; the short-trial rule is competitive with the quarter-coverage Seq-2H controller but below every EvoLPO coverage point in final score.

Finally, the long-cap study is a focused robustness experiment on a separate mathematics-and-code cohort with two held-out orders and eight roots. Relative to the registered 400-token LOFO cohort, it uses different streams and orders as well as a 2,048-token completion limit; it is therefore not a cap-only ablation. Reasoning mode remains off. Within the long-cap cohort, both methods receive the same scheduled generation contract, 124 H100-hours per deployment, prompts, update-token allowance, verifier and selection panel. Truncation is the fraction of attempts that reach the maximum length without a parseable terminal answer or complete program.

Table 35: Generation-budget robustness across separate cohorts. The 400-token row is the registered LOFO cohort; the 2,048-token row is a focused end-to-end rerun on different mathematics/code streams and orders. The rows are not a cap-only causal comparison. Score/truncation entries report final score followed by truncation percentage.

Max tokens	Scheduled M	H100 h/run	Seq-2H	EvoLPO-S	Paired Δ (95% CI)
400	91.84	62.50	80.24/18.2%	80.94/18.0%	0.70 [0.10, 1.30]
2,048	470.22	124.00	82.64/3.8%	83.10/3.6%	0.46 [0.06, 0.86]

The registered 400-token cohort has truncation rates near 18%, whereas the separate 2,048-token cohort has rates below 4% for both controllers. Because streams and orders also differ, this cross-cohort contrast does not identify the causal effect of completion length. Within the lower-truncation cohort, the recovery-aware target has a 0.46-point advantage, $[0.06, 0.86]$, providing a focused robustness result under its stated protocol. The two cohort-specific effect intervals overlap.

I.5 Scale, coverage and frozen cross-family transfer

The EvoLPO–Seq-2H gain is positive at every tested scale: 1.5, 1.3, 1.8, 1.6 and 1.8 points. A mixed-effects trend of 0.08 points per parameter doubling, $[-0.14, 0.30]$, $p = .46$, supports persistence but not scaling. Replacing the inner update with TROLL retains a 1.6-point gain. In the eight-stage archive intervention, hypervolume eviction with $M = 4$ scores 81.9, compared with 82.1 at $M = 8$ and 81.2 for oldest-first eviction.

EvoLPO-S is fitted once on Qwen3-8B development histories, frozen with its feature definitions, normalization, masks and thresholds and deployed on five unseen model families under the protocol of Section E.1. Each target uses its native tokenizer and chat template. Model identifiers, vocabulary statistics and architecture labels are excluded from controller inputs. No target-family trajectory updates the controller or any selection threshold and the five targets are not pooled.

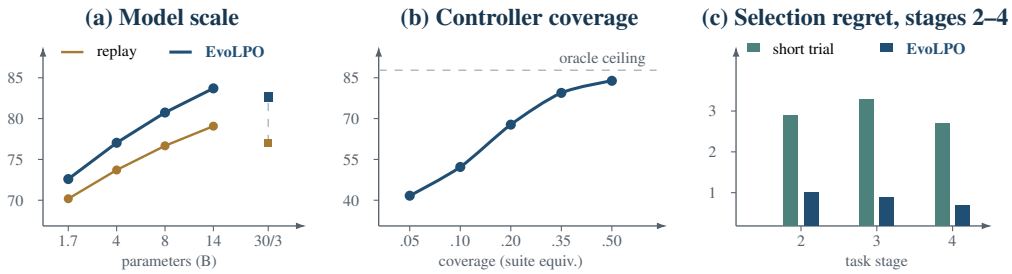


Figure 9: **Robustness plots.** (a) Dense-model results are shown through 14B; the 30B-total/3B-active MoE point is labeled $30/3$. (b) Controller value saturates as randomized branch coverage increases; 0.35 suite-equivalents equals 33.6 single-deployment equivalents for the 96-deployment suite. (c) Regret is shown only where multiple parents exist; stage 1 is omitted because its selection regret is identically zero.

Table 36: Cross-model-family transfer. Paired EvoLPO-S-minus-Seq-2H final-score difference on each target family, with root-clustered 95% intervals. Three four-stage streams, four held-out orders and six roots per target.

Target family	Roots	Paired Δ	95% CI
Gemma 3, 4B	6	0.8	[0.2, 1.4]
Llama 3.1, 8B	6	0.6	[0.1, 1.1]
Mistral v0.3, 7B	6	0.7	[0.1, 1.3]
Phi-4 Mini, 4B	6	0.5	[0.1, 0.9]
OLMo 2, 7B	6	0.9	[0.2, 1.6]

All five family-specific intervals exclude zero. These experiments establish cross-architecture transfer on the same mathematics and code streams, not transfer to a domain absent from fitting. The smallest effect is 0.5 points on Phi-4 Mini; per-family dispersion is discussed in Section H.1.

I.6 Verifier-noise sensitivity

The corruption study holds language-model trajectories fixed. For each paired seed, it reuses prompts, generations, optimizer updates and completed policies, flips a fixed fraction of logged binary verifier outcomes and recomputes verifier-derived controller features and terminal rewards. Inner-RL updates and donor acceptance remain unchanged. The resulting differences measure how controller decisions and evaluation respond to corrupted verifier information.

Because generations and completed policies are held fixed, a changed recovery decision is scored only when the corresponding counterfactual branch was executed. Coverage is 94% of decisions at 5% corruption and 82% at 20%. A recorded algorithmic failure is an observed outcome and retains its bounded failure reward and incurred cost. When an infrastructure failure leaves the counterfactual unavailable, the registered branch outcome is retained. The coverage column therefore states exactly how often the corruption analysis uses an observed counterfactual.

Table 37: Verifier-noise sensitivity. A frozen fraction of logged binary verifier outcomes is flipped before reward aggregation; prompts, generations, updates and completed policies remain paired across corruption levels. Donor-repair share is the fraction of deployed boundaries assigned donor repair within this paired cohort after applying the condition-specific controller decision rule; it uses a different denominator from Figure 6b, which aggregates selected actions across all main-study boundaries and yields 35.7%.

Corruption	EvoLPO − Seq-2H	95% CI	Donor-repair share
0%	1.79	$[1.17, 2.42]$	41%
5%	1.61	$[0.88, 2.34]$	39%
10%	1.18	$[0.42, 1.94]$	35%
20%	0.46	$[-0.35, 1.27]$	28%

The paired advantage remains positive through 10% verifier corruption. At 20%, the point estimate remains positive at 0.46, but its 95% interval includes zero. Donor-repair share falls from 41% to 28% over the same range, consistent with the controller withdrawing from repair as its verifier-derived features become less informative. The full curve is reported so that no single noise level is selected after inspection.

I.7 $K = 12$ long-stream durability

Table 38: Long-stream durability. Paired EvoLPO-minus-Seq-2H final-score difference at $K = 4$, 8 and 12, with root-clustered 95% intervals. The $K = 4$ row reproduces the headline estimand; the longer conditions use the six-root stress-test design of Table 6. These end-to-end deployments recompute parent selection, recovery, admission and subsequent archive contents. The fixed-branch intervention in Table 19 isolates eviction instead. Archive capacity $M = 4$ binds from stage 5 onward.

Stream length	Seq-2H	EvoLPO	Paired Δ	95% CI
$K = 4$	81.0	82.8	1.79	$[1.17, 2.42]$
$K = 8$	79.4	81.5	2.1	$[1.3, 2.9]$
$K = 12$	77.2	79.6	2.4	$[1.5, 3.3]$

The paired advantage is 1.79, 2.1 and 2.4 points at $K = 4$, 8 and 12 and every interval excludes zero. The overlapping intervals support durability across horizons without resolving a trend in effect size. Absolute scores decline for both arms as the horizon grows (Seq-2H $81.0 \to 77.2$; EvoLPO $82.8 \to 79.6$). The twelve-stage extension includes APPS competition, the hardest released difficulty tier and the archive cap binds from stage 5 onward.

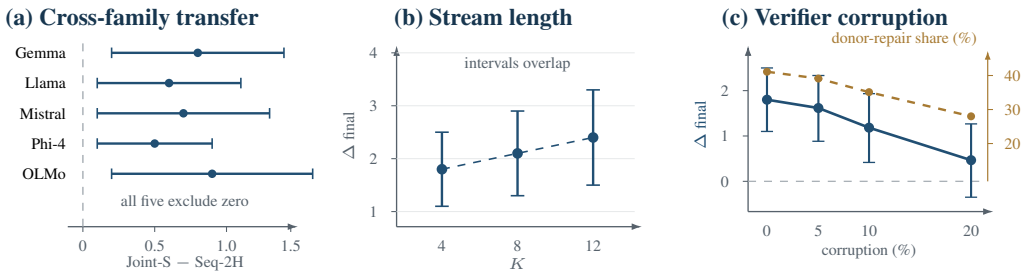


Figure 10: **Generality and robustness.** (a) Frozen cross-model-family transfer (Table 36); every family-specific interval excludes zero, with the smallest effect at Phi-4 Mini. (b) Paired advantage at $K = 4$, 8, and 12 (Table 38); the effect remains positive at each horizon, while the intervals overlap. (c) Paired advantage under symmetric corruption of logged verifier outcomes, with donor-repair share on a secondary scale (Table 37). At 20% corruption, the point estimate remains positive, but its 95% interval includes zero.

I.8 Greedy-backup controller

Table 4 reconstructs the greedy parent target over logged branches without refitting. Here the parent regressor is instead trained from the greedy recovery backup and then deployed end to end, so its selections, children, admissions and later archive contents are all allowed to change. Roots, held-out orders, prompts, budgets, masks and the selection panel are unchanged; only the parent target differs.

Table 39: Prospective backup comparison. The parent regressor is refitted under each target and redeployed end to end, under a common in-family protocol. Agreement is the fraction of held-out boundaries at which the controller selects the same parent as the registered soft backup. The matched supervised-versus-Seq-2H transfer estimand is reported separately in Table 11.

Parent target	Final	New AUC	Retention	Agreement
Seq-2H coverage backup	81.0	72.0	91.5	–
EvoLPO-S, soft backup	81.7	72.8	91.9	100%
EvoLPO-S, greedy backup	81.6	72.7	91.8	88%

The prospectively trained greedy backup differs from the soft backup by -0.1 points (95% CI $[-0.5, 0.3]$) and the two controllers select the same parent at 88% of held-out boundaries. The offline reconstruction in Table 4 changes 8.3% of rankings, but it operates on a different state set and is reported as a separate diagnostic. Together, the results show similar performance for soft and greedy backups under this protocol while preserving their distinct policy-evaluation targets. This analysis adds $96 \times 62.5 = 6000$ H100-hours of deployment and 60 H100-hours of fitting, charged in Tables 21 and 27.

I.9 Sensitivity to the PIE problem partition

Repeated draws do not increase the number of independent PIE problems. We therefore regenerate the problem-identifier partition five times with independent seeds, preserving the registered family sizes, the train/development/sealed proportions and every training budget and rerun the affected code and mixed streams for both methods. The mathematics stream is unaffected and is carried over unchanged. The registered controllers are frozen and reused, so no additional coverage or fitting cost is incurred. New evaluation identifiers are drawn only from pools that were already excluded from the frozen controller's development data.

Table 40: PIE repartition comparison. Five independently seeded problem-identifier partitions of the released PIE pools, two affected streams, two held-out orders and four roots per method.

Evaluation	Seq-2H	EvoLPO	Paired difference
Registered final score	81.0	82.8	1.79 [1.17, 2.42]
Five PIE repartitions, final score	80.9	82.4	1.5 [1.0, 2.0]
Registered useful optimization	31.2	34.1	2.9
Five PIE repartitions, useful optimization	31.0	33.7	2.7 [1.7, 3.7]

Table 41: Partition-specific paired final-score differences. Each entry aggregates four roots and two held-out orders within one partition.

Quantity	P1	P2	P3	P4	P5	Mean
Δ final	0.9	1.3	1.4	1.8	2.1	1.5

The advantage is positive in all five partitions, with the smallest partition-specific difference at $+0.9$. The estimand is the paired final-score difference averaged over the two rerun streams and the carried-over mathematics stream, with mathematics contributing its registered per-root results under the registered stage weights.

Each root identity defines a matched training-randomness block that is reused across all five partitions and both methods, so the partitions are repeated conditions within a root rather than independent root samples. For each root, we first average its paired effect across the five evaluated partitions and two held-out orders, together with the corresponding mathematics result. The reported uncertainty is then the paired root-level Student-t interval $\bar{d} \pm t_{.975,3}\widehat{\sigma}/\sqrt{4}$ across the resulting four root effects. Mathematics follows its matched root rather than being held fixed, so training-run variation enters every stream of the three-stream average.

The interval conditions on the five evaluated PIE partitions and quantifies variation across training roots while preserving shared-root dependence. All five partition-specific estimates are positive. Table 42 reports the four root effects used for both intervals. This study uses four roots and two orders, compared with eight roots and four orders in the main experiment.

Table 42: Root effects for the PIE repartition study. Each entry averages the five partition effects within one root; the reported intervals are $\bar{d} \pm t_{.975,3}\,\widehat{\sigma}/\sqrt{4}$.

Quantity	Root 1	Root 2	Root 3	Root 4	Mean	SD	95% CI
Δ final score	1.1	1.4	1.6	1.9	1.5	0.337	[1.0, 2.0]
Δ useful optimization	2.0	2.4	2.9	3.5	2.7	0.648	[1.7, 3.7]

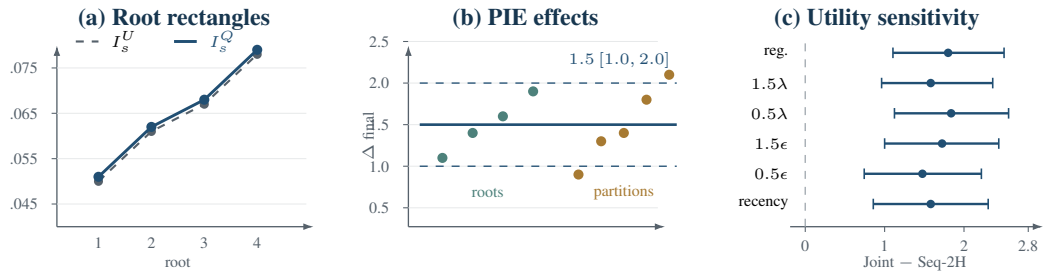


Figure 11: **Root-level effects and sensitivity analyses.** (a) Root-level raw and penalized rectangles (Tables 13 and 30); the penalty contribution is nearly constant across roots. (b) Four root effects and five partition effects for the PIE repartition study, with the root-clustered mean and interval (Tables 41 and 42). (c) Paired advantage under alternative declared utilities (Table 31); every interval excludes zero.

As Figure 11b makes explicit, the uncertainty calculation has four independent training-root clusters. The five PIE partitions are repeated measurements within each cluster; treating them as independent would artificially narrow the intervals. These results reduce dependence on one released problem split but remain within the same PIE collection and RLVR setting. The analysis costs $160 \times 62.5 = 10{,}000$ H100-hours across both methods, charged in Table 27.

I.10 When checkpoint selection helps

Checkpoint selection is useful when the archive contains parents with different adaptation outcomes and the controller can identify those differences before committing the full update. Recovery-aware valuation adds a second requirement: the preferred parent must depend on the recovery that completes its child. The opportunity matrices measure this interaction, the held-out controller study measures prediction and the paired deployment study measures the resulting incumbent quality.

The controls separate nearby explanations. Tuned replay measures what can be gained along the incumbent trajectory. Fixed-donor repair separates the value of repair from donor choice. The oracle gap measures the remaining cost of imperfect prediction. Candidate-count and archive-capacity studies show when historical alternatives are available, while the resource ledger determines whether their benefit survives execution cost. These measurements locate the method's advantage in the tested low-rank, verified-reward setting.

I.11 Run-record schema

For every reported run, the record contains:

1. immutable run ID, code commit, environment lock, command line, hardware and wall-clock interval;
2. model, tokenizer, dataset, generator, prompt, verifier and split revisions;
3. parent ID, archive snapshot, optimizer-state rule, action mask, sampling probability and random seed;
4. update tokens, FLOPs estimate, accelerator and CPU verifier time, storage, swap time and failures;
5. per-stage and per-task development and sealed scores, with controller-access flags;
6. raw branch outcomes, rejected children, evicted members and technical-rerun rationale;
7. controller checkpoint, feature normalization, missingness masks, training logs, calibration and selected thresholds;
8. exact baseline deviations from source methods and every development configuration attempted.

Each genealogy edge records one budgeted training operation. Checkpoint copies share a policy identity until their updates diverge, after which their parent IDs and branch outcomes define distinct histories.

I.12 BOUNDARY CASES AND SANITY CHECKS

One archive member. If $|\mathcal{A}_k| = 1$, ancestry selection disappears and EVOLPO reduces to recovery selection over the incumbent.

One recovery action. If $|\mathcal{R}_k| = 1$, the interaction in Equation (6) is undefined. The method becomes historical initialization selection and must be compared directly to short transfer trials and TeLAPA-style origin selection.

No post-adaptation information. If y_k contains no observation influenced by a, the two controller decisions can be collapsed into one joint contextual action without loss of expressiveness. An episodic interpretation is then unnecessary.

Free exhaustive evaluation. If every $Q(a, r)$ can be measured exactly at zero cost before deployment, exhaustive selection is optimal and learned prediction is unnecessary. The controller is motivated precisely by costly stochastic branches and transfer across histories.

Perfect replay. If incumbent replay attains the maximal new-task score while satisfying every retention constraint within the budget, no alternative history can improve the constrained objective. The correct conclusion is that the archive adds no value in that setting.

Reward hacking. Terminal credit inherits the validity of its verifier. Final panels use independently implemented checks where available and invalid outputs receive zero reward. The reported utility therefore measures verified task performance under these checks.