EPISTEMIC UNCERTAINTY FOR TEST-TIME DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Verifier-guided test-time search with large language models seeks one exceptional program within a small budget. TTT-DISCOVER trains the generating policy at test time with an entropic objective that emphasises the best sampled candidates, yet it can only credit programs the policy actually samples. We ask whether disagreement among policy adapters is a useful additional exploration signal. Uncertainty-Guided Test-Time Training (UG-TTT)[1] maintains a small ensemble of low-rank adapters over a frozen base model, measures per-token disagreement as the mutual information between the next token and the ensemble member (an epistemic-uncertainty proxy, not a calibrated posterior estimate), and adds a rollout-level aggregate of it to the advantage as an exploration bonus. At a matched budget of 384 rollouts on six TTT-DISCOVER tasks, UG-TTT attains the higher maximum reward in five of the six seed-42 comparisons and a 0.03% lower one on the Erdős problem. Circle-packing differences are positive at all three seeds (with differing comparator configurations), outcomes on an autocorrelation inequality and a heuristic-contest task are mixed across seeds, and the single-cell denoising verifier's error falls by 8.7% on Qwen3-8B and by 2.9% on a Qwen3-14B pair. In a one-run-per-arm component study on circle packing, the mutual-information bonus attains the highest maximum reward, and removing the adapter regulariser leaves disagreement nearly unchanged, indicating that diversified initialisation supplies the signal over six optimiser steps. Diversity depends on the window and the measure: at the end of training every seed-42 mathematical baseline has concentrated on one solution family while UG-TTT keeps at least three, and a rule-free structural measure favours UG-TTT in 9 of 12 run pairs over the late epochs and in 7 of 12 over the whole run. Ensemble disagreement is thus a practical exploration signal for budgeted test-time search; whether it beats simpler uncertainty bonuses, and whether diversity mediates its gains, remain open.

1 INTRODUCTION

Large language models (LLMs) increasingly take part in scientific and algorithmic discovery by proposing candidate programs that an automatic verifier scores (Qi et al., 2023; Ranković et al., 2025); AlphaEvolve (Novikov et al., 2025) and TTT-DISCOVER (Yuksekgonul et al., 2026) show that an LLM in a verification loop can rediscover, and occasionally improve on, human-designed solutions to open mathematical problems. In this setting the deliverable is the single best verified program found within a budget, so the quantity of interest is the maximum reward rather than the average.

Exploration under a maximum-oriented objective. TTT-DISCOVER trains the generating policy at test time with an entropic objective whose adaptive temperature weights each group of rollouts toward its best members, and it reuses high-scoring programs as parents through PUCT (Rosin, 2011). Both mechanisms act on candidates the search has already generated: entropic weighting can amplify an exceptional program once it is sampled, but it cannot credit a family of programs that the policy never proposes. With a few hundred rollouts, a policy that keeps proposing close variants of the same program can leave better alternatives unsampled (Yuksekgonul et al., 2026;

[1] Code, per-rollout logs and analysis scripts are publicly released; the repository link is withheld during review to preserve anonymity.

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.

Thomas et al., 2025). Because each step sharpens the policy toward its best members, we expect such concentration, and any diversity gap between methods, to be most visible late in the run rather than in a whole-run average. We ask whether an additional signal that directs sampling toward what the adapted policy has not settled improves the best verified result at a small budget.

Disagreement among adapted hypotheses. Discovery can require combinations of ideas that are poorly represented in a model's prior experience, which motivates exploring program families that the current policy rarely generates. A single network's predictive entropy is a natural exploration signal, but it mixes intrinsic ambiguity of the next token with uncertainty that further adaptation could resolve (Hüllermeier & Waegeman, 2021; Yadkori et al., 2024). Disagreement among several adapted hypotheses targets the second component: the mutual information between a prediction and the hypothesis that produced it is the BALD criterion (Houlsby et al., 2011), and ensembles of low-rank adapters (LoRA) compute it cheaply over a shared frozen base (Wang et al., 2023a; Balabanov & Linander, 2024). We hypothesise that this disagreement helps identify alternatives worth sampling, and we use it as an epistemic-uncertainty *proxy*: our adapters are trained jointly on shared data from a deliberately diversified initialisation, so they form an empirical ensemble rather than calibrated posterior samples. Our experiments measure search reward and program diversity; they do not measure pretraining-data coverage or establish scientific novelty.

Contributions. (i) UG-TTT, which extends TTT-DISCOVER with a K-adapter LoRA ensemble whose token-level disagreement is aggregated into one exploration bonus per rollout and added to the entropic advantage after group standardisation and advantage-scale matching (§2); without the ensemble and the regulariser it reduces to the TTT-DISCOVER objective (Prop. 3). (ii) An account of what keeps the ensemble informative: diversified up-projections make the members disagree from the first step, whereas the stacked down-projections already attain over 99.7% of the largest nuclear norm possible at their Frobenius norm at random initialisation (Prop. 2), so the nuclear-norm regulariser we adopt from Zhai et al. (2026) acts mainly on adapter scale over our horizon (§2.4). (iii) A matched-budget study on six TTT-DISCOVER tasks with additional seeds, a second base model, a component study, and two diversity measures over the whole run and its late epochs (§3). UG-TTT attains the higher maximum reward in five of six seed-42 comparisons, and one circle-packing run matches the value of the published Qwen3-8B run within its 384 rollouts; mixed and negative outcomes are reported alongside the gains.

2 UNCERTAINTY-GUIDED TEST-TIME TRAINING

2.1 SETUP

A task consists of a description d and a deterministic verifier R that scores a complete program. A parent state s is a previously verified program; the policy generates a response $o = (o_1, \ldots, o_{|o|})$ of thinking tokens and code, one token at a time, from a prompt q that embeds (d, s). The program extracted from o is the search action, and the verifier returns its terminal reward R_{exec} (0 for an invalid program). The metric of interest is the best reward reached within the budget, $R_{\max}$.

Each optimiser step draws 8 parent states from the PUCT buffer and samples $G{=}8$ rollouts from each; the rollouts of one parent form a *group*, within which advantages and the exploration bonus are computed. Following TTT-DISCOVER, rollout i receives the leave-one-out entropic advantage $A_i = w_i - 1$, with weights $w_i \propto \exp(\beta(s) R^{(i)}_{\text{exec}})$ at a per-group temperature $\beta(s)$ solved from R_{exec} alone for a fixed KL budget of $\ln 2$ (App. C). These advantages are not centred (their within-group sum is non-negative), they are invariant to an affine rescaling of the group's rewards, and they vanish when all rewards tie; when four or more of the eight rollouts tie at the maximum the KL budget is not attainable at finite β, and the solver saturates (App. C).

2.2 ENSEMBLE DISAGREEMENT AS AN EXPLORATION SIGNAL

We replace the single TTT-DISCOVER adapter with K low-rank perturbations of a frozen base, $\Theta^{(k)} = \Theta_{\text{base}} + B^{(k)} A^{(k)}$ with $A^{(k)} \in \mathbb{R}^{r \times d_{\text{in}}}$, $B^{(k)} \in \mathbb{R}^{d_{\text{out}} \times r}$ and $k \in \{1, \ldots, K\}$. At position t of a scored sequence, with member distributions $p_{k,t} = p_{\theta_k}(\cdot \mid q, o_{<t})$ and their mixture $\bar{p}_t =$

$\frac{1}{K}\sum_k p_{k,t}$,

$$\mathrm{MI}_t \;=\; H(\bar{p}_t) \;-\; \frac{1}{K}\sum_{k=1}^{K} H(p_{k,t}) \;=\; \frac{1}{K}\sum_{k=1}^{K} \mathrm{KL}\big(p_{k,t} \,\big\|\, \bar{p}_t\big)\,, \tag{1}$$

where H is Shannon entropy over the vocabulary. MI_t is the mutual information between the next token and a uniformly drawn member (the BALD quantity (Houlsby et al., 2011; Cao & Tsang, 2021) under the empirical ensemble, equivalently a generalised Jensen–Shannon divergence); it vanishes exactly when all members agree and grows when they place mass on incompatible tokens. Subtracting the mean member entropy separates between-member disagreement from within-member predictive entropy under this ensemble. We read MI_t as a proxy for epistemic uncertainty, not as calibrated posterior uncertainty: the members share data and advantages and are initialised apart on purpose (§2.4), so their disagreement can also reflect initialisation and optimisation, and LoRA dropout (p=0.05) is active in every forward pass, so each $p_{k,t}$ is also one Monte-Carlo-dropout draw (Gal & Ghahramani, 2016). Both terms of Eq. (1) are computed from the full (K, chunk, V) logit tensor in a chunked LM head (App. F). A shared base keeps one copy of the base weights in memory and lets the K adapter passes run as one batch; members built on different bases would also disagree for pretraining reasons unrelated to the task. We use K=5, the deep-ensemble default (Lakshminarayanan et al., 2017; Ovadia et al., 2019) and the largest that fits beside the 8B base on one 80–96 GB device (App. K reports $K \in \{2, 3\}$).

Most positions of a code rollout are boilerplate on which the adapters agree, so each rollout is reduced to one score by averaging its most uncertain positions,

$$U_i \;=\; \operatorname*{top\text{-}7\%\text{-}mean}_{t}\, \mathrm{MI}_t^{(i)}, \tag{2}$$

where t ranges over all scored positions of the sequence (q, o_i), prompt included (the prompt is shared by the rollouts of a group). The 7% fraction is a heuristic inside the 5–12% high-uncertainty decoding rates reported by He et al. (2025); it scales the number of included positions with length but does not by itself make U_i length-invariant.

Rollouts are generated by an epoch-rotated round-robin over adapters ($80/80/80/72/72$ per adapter over six epochs). Every rollout is scored through all K adapters, and every adapter is trained on all G rollouts of a group through a token-level importance-weighted surrogate against the logged sampling log-probabilities of the adapter that generated the rollout (unclipped, as in the TTT-DISCOVER reference configuration; App. C). For the $K-1$ non-generating adapters this reweights each token at its sampled prefix; it is a surrogate for, not an exact correction to, their on-policy objectives.

2.3 DISAGREEMENT-SHAPED ADVANTAGE

The raw scale of U_i varies by orders of magnitude across tasks and epochs, so we standardise it within the group and clip it,

$$\tilde{U}_i \;=\; \mathrm{clip}\!\left(\frac{U_i - \overline{U}}{\sigma_U},\; -3,\; 3\right), \qquad \overline{U} = \tfrac{1}{G}\sum_i U_i,\;\; \sigma_U = \sqrt{\tfrac{1}{G-1}\textstyle\sum_i (U_i - \overline{U})^2}, \tag{3}$$

with $\tilde{U}_i \equiv 0$ when $\sigma_U = 0$ (the clip never binds for $G \le 10$; Prop. 1, App. E). The exploration coefficient follows the group's entropic temperature,

$$\gamma_{\mathrm{eff}}(s) \;=\; \alpha \cdot \min\!\left(\frac{\beta(s)}{\beta_{\mathrm{ref}}},\; \gamma_{\max}\right), \tag{4}$$

with base rate α, reference temperature β_{ref} and cap $\gamma_{\max}$, and the shaped advantage that replaces A_i in every adapter's loss is

$$A_i^{\mathrm{shaped}} \;=\; A_i \;+\; \gamma_{\mathrm{eff}}(s) \cdot \overline{|A|} \cdot \tilde{U}_i\,, \qquad \overline{|A|} \;=\; \frac{1}{G}\sum_j |A_j|. \tag{5}$$

Disagreement is measured per token but credited per rollout: A_i^{shaped} is applied to every token of rollout i. The prefactor $\overline{|A|}$ matches the bonus to the group's realised entropic advantages. Because $\beta(s)$ is solved for a fixed KL budget, those advantages do not shrink as the group's rewards draw together; the bonus vanishes only when all execution rewards tie, where $A \equiv 0$. The coefficient

γ_{eff} grows with $\beta(s)$, which scales inversely with the reward spread in the task's native units, up to $\alpha\gamma_{\max}$, so the bonus gains weight relative to A as a group's rewards draw together; reward units and spacing move $\beta(s)$ as well, so $\beta(s)$ is not by itself a detector of policy collapse. The bonus is deliberately biased toward high-disagreement rollouts; that bias is the exploration mechanism.

Remark 1 (β*-coupling in practice*). Nothing in Eq. (4) is scheduled or tuned per task. In the twelve reported UG-TTT runs, 68% of trained groups lie below the cap, and in the 9% whose execution rewards all tie the bonus vanishes with $|\overline{A}|$ (App. N). On identical rollouts and the deployed score, a strength-matched constant coefficient changes the shaped advantages little (median within-group cosine 0.993 over 526 groups), so we treat the coupling as an implementation default rather than a load-bearing component.

2.4 Ensemble diversity: initialisation and a nuclear-norm heuristic

With the standard zero-initialised up-projections the K members are functionally identical and ensemble MI stays near 10^{-5} (§3.3). We therefore initialise every $B^{(k)}$ with small Gaussian entries (σ=0.01, one seed per member), which makes the members distinct from the first forward pass; the K=1 baseline's adapter receives the same draw as the ensemble's first member. Following Zhai et al. (2026), who maximise a Frobenius-normalised nuclear norm of stacked LoRA down-projections to diversify reward-model ensembles, we also add a regulariser on the per-layer stacked down-projections,

$$W_\ell \;=\; \big[A_\ell^{(1)};\,\ldots;\,A_\ell^{(K)}\big] \;\in\; \mathbb{R}^{Kr\times d_{\text{in}}}, \qquad \mathcal{L}_{\text{NNM}} \;=\; -\,\frac{1}{L}\sum_{\ell=1}^{L}\big\|W_\ell\big\|_*, \tag{6}$$

with $\|\cdot\|_*$ the nuclear norm and L the number of adapted linear layers; unlike Zhai et al., we use the raw norm. Its geometry is characterised by the following fixed-norm statement.

Proposition 2 (*Equal-norm maximiser*). *Fix $\|A_\ell^{(k)}\|_F = c$ for all k. If $Kr \le d_{\text{in}}$, every global maximiser of $\|W_\ell\|_*$ on this set consists of blocks whose row spaces are mutually orthogonal in $\mathbb{R}^{d_{\text{in}}}$, each with r singular values equal to $c/\sqrt{r}$.*

The proof (App. E) rests on $\|W\|_* \le \sqrt{\operatorname{rank}(W)}\,\|W\|_F$. Proposition 2 concerns parameter geometry only. It does not guarantee distinct predictions (with $B^{(k)} = 0$ all members equal the base model at the maximiser), and the raw regulariser does not impose its constraint: the function-preserving rescaling $A^{(k)} \to tA^{(k)}$, $B^{(k)} \to B^{(k)}/t$ multiplies $\|W_\ell\|_*$ by t without changing any member, and nothing in the objective penalises it (the optimiser applies no weight decay). Moreover, random rows in $\mathbb{R}^{d_{\text{in}}}$ with $d_{\text{in}} \ge 4096$ are nearly orthogonal: at initialisation our stacked matrices already attain over 99.7% of the bound $\sqrt{Kr}\,\|W_\ell\|_F$, while the logged nuclear norm grows by 0.4–1.7% over training (App. F), so most of that growth is scale. We therefore treat NNM as a heuristic and measure its contribution in §3.3.

2.5 Training objective

Each optimiser step minimises

$$\mathcal{L}_{\text{total}} \;=\; \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{\text{PG}}^{(k)}(\theta_k) \;+\; \lambda_{\text{NNM}}\,\mathcal{L}_{\text{NNM}}, \tag{7}$$

where $\mathcal{L}_{\text{PG}}^{(k)}$ is adapter k's importance-weighted loss with the shaped advantage, and the KL anchor to the frozen base enters, as in the TTT-Discover reference implementation, as a per-token correction of the advantage computed from the generating adapter's sampling log-probabilities and shared by all K losses (App. C). Algorithm 1 (App. D) gives one optimiser step; the adapters are processed sequentially to bound activation memory (Ye et al., 2024). When λ_{NNM}=0, Eq. (7) reduces algebraically to the TTT-Discover objective if K=1 or if all members define the same policy (Prop. 3); this is an objective-level statement, not a claim of identical optimiser trajectories.

3 EXPERIMENTS

We evaluate UG-TTT on six TTT-DISCOVER tasks (Yuksekgonul et al., 2026) with their prompts and verifiers unchanged (App. B): four mathematical tasks, a step-function autocorrelation inequality (**AC1**, minimise C_1), the autocorrelation lower-bound problem (**AC2**, maximise C_2), packing $n{=}26$ circles in the unit square (**CP26**, maximise the radius sum) and the Erdős minimum-overlap problem (**Erdős**, minimise C_5); a heuristic-contest task (**AHC039**, an AtCoder Heuristic Contest geometry problem in C++, scored on the vendored public inputs); and single-cell expression denoising (**Denoising**, Python; reward $1/\mathrm{MSE}$ on the held-out molecules of a fixed molecular cross-validation split of one dataset). Every reported number is a search-verifier score: we do not evaluate the selected programs on untouched datasets or hidden test sets.

Setup. All runs share the configuration of App. G: Qwen3-8B (Yang et al., 2025) (Qwen3-14B for one denoising pair), rank-16 LoRA, $G{=}8$ rollouts per parent, 8 parents per optimiser step, and six optimiser steps (384 rollouts). UG-TTT adds $K{=}5$, $\lambda_{\mathrm{NNM}}{=}0.075$ and $\alpha{=}0.1$. The baseline is TTT-DISCOVER's objective in our implementation ($K{=}1$, $\alpha{=}0$, $\lambda_{\mathrm{NNM}}{=}0$), with two exceptions in Tab. 1: the seed-42 CP26 comparator is a zero-initialised five-adapter ensemble without NNM that runs the full objective (a degenerate-ensemble control, §3.3), and streaming early-stop, a thinking-length cap gated by windowed MI (App. H), ran on the seed-42 AC1 UG-TTT arm and on both seed-42 CP26 arms; every other run has it off. Paired arms have equal rollout budgets and token limits; their realised token counts differ. The published TTT-DISCOVER runs use 50 steps of 512 rollouts, and we do not claim that our deltas transfer to that scale. A rollout is *correct* when the verifier accepts its program ($R{>}0$).

Diversity measures. On the four mathematical tasks each correct rollout receives one algorithmic-family label from an ordered list of rules (vocabularies written after inspecting rollouts from these runs), and *family entropy* H is the entropy of the family distribution. On every task, a rule-free *structural entropy* embeds each correct program (AST fingerprint for Python, TF–IDF n-grams for C++), clusters both arms jointly by k-means at $k \in \{6, 8, 10\}$, and compares per-arm cluster entropy over the whole run and over epochs 4–5; the whole-run measure was fixed before the Qwen3-14B pair was run and is our primary rule-free endpoint, and the epoch split, motivated by the hypothesis of §1, was computed afterwards (App. M). Rollouts share parents and an adaptive buffer, so they are not exchangeable, and we report rollout-permutation statistics only descriptively (App. M).

3.1 MAIN RESULTS

At seed 42, UG-TTT attains the higher $R_{\max}$ on five of the six tasks and trails on Erdős by 0.0007 (0.03%; Tab. 1); the AHC039 gain ($+0.0008$) reverses at a second seed (§3.2). On CP26 the UG-TTT run matches the value of the published TTT-DISCOVER Qwen3-8B run (2.635983) to six decimals within its 384-rollout budget; the published run used 25,600 rollouts, and we compare neither first-hitting times nor total compute. On AC1 part of the gap precedes learning: the best epoch-0 rollout, drawn before the first update, is 0.6338 for UG-TTT against 0.5271 for the baseline, and we ran no frozen-ensemble control that would separate the ensemble's initial proposals from its updates (App. K). On denoising, $R = 1/\mathrm{MSE}$, so the $+0.4469$ ($+9.5\%$) gain is an 8.7% lower reconstruction error ($0.2130 \rightarrow 0.1945$) on the search verifier. The best AC2 program calls a library differential-evolution solver, a family no correct baseline rollout uses, and the best Erdős UG-TTT program starts from a Fourier initialisation the baseline never adopts (Apps. I, J); these are well-known techniques, and their adoption does not by itself show that the bonus caused it. Figure 1 plots the best reward so far against generated tokens: equal rollout budgets consumed different numbers of tokens, fewer for UG-TTT on five tasks (on AC1 and CP26 partly through streaming early-stop) and more on AC2, while each UG-TTT rollout also costs a K-way scoring pass.

3.2 SEEDS AND BASE MODEL

Table 2 adds seeds 2 and 3 on CP26 and AC1, seed 2 on AHC039, and a Qwen3-14B denoising pair at seed 1, all with streaming off on both arms. CP26 differences are positive at all three seeds ($+0.0001$, $+0.0075$, $+0.0058$; the seed-42 comparator is the control of Tab. 1), and UG-TTT finishes within 0.0007 of the published value at two of them while no comparator run comes within

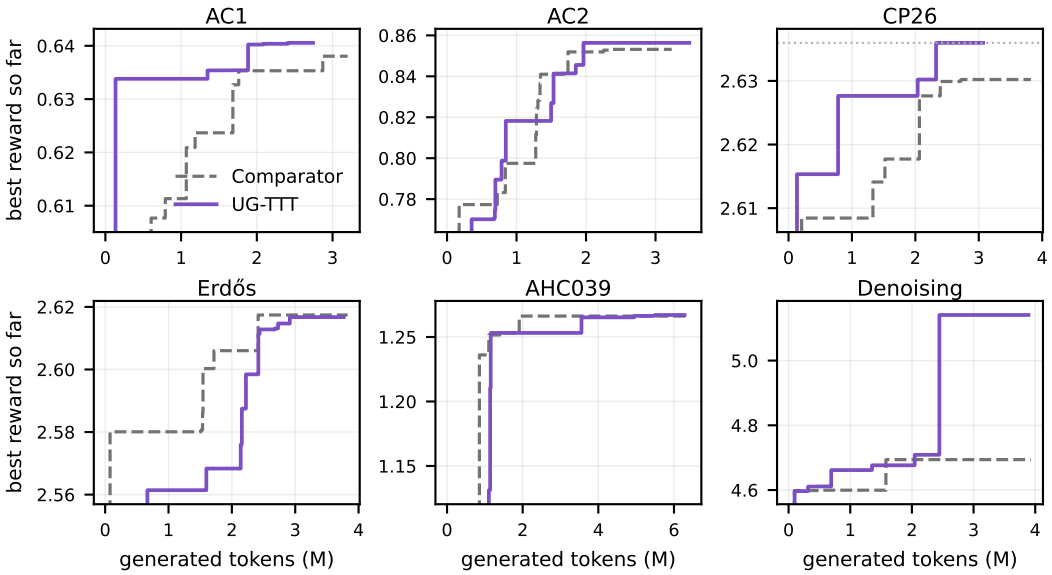


Figure 1: **Best verifier reward so far against cumulative generated tokens** for the six seed-42 pairs (384 rollouts per arm; for an interrupted epoch, its last logged pass). Purple: UG-TTT; gray dashed: comparator (the zero-initialised $K{=}5$ control on CP26, the $K{=}1$ baseline elsewhere); dotted line: the published Qwen3-8B CP26 value. Totals (comparator/UG-TTT, million tokens): AC1 3.20/2.74, AC2 3.23/3.49, CP26 3.82/3.07, Erdős 3.83/3.77, AHC039 6.40/6.28, denoising 3.93/3.89.

Table 1: **Seed-42 comparisons on six TTT-DISCOVER tasks** (one run per arm; six optimiser steps, 384 rollouts; deltas from unrounded values). $R_{\max}$: best verifier reward reached during training, higher is better (R per task in App. B). Structural H: rule-free cluster entropy (bits) of each arm's correct programs, shown per arm at $k{=}8$ over epochs 4–5, the window in which the hypothesis of §1 predicts a gap; both ΔH columns (UG-TTT minus comparator, epochs 4–5 and whole run) are medians over $k \in \{6, 8, 10\}$, so the cells need not subtract exactly (all k in App. M); the whole-run measure is the conservative primary endpoint. Bold marks the higher arm. Final-epoch family entropies of the mathematical tasks are in Tab. 10. [a]Comparator: a zero-initialised $K{=}5$ ensemble without NNM running the full objective, with streaming early-stop on both arms; every other comparator is the $K{=}1$ baseline. [b]Streaming early-stop on the UG-TTT arm only. The AC1 baseline log and both AHC039 logs each hold one re-sampled epoch after an interruption (448, 440 and 448 rollouts; App. G); $R_{\max}$ is unchanged without them. The CP26 UG-TTT value equals the published TTT-DISCOVER Qwen3-8B value at six decimals.

	$R_{\max}$ during training			Structural H, epochs 4–5			Whole run
Task	Comparator	UG-TTT	Δ	Comparator	UG-TTT	ΔH	ΔH
AC1[b]	0.6381	**0.6406**	+0.0025	**1.79**	1.51	−0.27	−0.46
AC2	0.8532	**0.8563**	+0.0031	1.47	**2.00**	+0.53	+0.50
CP26[a]	2.6302	**2.6360**	+0.0058	1.41	**2.06**	+0.76	+0.47
Erdős	**2.6174**	2.6167	−0.0007	**1.92**	1.55	−0.37	−0.11
AHC039	1.2662	**1.2671**	+0.0008	0.73	**1.38**	+0.65	+0.36
Denoising	4.6940	**5.1409**	+0.4469	1.79	**2.24**	+0.45	+0.46

0.0057. AC1 is mixed (-0.0027, $+5 \times 10^{-5}$, $+0.0025$); its seed-2/3 reruns give both arms a prompt that carries a previous best program, so those comparisons hold within a row only. AHC039 is mixed as well (-0.0227 at seed 2, $+0.0008$ at seed 42; mean difference -0.0109, i.e. -0.85%); two pairs establish neither an improvement nor equivalence. The Qwen3-14B denoising pair changes base model and seed together, so it isolates neither: UG-TTT again attains the higher $R_{\max}$ ($+0.1438$, $+2.95\%$, a 2.9% lower MSE). With at most three pairs per task we make no significance claims. A

Table 2: **Additional seeds and base model** (configuration of Tab. 1 unless noted; one run per cell; deltas from unrounded values; the seed-42 rows repeat Tab. 1). [a]Degenerate-ensemble comparator of Tab. 1. [c]Rerun prompts for both arms carry a previous best program, so the comparison is within the row. The AC1 seed-3 gap is 5×10^{-5} (printed $+0.0000$). All rows other than seed-42 CP26/AC1 run streaming off on both arms.

Task	Seed	Baseline	UG-TTT	$\Delta R_{\max}$
CP26	2	2.6299	2.6300	$+0.0001$
CP26	3	2.6278	2.6353	$+0.0075$
CP26[a]	42	2.6302	2.6360	$+0.0058$
AC1[c]	2	0.6628	0.6601	-0.0027
AC1[c]	3	0.6601	0.6602	$+0.0000$
AC1	42	0.6381	0.6406	$+0.0025$
AHC039	2	1.2933	1.2706	-0.0227
AHC039	42	1.2662	1.2671	$+0.0008$
Denoising	42	4.6940	5.1409	$+0.4469$
Denoising	1 (Qwen3-14B)	4.8823	5.0261	$+0.1438$

Table 3: **CP26 component pilot** (seed 42, context 8,192, K=5; one run per arm in a matched configuration; observed ordering, no significance claimed). Token-entropy arm: all-position mean of $-\bar{p}_t(o_t) \log \bar{p}_t(o_t)$ at the sampled token; member-variance arm: all-position mean of $\mathrm{Var}_k \log p_{k,t}(o_t)$; the full method uses Eq. (1) with the top-7% mean (App. L). Mean MI: per-rollout ensemble MI averaged over epoch 5 (the α=0 arm does not compute it).

Configuration	$R_{\max}$	Δ vs. full	Mean MI (ep. 5)
Full UG-TTT (MI bonus)	**2.6353**	—	1.83×10^{-2}
Token-entropy bonus	2.6343	-0.0010	1.73×10^{-2}
Member-variance bonus	2.6311	-0.0042	1.87×10^{-2}
α=0 (no bonus; NNM on)	2.6307	-0.0046	—
No NNM (MI bonus on)	2.6275	-0.0078	1.79×10^{-2}
Streaming on (full method)	2.6270	-0.0083	1.83×10^{-2}

$K \in \{2, 3, 5\}$ sweep at one CP26 seed (App. K) shows that smaller ensembles run, not that K=5 is optimal.

3.3 Which components matter: a reduced-context pilot on CP26

Table 3 compares the full method on CP26 with five single-change arms at seed 42 and an 8,192-token context (shorter than the 32,768 of Tab. 1, for budget); the full method is re-run in this configuration and one run is made per arm. The MI arm has the highest $R_{\max}$, exceeding the token-entropy arm by 0.0010 and the member-variance arm by 0.0042. Both alternative scores are computed from the log-probabilities of the sampled token and averaged over all positions, whereas the MI bonus uses the full-vocabulary quantity of Eq. (1) and the top-7% mean, so each comparison changes the signal and its aggregation together. With one run per arm, these orderings establish neither reproducibility nor an explanation such as the separation of epistemic from aleatoric uncertainty. On diversity the grid gives the MI bonus no advantage: the member-variance arm ends the most diverse (App. L).

Initialisation, NNM and disagreement. In the Tab. 1 configuration, the zero-initialised CP26 comparator's ensemble MI never exceeds $\approx 10^{-5}$, yet group standardisation (Eq. (3)) turns that numerically tiny score into an $O(1)$ bonus, so the comparator is a degenerate-ensemble control, not an α=0 run (App. L); the full method surpasses it while generating 20% fewer tokens (3.07 versus 3.82 million). In the matched-initialisation grid, removing NNM leaves epoch-5 MI essentially unchanged (1.79 versus 1.83×10^{-2}) and lowers $R_{\max}$ by 0.0078 in this single run. Together with the geometry of §2.4, this indicates that diversified initialisation, not NNM, supplies the disagreement signal over six optimiser steps; NNM's measured effect is on reward, through a mechanism we have not identified.

3.4 DIVERSITY IS MEASURE-DEPENDENT

We report diversity as a secondary observation, because the two measures often disagree. Under the family taxonomy, every seed-42 mathematical baseline ends with one family holding 82 to 93% of its correct rollouts and $H \leq 0.70$ bits, whereas UG-TTT ends with at least three live families and H between 1.12 and 1.76 bits (Tab. 10, App. K); across the six CP26 and AC1 seed pairs the final-epoch gap favours UG-TTT in four. Under the rule-free structural measure, UG-TTT is the more diverse arm over epochs 4–5 in 9 of the 12 run pairs (10 of 12 at epoch 5 alone; the exceptions are AC1 seed 42, Erdős and CP26 seed 3, the last at -0.06 bits) and over the whole run in 7 of 12, including the three Qwen3-8B AHC039 and denoising pairs; over the whole run it is the less diverse one on AC1 (seeds 2 and 42), Erdős, CP26 seed 3 and the Qwen3-14B denoising pair, whose reversal is confined to epochs 0–2. The whole-run count is unchanged when both arms are compared at a common number of programs, and the window split was made after the whole-run analysis (App. M). The two measures agree in sign on only three of the eight mathematical pairs. More structural variation can reflect harmless rearrangements of code, and more family labels can reflect classifier choices; neither measures scientific novelty. We therefore do not claim that diversity mediates the reward gains, and we do not attribute diversity to the MI bonus: per-rollout MI does not predict which rollouts win (App. O).

LIMITATIONS

Evidence per task is thin: at most three run pairs per task, one run per arm in the component pilot (at reduced context), and a 384-rollout budget far below the published runs. We did not run a frozen-ensemble search control, so the contribution of adaptation is not separated from that of sampling from five initial adapters, and the alternative uncertainty bonuses differ from the MI bonus in aggregation as well as in signal. All results are search-verifier scores; the selected denoising programs were not evaluated on untouched datasets, nor the AHC039 programs on hidden tests. Disagreement is an uncalibrated proxy, and because the ensemble both defines and is rewarded for it, the bonus can favour persistent disagreement, including on invalid programs (correct rollouts tend to carry lower MI; App. O). The cross-adapter update is an unclipped token-level surrogate; $(\alpha, \beta_{\text{ref}}, \gamma_{\max})$ and the top-7% fraction are heuristics; the regulariser acts mainly on scale over our horizon; the ensemble costs a K-fold adapter and scoring overhead; and streaming early-stop has a per-task reward price (App. H). Verifiers are deterministic program checkers; noisy or learned verifiers would require recalibrating the bonus.

4 BACKGROUND AND RELATED WORK

Test-time optimisation and its collapse modes. Reinforcement learning from human feedback aligns LLMs through PPO-style optimisation against a learned reward model (Havrilla et al., 2024; Hong et al., 2025; Rafailov et al., 2024; Guo et al., 2025); a scalar reward can compress the objective, suppress exploration and induce overconfidence (Khalaf et al., 2025; Xia et al., 2025; Shorinwa et al., 2025; Farquhar et al., 2024; Tao et al., 2025). Test-time training adapts parameters at inference from self-supervised signals or retrieved neighbours (Hu et al., 2025; Hardt & Sun, 2023; Akyürek et al., 2024), often through parameter-efficient adapters (Moradi et al., 2025), and can concentrate on a few solutions under sparse outcome rewards (Thomas et al., 2025); LLM-driven program search with evolutionary or learned proposal distributions includes FunSearch, AlphaEvolve, ShinkaEvolve and TTT-DISCOVER (Romera-Paredes et al., 2024; Novikov et al., 2025; Lange et al., 2025; Yuksekgonul et al., 2026).

Uncertainty in the training signal. Uncertainty quantification (Abdar et al., 2021; Charpentier et al., 2022; Hüllermeier & Waegeman, 2021; Caldeira & Nord, 2021) is used in LLMs both to assess outputs, through token probabilities, verbalised confidence, ensemble disagreement and conformal intervals (Xia et al., 2025; Shorinwa et al., 2025; Farquhar et al., 2024; Tao et al., 2025; Liu et al., 2025; Vashurin et al., 2025; Huang et al., 2025; Fadeeva et al., 2023), and to shape learning: ensemble disagreement drives exploration in reinforcement learning (Osband et al., 2016; Pathak et al., 2019), epistemic reward uncertainty guides exploration in RLHF (Dwaracherla et al., 2024), and reward-model uncertainty is penalised to curb over-optimisation (Zhai et al., 2026). Curiosity bonuses and entropy regularisation counter exploitation in LLM training (Du et al., 2023; Liao et al., 2025; Gao et al., 2025; Levy et al., 2025; Cheng et al., 2026), though novelty rewards weaken in large

action spaces (Dai et al., 2025; Yuan et al., 2022), and AutoDiscovery (Agarwal et al., 2025) scores Bayesian surprise with a single frozen LLM. Our focus is narrower: token-distribution disagreement within a jointly adapted policy ensemble, used as a positive bonus for finite-budget test-time search. Its regulariser comes from Zhai et al. (2026); other LoRA orthogonality penalties act within one adapter (Zhang et al., 2023) or against previously frozen adapters (Wang et al., 2023b).

5 Conclusion

We asked whether disagreement among policy adapters helps verifier-guided test-time search beyond an entropic objective that already favours the best sampled candidates. UG-TTT turns the token-level mutual information of a LoRA ensemble, read as an epistemic-uncertainty proxy, into a rollout-level exploration bonus. At a matched 384-rollout budget it attains the higher maximum reward in five of six seed-42 comparisons, with positive circle-packing differences at three seeds, a lower denoising error in two model–seed configurations, mixed AC1 and AHC039 outcomes and a marginal Erdős loss; one circle-packing run matches the published Qwen3-8B value. The disagreement signal comes from diversified adapter initialisation over this horizon, the nuclear-norm regulariser acts mainly on scale, and the diversity gap opens late in training (9 of 12 pairs over epochs 4–5 under a rule-free measure, 7 of 12 over the whole run), as expected if the baseline policy concentrates as it trains, with a size that depends on the measure. Establishing an advantage over simpler uncertainty bonuses, separating adaptation from ensemble sampling, and evaluating selected programs beyond the search verifier are the next steps.

AI use statement

In this work, we used generative AI tools (Claude models by Anthropic, including Claude Code) for implementing methods (assistance in writing the training-loop, analysis, and figure-generation code), for designing and providing feedback on experiments, and for interpreting results (assistance with the statistical analyses of the logged rollouts). We have not used generative AI tools to generate synthetic data or to clean or reformat datasets. Additionally, we used generative AI tools to draft and edit parts of the paper for readability and to resolve LaTeX formatting issues. We have reviewed all AI-assisted work: AI-generated code was verified and tested by the authors, and the reported results were recomputed from the logged rollouts with the authors' own analysis scripts. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics statement

UG-TTT rewards disagreement among policy adapters while optimising a verifier that may be incomplete. Under any task with a cheap programmatic verifier, and a loosely specified verifier in particular (for instance, automated exploit search), a stronger search can find and exploit the verifier's loopholes. Two mitigations follow from the method: audit the verifier before scaling compute, and log ensemble disagreement, since a degenerate ensemble (MI near 10^{-5}, §3.3) cannot deliver the bonus and App. O bounds what the signal predicts. Per rollout, UG-TTT costs K adapter backward passes and a K-way scoring pass, so we make no claim of lower total compute.

Reproducibility statement

All six tasks, their prompts and their verifiers are taken unchanged from TTT-Discover (Yuksekgonul et al., 2026). Algorithm 1 specifies one optimiser step, and App. G lists the hyperparameters, optimiser settings, loss reduction and the configuration of every reported run, including the exceptions noted in Tab. 1. Every family-entropy number in the paper is computed with the released rule-based classifiers, and every structural-entropy number with the released clustering scripts; interrupted runs and continuation logs are handled as stated in Apps. G and N. The training code, the verifiers, the classifiers, per-rollout logs for all reported runs, and the analysis, verification and figure-generation scripts are publicly released; every number in the paper regenerates from them.

The repository link is withheld during review to preserve anonymity and will be given in the camera-ready.

REFERENCES

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

Dhruv Agarwal, Bodhisattwa Prasad Majumder, Reece Adamson, Megha Chakravorty, Satvika Reddy Gavireddy, Aditya Parashar, Harshit Surana, Bhavana Dalvi Mishra, Andrew McCallum, Ashish Sabharwal, et al. Autodiscovery: Open-ended scientific discovery via bayesian surprise. *arXiv preprint arXiv:2507.00310*, 2025.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*, 2024.

Oleksandr Balabanov and Hampus Linander. Uncertainty quantification in fine-tuned llms using lora ensembles. *arXiv preprint arXiv:2402.12264*, 2024.

João Caldeira and Brian Nord. Deeply uncertain: comparing methods of uncertainty quantification in deep learning algorithms. *Machine Learning: Science and Technology*, 2(1):015002, 2021.

Xiaofeng Cao and Ivor W Tsang. Bayesian active learning by disagreements: A geometric perspective. *arXiv preprint arXiv:2105.02543*, 2021.

Bertrand Charpentier, Ransalu Senanayake, Mykel Kochenderfer, and Stephan Günnemann. Disentangling epistemic and aleatoric uncertainty in reinforcement learning. *arXiv preprint arXiv:2206.01558*, 2022.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30377–30385, 2026.

Runpeng Dai, Linfeng Song, Haolin Liu, Zhenwen Liang, Dian Yu, Haitao Mi, Zhaopeng Tu, Rui Liu, Tong Zheng, Hongtu Zhu, et al. Cde: Curiosity-driven exploration for efficient reinforcement learning in large language models. *arXiv preprint arXiv:2509.09675*, 2025.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pp. 8657–8677. PMLR, 2023.

Vikranth Dwaracherla, Seyed Mohammad Asghari, Botao Hao, and Benjamin Van Roy. Efficient exploration for LLMs. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12215–12227, 2024.

Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, et al. Lm-polygraph: Uncertainty estimation for language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 446–461, 2023.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, 2016.

Jingtong Gao, Ling Pan, Yejing Wang, Rui Zhong, Chi Lu, Maolin Wang, Qingpeng Cai, Peng Jiang, and Xiangyu Zhao. Navigate the unknown: Enhancing llm reasoning with intrinsic motivation guided exploration. *arXiv preprint arXiv:2505.17621*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. *arXiv preprint arXiv:2305.18466*, 2023.

Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024.

Kaifeng He, Mingwei Liu, Chong Wang, Zike Li, Yanlin Wang, Xin Peng, and Zibin Zheng. Adadec: Uncertainty-guided adaptive decoding for llm-based code generation. *arXiv preprint arXiv:2506.08980*, 2025.

Haitao Hong, Yuchen Yan, Xingyu Wu, Guiyang Hou, Wenqi Zhang, Weiming Lu, Yongliang Shen, and Jun Xiao. Cooper: Co-optimizing policy and reward models in reinforcement learning for large language models. *arXiv preprint arXiv:2508.05613*, 2025.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Jinwu Hu, Zhitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. Test-time learning for large language models. *arXiv preprint arXiv:2505.20633*, 2025.

Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, Felix Juefei-Xu, and Lei Ma. Look before you leap: An exploratory study of uncertainty analysis for large language models. *IEEE Transactions on Software Engineering*, 51(2):413–429, 2025.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506, 2021.

Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio du Pin Calmon. Inference-time reward hacking in large language models. *arXiv preprint arXiv:2506.19248*, 2025.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

Guillaume Levy, Cédric Colas, Pierre-Yves Oudeyer, Thomas Carta, and Clément Romac. Worldllm: Improving llms' world modeling using curiosity-driven theory-making. *arXiv preprint arXiv:2506.06725*, 2025.

Mengqi Liao, Xiangyu Xi, Chen Ruinian, Jia Leng, Yangen Hu, Ke Zeng, Shuai Liu, and Huaiyu Wan. Enhancing efficiency and exploration in reinforcement learning for llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1451–1463, 2025.

Xiaoou Liu, Tiejin Chen, Longchao Da, Chacha Chen, Zhen Lin, and Hua Wei. Uncertainty quantification and confidence calibration in large language models: A survey. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 6107–6117, 2025.

Mohammad Mahdi Moradi, Hossam Amer, Sudhir Mudur, Weiwei Zhang, Yang Liu, and Walid Ahmed. Continuous self-improvement of large language models by test-time training with verifier-driven sample selection. *arXiv preprint arXiv:2505.19475*, 2025.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5062–5071, 2019.

Biqing Qi, Kaiyan Zhang, Haoxiang Li, Kai Tian, Sihang Zeng, Zhang-Ren Chen, and Bowen Zhou. Large language models are zero shot hypothesis proposers. *arXiv preprint arXiv:2311.05965*, 2023.

Rafael Rafailov, Yaswanth Chittepu, Ryan Park, Harshit Sushil Sikchi, Joey Hejna, Brad Knox, Chelsea Finn, and Scott Niekum. Scaling laws for reward model overoptimization in direct alignment algorithms. *Advances in Neural Information Processing Systems*, 37:126207–126242, 2024.

Bojana Ranković, Ryan-Rhys Griffiths, and Philippe Schwaller. Large language models as uncertainty-calibrated optimizers for experimental discovery. *arXiv preprint arXiv:2504.06265*, 2025.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Christopher D Rosin. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230, 2011.

Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z Ren, and Anirudha Majumdar. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. *ACM Computing Surveys*, 58(3):1–38, 2025.

Linwei Tao, Yi-Fan Yeh, Minjing Dong, Tao Huang, Philip Torr, and Chang Xu. Revisiting uncertainty estimation and calibration of large language models. *arXiv preprint arXiv:2505.23854*, 2025.

Morgan Thomas, Albert Bou, and Gianni De Fabritiis. Test-time training scaling laws for chemical exploration in drug design. *Journal of Chemical Information and Modeling*, 65(24):13178–13186, 2025.

Christos Thrampoulidis, Sadegh Mahdavi, and Wenlong Deng. Advantage shaping as surrogate reward maximization: Unifying pass@ k policy gradients. *arXiv preprint arXiv:2510.23049*, 2025.

Roman Vashurin, Ekaterina Fadeeva, Artem Vazhentsev, Lyudmila Rvanova, Daniil Vasilev, Akim Tsvigun, Sergey Petrakov, Rui Xing, Abdelrahman Sadallah, Kirill Grishchenkov, et al. Benchmarking uncertainty quantification methods for large language models with lm-polygraph. *Transactions of the Association for Computational Linguistics*, 13:220–248, 2025.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

Xi Wang, Laurence Aitchison, and Maja Rudolph. Lora ensembles for large language model fine-tuning. *arXiv preprint arXiv:2310.00035*, 2023a.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023b.

Zhiqiu Xia, Jinxuan Xu, Yuqian Zhang, and Hang Liu. A survey of uncertainty estimation methods on large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 21381–21396, 2025.

Yasin Abbasi Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. To believe or not to believe your llm. *arXiv preprint arXiv:2406.02543*, 2024.

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

Zhengmao Ye, Dengchun Li, Zetao Hu, Tingfeng Lan, Jian Sha, Sicong Zhang, Lei Duan, Jie Zuo, Hui Lu, Yuanchun Zhou, and Mingjie Tang. mLoRA: Fine-tuning LoRA adapters via highly-efficient pipeline parallelism in multiple GPUs. *arXiv:2312.02515*, 2024.

Mingqi Yuan, Man-On Pun, and Dong Wang. Rényi state entropy maximization for exploration acceleration in reinforcement learning. *IEEE transactions on artificial intelligence*, 4(5):1154–1164, 2022.

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

Yuanzhao Zhai, Yu Lei, Han Zhang, Yue Yu, Kele Xu, Dawei Feng, Bo Ding, and Huaimin Wang. Uncertainty-penalized reinforcement learning from human feedback with diversified reward lora ensembles. *Information Processing & Management*, 63(3):104548, 2026.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

APPENDIX

A NOTATION

Table 4: Symbols used in Sec. 2.

Symbol	Meaning
K, k	number of LoRA adapters (ensemble members) and member index
r	LoRA rank; $A^{(k)} \in \mathbb{R}^{r \times d_{\text{in}}}$ (down-projection), $B^{(k)} \in \mathbb{R}^{d_{\text{out}} \times r}$ (up-projection)
Θ_{base}, $\Theta^{(k)}$	frozen base weights; weights of member k, $\Theta_{\text{base}} + B^{(k)}A^{(k)}$
s, q, d	parent state (a previously verified program), prompt embedding (d, s), task description
G, i	group size (rollouts sampled from one parent state) and rollout index within the group
$o = (o_1, \ldots, o_{\lvert o \rvert})$	a rollout (thinking tokens and code); $o_{<t}$ its prefix before position t
$R^{(i)}_{\text{exec}}$, $R_{\max}$	verifier reward of the program in rollout i; best reward reached during a run
$\beta(s)$, β_{ref}	adaptive entropic temperature of the group; reference temperature in Eq. (4)
w_i, A_i	leave-one-out entropic weight and advantage (Eq. (10))
$p_{k,t}$, $\bar{p}_t$	next-token distribution of member k at position t; their arithmetic mixture
MI_t	full-vocabulary ensemble mutual information at position t (Eq. (1))
U_i, $\tilde{U}_i$	top-7% mean of MI_t over the scored positions of rollout i; its group-standardised, clipped value
$\overline{\lvert A \rvert}$	mean absolute advantage of the group (advantage scale)
α, $\gamma_{\max}$, $\gamma_{\text{eff}}(s)$	base exploration rate, cap ratio, and effective exploration coefficient (Eq. (4))
A^{shaped}_i	disagreement-shaped advantage (Eq. (5))
W_ℓ, $\lVert \cdot \rVert_*$, L	stacked down-projections at layer ℓ, nuclear norm, number of adapted linear layers
λ_{KL}, λ_{NNM}	weights of the KL-to-base correction (App. C) and of the NNM regulariser (Eq. (7))
H	Shannon entropy: over the vocabulary in MI_t; over algorithmic families on the four mathematical tasks (§3); over k-means clusters of program embeddings on every task, the rule-free measure of App. M (bits)

B TASKS, REWARDS AND REFERENCE VALUES

All six tasks, their prompts, and their verifiers are taken unchanged from TTT-DISCOVER (Yuksekgonul et al., 2026). **AC1** asks for a step function on a 1000-element grid whose autocorrelation inequality constant C_1 is as small as possible; the verifier returns $R = 1/C_1$. **AC2** asks for a non-negative step function maximising the autocorrelation lower-bound constant C_2; $R = C_2$. **CP26** asks for a packing of 26 circles in the unit square maximising the sum of radii; R is that sum. **Erdős** asks for a step function bounding the Erdős minimum-overlap constant C_5 from above; $R = 1/C_5$. **AHC039** asks for a C++ program for an AtCoder Heuristic Contest geometry problem; R is the contest scoring function evaluated on 150 vendored inputs from the public generator (seeds 0–149), the same at every evaluation: R is the mean per-case score divided by 1500, and $R = 0$ unless all 150 cases are accepted; the official hidden tests are not used, and both arms compile with the same host toolchain, so only arm-to-arm differences are meaningful. **Denoising** is the OpenProblems single-cell denoising task: the program receives one half of a molecular cross-validation split of the Pancreas (inDrop1) count matrix and returns denoised counts; the verifier computes the mean squared error (MSE) between the log-normalised denoised matrix and the held-out molecules, and a Poisson score. A program is valid when its Poisson score is at least 97% of the way from the no-denoising reference to the ideal value, and then $R = 1/\text{MSE}$ (otherwise $R = 0$). The split (seed 42)

is the same at every evaluation, so its held-out molecules are queried throughout the search and do not form a final test set; the published protocol reports the selected denoisers on the held-out PBMC and Tabula Muris Senis lung datasets, which we did not do. Because $R = 1/\mathrm{MSE}$, a relative gain in R is a smaller relative drop in MSE: $+9.5\%$ in R is -8.7% in MSE ($0.2130 \to 0.1945$) at Qwen3-8B, and $+2.95\%$ is -2.9% ($0.2048 \to 0.1990$) at Qwen3-14B.

The AHC039 and denoising rewards have no same-model published reference, so they do not enter Tab. 5. That table converts the four mathematical rewards of Tab. 1 to their constants and lists the values reached by the published TTT-DISCOVER runs with a 120B-parameter model at 50 steps of 512 rollouts (on AC2 the best prior AI value, 0.9610, is higher (Yuksekgonul et al., 2026)); our comparisons are same-model, same-budget differences, and our values remain below the published 120B values on AC1, AC2 and Erdős. The CP26 UG-TTT run of Tab. 1 reaches the value 2.635983 reported for the published TTT-DISCOVER run on the same Qwen3-8B model (Tab. 3 of Yuksekgonul et al. (2026)), which used 50 steps of 512 rollouts (their §4.1.2 and Tab. 9); it does so within a 384-rollout budget, first at epoch 3. This compares completed budgets, not the number of rollouts either method needed to first reach the value, and not compute.

Table 5: Constants implied by the four mathematical rows of Tab. 1 (exact $R_{\max}$ inverted where $R = 1/\text{bound}$), and the values achieved by the published TTT-DISCOVER runs (120B model, 25,600 rollouts; the CP26 entry is the published Qwen3-8B run, Tab. 3 of Yuksekgonul et al., 2026). Constants are computed from unrounded $R_{\max}$ and then rounded, so they can differ in the last digit from constants obtained by inverting the rounded R of Tab. 1. ShinkaEvolve (Lange et al., 2025) reports the same CP26 value within 150 program evaluations using an ensemble of closed frontier models, so evaluation counts to this value vary by orders of magnitude across systems; our 384-versus-25,600 statement compares the budgets of two runs of the same model and code only.

Task	Constant	Baseline	UG-TTT	TTT-DISCOVER (published)
AC1	C_1 (upper bound, ↓)	1.5673	1.5611	1.50287
AC2	C_2 (lower bound, ↑)	0.8532	0.8563	0.9591
CP26	sum of radii (↑)	2.6302	2.6360	2.635983 (Qwen3-8B)
Erdős	C_5 (upper bound, ↓)	0.38205	0.38215	0.380876

C BASELINE OBJECTIVE AND UPDATE, AS IMPLEMENTED

Entropic objective. For a parent state s, TTT-DISCOVER's per-parent objective is the log-partition

$$J_s(\theta) \;=\; \log \mathbb{E}_{o\sim\pi_\theta(\cdot|q,s)}\big[e^{\beta(s)\,R_{\mathrm{exec}}(o)}\big], \tag{8}$$

averaged over parents drawn from the PUCT reuse distribution, with $\beta(s)$ treated as a constant (it is computed from the rewards and not differentiated). Its gradient is $\mathbb{E}[\,w_\beta(o)\nabla_\theta \log \pi_\theta(o)\,]$ with $w_\beta(o) = e^{\beta R_{\mathrm{exec}}(o)}/\mathbb{E}[e^{\beta R_{\mathrm{exec}}}]$, which the leave-one-out weights below estimate from the group; this is the weighting both TTT-DISCOVER's reference implementation and ours use. The normalised utility $\beta(s)^{-1}J_s$ explains the objective's intent: it interpolates between the expected reward ($\beta \to 0$) and the per-parent maximum ($\beta \to \infty$), and for small β it equals $\mathbb{E}[R] + \frac{\beta}{2}\mathrm{Var}(R) + O(\beta^2)$, so at a fixed mean it rewards rather than penalises reward variance. Its gradient carries an extra factor $\beta(s)^{-1}$ that differs across groups, so the implemented update follows Eq. (8), not the normalised utility.

Adaptive temperature. For each group, $\beta(s)$ targets

$$\mathrm{KL}\big(q_\beta \,\big\|\, \mathrm{Unif}_G\big) \;=\; \ln 2, \qquad q_\beta(i) \propto \exp\big(\beta\, R^{(i)}_{\mathrm{exec}}\big). \tag{9}$$

For non-constant rewards, $\mathrm{KL}(q_\beta\|\mathrm{Unif}_G)$ is continuous and strictly increasing in β, from 0 at $\beta = 0$ to $\log(G/m)$ as $\beta \to \infty$, where m is the number of rollouts tied at the group maximum (the limit of q_β is uniform over them). A finite root therefore exists if and only if $m < G/2$; with G=8, four tied maxima reach $\ln 2$ only in the limit, and five or more cannot reach it. The solver doubles an upper bracket from 1 until the KL reaches $\ln 2$ or the bracket reaches 2^{20}, then bisects 60 times; when the target is not reached it returns $\beta = 2^{20}$, at which the weights equal their $\beta \to \infty$ limit to

working precision (with exactly four tied maxima, rounding lets it stop at a large finite β instead). All softmax and KL evaluations are shifted by the group maximum (log-sum-exp), so they remain stable at large β. In the twelve reported UG-TTT runs the solver returned its ceiling in 57 of 576 trained groups (9.9%), 50 of them groups whose execution rewards all tie, where $A_i \equiv 0$; in the $K{=}1$ baselines, which drop fully tied groups, it did so in 12 of 486 (2.5%) (App. N).

Leave-one-out advantages. Given $\beta(s)$, rollout $i \in \{1, \dots, G\}$ receives

$$w_i \;=\; \frac{\exp\!\big(\beta\, R^{(i)}_{\text{exec}}\big)}{\frac{1}{G-1}\sum_{j\neq i}\exp\!\big(\beta\, R^{(j)}_{\text{exec}}\big)}, \qquad A_i \;=\; w_i - 1 \;=\; \frac{(G-1)\,q_i}{1-q_i} - 1, \tag{10}$$

with $q_i = q_\beta(i)$. These are leave-one-out entropic advantages with a constant baseline, not group-centred ones: $\sum_i A_i = (G-1)\sum_i q_i/(1-q_i) - G \geq 0$ by convexity of $x \mapsto x/(1-x)$, with equality only for uniform weights (for rewards $(1, 0, \dots, 0)$ at $G{=}8$, $\beta \approx 2.47$ and $\sum_i A_i \approx 6.53$). In practice the offset is large: over the trained groups of the reported UG-TTT runs the within-group sum has median 2.1, is at least 1 in 90% of groups, and reaches 6.53 when a single rollout is best. Because $\beta(s)$ is solved for a fixed KL, the weights are invariant to an affine rescaling $R_{\text{exec}} \mapsto c + \varepsilon R_{\text{exec}}$ ($\varepsilon > 0$): β scales as $1/\varepsilon$ and q_β is unchanged, as long as the root lies inside the bracket. Rollouts whose shaped advantage has magnitude below 10^{-8} are not trained on.

Policy loss. Every reported run uses the unclipped token-level importance-weighted loss of the TTT-DISCOVER reference configuration. For adapter k and the n_b rollouts trained in one optimiser step,

$$\mathcal{L}^{(k)}_{\text{PG}}(\theta_k) \;=\; -\frac{1}{n_b}\sum_i \frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \rho^{(k)}_{i,t}(\theta_k)\,\hat{A}_{i,t}, \qquad \rho^{(k)}_{i,t}(\theta_k) = \frac{\pi_{\theta_k}(o_{i,t} \mid q, o_{i,<t})}{\pi_{\text{b}}(o_{i,t} \mid q, o_{i,<t})}, \tag{11}$$

where π_{b} is the logged sampling distribution of the adapter that generated rollout i, and $\hat{A}_{i,t}$ is A_i^{shaped} plus the per-token KL correction below (the baseline, $K{=}1$, uses A_i). No clipping is applied; a clip parameter present in our configuration files is not read by this loss. For the generating adapter at the start of a step the ratio equals one up to LoRA-dropout noise, since numerator and denominator are computed under independent dropout masks (App. F); for the other $K{-}1$ adapters it reweights each token at its sampled prefix. This is a surrogate: an exact change of measure between members' sequence distributions would require products of per-token ratios along each prefix, which we do not use, and the shared shaped advantages come from rollouts of several members. Per-token ratios were not logged.

Token-wise KL correction. With $d_{i,t} = \log \pi_{\text{b}}(o_{i,t} \mid \cdot) - \log \pi_{\text{base}}(o_{i,t} \mid \cdot)$, computed from the generating adapter's logged sampling log-probabilities and the frozen base, and $\bar{d}$ its mean over all response tokens of the group, the per-token advantage is $\hat{A}_{i,t} = A_i^{\text{shaped}} + \lambda_{\text{KL}}\,(\bar{d} - d_{i,t})$. Under sampling from π_{b} this is the single-sample (k1) estimator of the gradient of $\lambda_{\text{KL}}\text{KL}(\pi_{\text{b}} \| \pi_{\text{base}})$, centred within the group and folded into the advantage, as in the reference implementation. It is shared by all K member losses, so each member is anchored to the base through the generating member's log-ratio rather than its own.

Constant-reward filter. Groups whose *total* rewards $R^{(i)}_{\text{exec}} + \alpha U_i$ are all equal are dropped before the update. With $\alpha > 0$ the disagreement term varies, so groups whose execution rewards all tie are kept: 50 of the 576 trained groups of the reported UG-TTT runs, almost all of them groups in which every program failed. Their shaped advantages are at most 3×10^{-8} in magnitude, and their rollouts enter the loss, if at all, essentially only through the KL correction. With $\alpha = 0$ such groups are dropped (42 of 528 groups in the baselines). The disagreement bonus therefore supplies no exploration signal in groups where every program fails.

D TRAINING ALGORITHM

Algorithm 1 One UG-TTT optimiser step at epoch e (B parent groups of G rollouts).

Require: Ensemble $\{\theta_k\}_{k=1}^{K}$; PUCT buffer; coefficients α, β_{ref}, $\gamma_{\max}$, λ_{KL}, λ_{NNM}.

$\mathcal{T} \leftarrow \varnothing$ ▷ trained rollouts of this step
for $b = 1, \dots, B$ **do**
  Select a parent s from the PUCT buffer; build the prompt q from (d, s)
  for $i = 1, \dots, G$ **do**
    $k_i \leftarrow \big((eG + i - 1) \bmod K\big) + 1$; $o_i \sim \pi_{\theta_{k_i}}(\cdot \mid q)$, logging sampling log-probabilities
    $R_{\text{exec}}^{(i)} \leftarrow \textsc{Verifier}(o_i)$
  end for
  Score every (q, o_i) through all K adapters in one batched pass; compute $\text{MI}_t^{(i)}$ at every scored position (Eq. (1)) and U_i (Eq. (2))
  if all $R_{\text{exec}}^{(i)} + \alpha U_i$ are equal **then continue** ▷ constant-reward filter
  end if
  $\beta(s) \leftarrow \textsc{SolveBeta}(R_{\text{exec}}^{(1:G)})$; $A_i \leftarrow w_i - 1$ ▷ Eqs. (9), (10)
  $\tilde{U}_i$ by Eq. (3); $\gamma_{\text{eff}}(s)$ by Eq. (4); $A_i^{\text{shaped}} \leftarrow A_i + \gamma_{\text{eff}}(s)\,\overline{|A|}\,\tilde{U}_i$ ▷ Eq. (5)
  $\hat{A}_{i,t} \leftarrow A_i^{\text{shaped}} + \lambda_{\text{KL}}(\bar{d} - d_{i,t})$ ▷ App. C
  Add every rollout with $|A_i^{\text{shaped}}| \geq 10^{-8}$ to $\mathcal{T}$
  Update the PUCT buffer with the verified programs
end for
$n_b \leftarrow |\mathcal{T}|$
for each rollout in $\mathcal{T}$ and $k = 1, \dots, K$ **do**
  Forward adapter k; call `backward()` on its per-rollout term of $\frac{1}{K}\mathcal{L}_{\text{PG}}^{(k)}$; free the graph ▷ Eq. (11)
end for
Call `backward()` on $\lambda_{\text{NNM}}\,\mathcal{L}_{\text{NNM}}/n_b$ ▷ Eq. (6)
AdamW step on every adapter; zero gradients

In all reported runs B=8 and there is one optimiser step per epoch. The baseline runs the same loop with K=1, α=0 and λ_{NNM}=0 and skips the ensemble scoring pass. Note that the regulariser enters with weight λ_{NNM}/n_b relative to the per-rollout policy loss.

E PROOFS

E.1 PROPOSITION 1 (THE CLIP AND THE SIGNED SUM OF SHAPED ADVANTAGES)

Proposition 1. *Let $z_i = (U_i - \overline{U})/\sigma_U$ and $\mathcal{C} = \{i : |z_i| > 3\}$. Then $\big|\sum_i A_i^{\text{shaped}} - \sum_i A_i\big| \leq \gamma_{\text{eff}}(s)\,\overline{|A|}\,|\mathcal{C}|(\sqrt{G-1} - 3)$, and the difference vanishes whenever the clip is inactive ($\mathcal{C} = \varnothing$); the unbiased normalisation $\sum_i z_i^2 = G - 1$ forces $\mathcal{C} = \varnothing$ for $G \leq 10$.*

Fix a parent state s and its group of G rollouts. If $\sigma_U = 0$, Eq. (3) sets $\tilde{U}_i \equiv 0$ by convention, and $A_i^{\text{shaped}} = A_i$ identically. Otherwise

$$\sum_{i=1}^{G} \tilde{U}_i \;=\; \sum_{i=1}^{G} \text{clip}(z_i,\, -3,\, 3)\,.$$

By centring, $\sum_{i=1}^{G} z_i = 0$, so

$$\sum_{i=1}^{G} \tilde{U}_i \;=\; \sum_{i \notin \mathcal{C}} z_i \;+\; \sum_{i \in \mathcal{C}} \text{sign}(z_i) \cdot 3 \;=\; \sum_{i \in \mathcal{C}} \big(\text{sign}(z_i) \cdot 3 - z_i\big).$$

For $i \in \mathcal{C}$, $|\text{sign}(z_i) \cdot 3 - z_i| = |z_i| - 3$, so $\big|\sum_i \tilde{U}_i\big| \leq \sum_{i \in \mathcal{C}} (|z_i| - 3)$. Since $\sum_i z_i^2 = G - 1$, each $|z_i| \leq \sqrt{G-1}$, giving $\sum_{i \in \mathcal{C}} (|z_i| - 3) \leq |\mathcal{C}|(\sqrt{G-1} - 3)$, and $\mathcal{C} = \varnothing$ whenever $G \leq 10$.

Substituting into Eq. (5), where neither $\gamma_{\text{eff}}(s)$ nor $\overline{|A|}$ depends on i,

$$\left|\sum_{i=1}^{G} A_i^{\text{shaped}} - \sum_{i=1}^{G} A_i\right| \;=\; \gamma_{\text{eff}}(s)\,\overline{|A|}\left|\sum_{i=1}^{G} \tilde{U}_i\right| \;\le\; \gamma_{\text{eff}}(s)\,\overline{|A|}\,|\mathcal{C}|\big(\sqrt{G-1}-3\big).$$

The statement concerns the signed within-group sum of advantages only; it bounds neither the sum of absolute advantages, nor the gradient norm, nor gradient bias. The substitution $A_i \mapsto A_i^{\text{shaped}}$ does not preserve the expected policy gradient (Thrampoulidis et al., 2025), since $\tilde{U}_i$ depends on rollout i: the bonus is deliberately biased. □

E.2 Proposition 2 (equal-norm maximiser)

Fix a layer ℓ and write $A_k := A_\ell^{(k)} \in \mathbb{R}^{r\times d_{\text{in}}}$. The problem is

$$\max_{A_1,\dots,A_K} \big\|W\big\|_*, \qquad W = \big[A_1;\dots;A_K\big] \in \mathbb{R}^{Kr\times d_{\text{in}}}, \qquad \|A_k\|_F = c\;\forall k.$$

Let $\sigma_1,\dots,\sigma_m$ be the nonzero singular values of W, with $m = \operatorname{rank}(W) \le Kr$ (as $Kr \le d_{\text{in}}$). By Cauchy–Schwarz,

$$\|W\|_*^2 \;=\; \Big(\sum_{i=1}^{m}\sigma_i\Big)^2 \;\le\; m\sum_{i=1}^{m}\sigma_i^2 \;\le\; Kr\cdot\|W\|_F^2 \;=\; K^2 r\,c^2,$$

with equality iff $m = Kr$ and all Kr singular values are equal. Their common value s satisfies $Kr\,s^2 = Kc^2$, so $s = c/\sqrt{r}$. At full row rank, equal singular values are equivalent to $WW^\top = s^2 I_{Kr}$: the rows of W are mutually orthogonal with common norm s. Within a block this gives A_k r singular values equal to $c/\sqrt{r}$; across blocks it makes the row spaces of $A_1,\dots,A_K$ mutually orthogonal, which is feasible because $Kr \le d_{\text{in}}$. □

Remarks. (i) The first inequality, $\|W\|_* \le \sqrt{\operatorname{rank}(W)}\,\|W\|_F \le \sqrt{Kr}\,\|W\|_F$, holds for any W, without the equal-norm constraint; the ratio $\rho(W) = \|W\|_*/(\sqrt{Kr}\,\|W\|_F) \in (0,1]$ measures how close a stacked matrix is to the equal-singular-value configuration at its own scale, and App. F reports it for our initialisation. (ii) The proposition says nothing about predictions: with $B^{(k)} = 0$ for all k, a maximiser leaves every member equal to the base model and $\text{MI}_t \equiv 0$. (iii) The raw objective $-\|W_\ell\|_*$ has no minimiser: along $A^{(k)} \to tA^{(k)}$, $B^{(k)} \to B^{(k)}/t$ the members are unchanged while $\|W_\ell\|_*$ grows linearly in t.

E.3 Proposition 3 (TTT-Discover as a special case)

Proposition 3. *If $\lambda_{\text{NNM}} = 0$ and either (a) $K = 1$, or (b) all K adapters define the same policy, then $\mathcal{L}_{\text{total}}$ in Eq. (7) reduces to the* TTT-Discover *objective of Yuksekgonul et al. (2026).*

Under either branch the mixture $\bar{p}_t$ coincides with each member's distribution, so $\text{MI}_t \equiv 0$, $U_i \equiv 0$, $\sigma_U = 0$, and by the convention below Eq. (3) $\tilde{U}_i \equiv 0$, hence $A_i^{\text{shaped}} = A_i$. In branch (b) the K member losses are then identical, so their average equals one of them; in branch (a) there is one loss. The hypothesis $\lambda_{\text{NNM}} = 0$ removes $\mathcal{L}_{\text{NNM}}$ (it is needed even when $K = 1$, since $-\|A_\ell^{(1)}\|_*$ depends on θ_1). What remains is the importance-weighted entropic loss of App. C with its KL correction, which is the TTT-Discover objective. This is a statement about objectives; identical optimiser trajectories would additionally require identical dropout draws, gradient scaling and optimiser state. □

F Implementation notes

Chunked scoring. Two logit tensors arise during scoring. (i) The ensemble-MI path forms a $(K,\,\text{chunk},\,V)$ fp32 logit tensor in a chunked LM head; for Qwen3-8B with $K = 5$, $V = 152{,}064$ and $\text{chunk} = 256$, that tensor occupies 0.78 GB ($K\cdot\text{chunk}\cdot V\cdot 4$ bytes), and the log-softmax and probability tensors derived from it have the same size. The two entropy terms of Eq. (1) reuse the chunk's log-softmax computed for the target-token gather; we did not time them separately. (ii) The base-model log-probabilities used by the KL correction would require a $(1, T, V)$ fp32 tensor of

15.8 GB at T=26,000 if formed at once; we stream the log-softmax over chunks of 256 positions instead.

Shared frozen base. Members built on different pretrained bases would disagree for reasons of pretraining data that do not decay as the adapters fit the task. A shared base also keeps one copy of the base weights; each adapter still needs its own forward computation, because adapters on the attention projections change every downstream hidden state, and the K passes are batched. This is a design argument; we did not run a cross-base ensemble.

Scored positions and the top-7% mean. Wang et al. (2025) argue that the reinforcement-learning signal in LLMs concentrates on a high-entropy minority of tokens. Our U_i averages the $\lfloor 0.07\, n \rfloor$ largest values of MI_t (at least one) over the n scored positions of (q, o_i), prompt included; the prompt is identical within a group. Across the twelve full-context UG-TTT runs, the median share of the top-7% positions that fall in the prompt is 5–14% at Qwen3-8B, below the prompt's 13–27% share of all scored positions, and 21% at Qwen3-14B, so U_i is dominated by response tokens. The fraction is a heuristic; App. N compares aggregations only on a logged surrogate score.

LoRA dropout. LoRA dropout (p=0.05) is applied inside the LoRA branch of every adapted projection in every forward pass, including sampling and scoring: the branch calls dropout functionally, so switching the model to evaluation mode does not disable it. Sampling, scoring and training passes therefore draw independent masks; the base path is deterministic.

Sequential per-adapter backward. A joint K-adapter forward–backward retains K computation graphs and exceeds 80 GB of activation memory in our setting, so the training pass iterates over adapters (Alg. 1; the mLoRA scheme of §2.5): each adapter's forward and backward are computed in turn and its graph is freed before the next; gradients accumulate for the final AdamW step.

Adapter initialisation. Each $A^{(k)}$ receives mLoRA's default Kaiming-normal initialisation and each $B^{(k)}$ small Gaussian entries ($\sigma = 0.01$) instead of zeros, member k with its own seed, so the members differ from the first forward pass and Eq. (1) is non-degenerate at epoch 0. The K=1 baseline's adapter is initialised with the same seed and code as the ensemble's first member, so the baseline starts from one of the ensemble's initial hypotheses rather than from the unmodified base model. The zero-initialised CP26 comparator of §3.3 lacks this diversification: its members are functionally identical before training.

Nuclear-norm geometry at initialisation and during training. For the stacked down-projection $W_\ell \in \mathbb{R}^{Kr \times d_{\text{in}}}$ of Prop. 2, write $\rho(W_\ell) = \|W_\ell\|_* / (\sqrt{Kr}\, \|W_\ell\|_F) \le 1$. With Kaiming-normal rows and $d_{\text{in}} \gg Kr$ the rows are already nearly orthogonal: averaged over the 144 adapted projections of Qwen3-8B, $\rho = 0.9975$ at initialisation for K=5 (0.9990 for K=2, 0.9985 for K=3), and 0.9980 for the 160 projections of Qwen3-14B. A CPU replay of the seeded initialisation reproduces every logged step-0 nuclear norm to 5×10^{-6} relative. The regulariser can therefore raise $\|W_\ell\|_*$ by at most 0.25% without raising $\|W_\ell\|_F$. The logged layer-mean nuclear norm rises monotonically in every run and, from step 0 to the last logged step (five updates; the weights after the sixth are not logged), by 0.8–1.7% in the full-context K=5 runs (seed-42 pairs: CP26 0.78%, AC1 1.07%, AC2 1.20%, Erdős 1.36%, AHC039 1.50%, denoising 1.02%; Qwen3-14B 1.66%), by 0.4–0.6% in the reduced-context pilot arms, and by 0.29% (K=2) and 0.58% (K=3). Every run exceeds its geometric headroom, by 1.5–8×. Since $\|W\|_* \le \sqrt{Kr}\, \|W\|_F$ survives averaging over layers, the mean Frobenius norm must have grown by at least 0.5–1.5% in the full-context runs: at least two thirds of the logged growth is scale. The logged scalar does not determine whether ρ rose or fell, says nothing about $B^{(k)}$ or $B^{(k)}A^{(k)}$, and cannot separate growth driven by the regulariser from growth driven by the policy gradient, because the arms without NNM do not log the norm. Values and the replay are in the released verification scripts.

Restricting the regulariser to $A^{(k)}$. In pilot runs, regularising both $A^{(k)}$ and $B^{(k)}$ produced oscillatory optimisation. Restricting the regulariser to $A^{(k)}$ avoided that behaviour but does not remove the function-preserving rescaling between $A^{(k)}$ and $B^{(k)}$ (Prop. 2, remark (iii)); the paragraph above

bounds how much of the logged growth can be geometry, and whether the measured reward effect of NNM (§3.3) acts through scale is unresolved.

G HYPERPARAMETERS AND RUN CONFIGURATIONS

Tables 7 and 8 list the hyperparameters of all reported runs; values are identical across the six tasks unless noted. Every run uses one NVIDIA RTX PRO 6000 Blackwell GPU (96 GB). The seed reruns, the AHC039 and denoising pairs, the K sweep and the Qwen3-14B pair used PyTorch 2.9.1, CUDA 12.8 and transformers 4.57; the logs of the original AC1, AC2, CP26 and Erdős seed-42 pairs and of the component pilot do not record the software versions.

What each comparison varies. Table 6 lists, for each group of reported comparisons, what differs between the two arms besides the UG-TTT components, and hence what the comparison can establish.

Table 6: **Comparison protocol.** "Components" are the ensemble (K=5), the disagreement bonus (α=0.1) and NNM (λ_{NNM}=0.075). All pairs share the seed, task prompt, verifier, token limits and the 384-rollout budget; realised token counts differ.

Comparison	Differences between arms	What it can establish
Tab. 1/2 pairs with a K=1 baseline (all but the two below)	the three components; the baseline's adapter has the same initial draw as the ensemble's first member	effect of the full recipe against TTT-DISCOVER at this budget
CP26, seed 42	comparator is a zero-initialised K=5 ensemble without NNM, bonus active on near-zero MI; streaming on both arms, firing at different rates	full recipe against a degenerate-ensemble control
AC1, seed 42	the three components, plus streaming early-stop on the UG-TTT arm only	full recipe with streaming against the baseline without it
AC1, seeds 2 and 3	the three components; both arms' prompts carry a previous best program	within-row comparisons under a changed starting condition
Denoising, Qwen3-14B	the three components; base model and seed differ from the Qwen3-8B pair	recurrence of the denoising comparison, isolating neither factor
Tab. 3 arms	one component each, at context 8,192, one run per arm	pilot evidence on components in that configuration

Configuration variants. Three run families deviate from the shared configuration, in the stated ways only. (i) The component pilot of Tab. 3: context window 8,192 (phase-1 cap 6,000), seed 42; the arms differ from one another only in the ablated component. (ii) The K=2, 3 runs: full configuration at seed 2 with a 32,000-token generation cap, which never binds (App. K). (iii) The Qwen3-14B denoising pair: base model Qwen3-14B and seed 1. Streaming is on only where Tab. 8 lists it.

Interrupted runs. The AC1 seed-42 baseline and both AHC039 seed-42 arms were interrupted once and resumed, so each log carries one re-sampled epoch (448, 440 and 448 logged rollouts). On resume the adapters and the PUCT buffer reload from the last completed epoch's checkpoint, so the interrupted pass contributes nothing to the weights or the buffer; the AdamW moments are not checkpointed and restart from zero at the resumed epoch. Separately, in the AHC039 seed-42 UG-TTT run 48 of the 384 kept rollouts (six groups, epochs 2–5) received prompts truncated at the tokeniser's 4,096-token cutoff, and 73% of those failed to execute; the paired baseline had no prompt over that length. We report the pair as logged. We count every logged rollout in $R_{\max}$ and in the diversity measures; $R_{\max}$ is unchanged if the interrupted partial epochs are set aside.

Larger base model. One pair runs on Qwen3-14B (denoising, seed 1; context 32,768; streaming off on both arms; 384 logged rollouts per arm). Baseline/UG-TTT: $R_{\max}$ 4.8823/5.0261 (+0.1438, +2.95%; MSE 0.2048 → 0.1990), first attained at epoch 5/3; best reward within epoch 0

$4.7045/4.6878$, so the baseline leads before the first optimiser step; correct rollouts $108/135$; generated tokens $3.37/3.33$ million. On this pair the whole-run structural measure favours the baseline ($\Delta H = -0.36$ bits, median over k; App. M).

Table 7: **Shared hyperparameters** (identical for TTT-DISCOVER and UG-TTT on all six tasks).

Hyperparameter	Value	Notes
Model		
Base model	Qwen3-8B	fp16; Qwen3-14B for the seed-1 denoising pair
LoRA target modules	q, k, v, o projections	all attention projections
LoRA rank r	16	
LoRA scaling α_{LoRA}	32	effective scale $= \alpha_{\mathrm{LoRA}}/r = 2$
LoRA dropout	0.05	active in every forward pass (App. F)
LoRA initialisation	A: Kaiming normal; B: $\mathcal{N}(0, 0.01^2)$	per-member seed; baseline = member 1's draw
Optimiser		
Optimiser	AdamW	β_1=0.9, β_2=0.95, ϵ=10^{-8}, weight decay 0
Learning rate	4×10^{-5}	constant
Gradient clipping	none	no LR schedule or warmup
RL training		
Epochs / optimiser steps	6 / 6	one step per epoch
Group size G	8	rollouts per parent
Groups per optimiser step	8	64 rollouts per step
Rollout temperature	1.0	
Advantage estimator	entropic adaptive-β	App. C
β_{ref}	2.0	$\beta(s)$ solved for KL $= \ln 2$
Policy loss	importance-weighted, unclipped	Eq. (11); token mean per rollout
Loss reduction	mean over trained rollouts and adapters	rollouts with $\lvert A\rvert < 10^{-8}$ skipped
KL correction λ_{KL}	0.01	group-centred, per token (App. C)
Remove constant-reward groups	yes	tested on $R_{\mathrm{exec}} + \alpha U$
Sampling, context and verifier		
Sampler	PUCT with backpropagation	as in TTT-DISCOVER
Context window	32 768 tokens	8,192 in the component pilot
Thinking-token cap	26 000	
Code budget after a streaming stop	6 000 tokens	streaming runs only
Verifier limits	1000 s program budget; 1000–1100 s timeout; 1–2 CPUs	per task and run family; AHC039 uses the ALE-Bench judge (2 s per case)

Table 8: **UG-TTT-specific hyperparameters.** The baseline sets K=1, α=0, λ_{NNM}=0 and disables streaming; the CP26 seed-42 comparator is the exception of Tab. 6.

Hyperparameter	Value	Notes
Ensemble		
Number of adapters K	5	epoch-rotated round-robin generation
Disagreement bonus (Sec. 2.3)		
Score	full-vocabulary ensemble MI	Eq. (1); top-7% mean over scored positions, Eq. (2)
Base rate α	0.1	
$\gamma_{\max}$	10.0	caps $\gamma_{\text{eff}}/\alpha$ (Eq. (4))
Standardised-score clip	$[-3, 3]$	never binds for $G \leq 10$
Nuclear-norm regulariser (Sec. 2.4)		
λ_{NNM}	0.075	on $A^{(k)}$ only; weight λ_{NNM}/n_b per step
Normalisation	none	raw nuclear norm
Streaming early-stop (App. H)		
Enabled	AC1, CP26 (seed 42)	AC1: UG-TTT arm; CP26: both arms
Disabled	all other runs	both arms
Window size W	4 096 tokens	
Check interval C	2 048 tokens	
Min generation before check	4 096 tokens	
MI threshold percentile	25th (CP26), 5th (AC1)	of historical window MI
Warmup epochs	3	no streaming before epoch 3

H STREAMING MI EARLY-STOP AND LENGTH-REWARD DIAGNOSTIC

Cost and reward price. Streaming early-stop caps the thinking phase at 4,096 tokens from epoch 3; in the streamed Tab. 1 runs its MI gate truncated every rollout that reached the cap (192/192 on both AC1 and CP26), saving 569 and 684 tokens per rollout. Paired controls show it costs $R_{\max}$ on AC2 (-0.0148), on Erdős (-0.0017) and, at reduced context, on CP26 (-0.0083; Tab. 3), so streaming is a token-cost lever with a per-task reward price.

Mechanism as logged. The streaming early-stop is implemented as a hard cap on the thinking phase: when a rollout reaches 4096 thinking tokens (occasionally 6144 on Erdős where the cap was widened), the windowed ensemble MI is compared once to a running percentile of historical MI (25th on CP26, 5th on AC1); a rollout below threshold is closed and forced into a bounded code phase, a rollout above threshold would continue to natural termination. Streaming is inactive during the warm-up epochs 0–2. In the two streamed Tab. 1 runs the gate truncated every rollout that reached the cap in epochs 3–5 (192/192 on AC1 and on CP26; every rollout exceeded 4096 thinking tokens), so over those epochs the mechanism acted as a thinking-length cap, and the whole-run firing rate of 50.0% is the share of the six-epoch budget that falls in the active epochs; a baseline with the same cap was not run. The recorded `streaming_mi_stop_step` coincides with the cap because that is where the gate is queried.

Per-task firing and savings (six-epoch budget, 384 rollouts).

Task	Firing rate	Tokens saved / rollout	Tokens saved / firing	Setting in Tab. 1
AC1	50.0%	569	1,138	enabled
CP26	50.0%	684	1,368	enabled
AC2	disabled	—	—	disabled
Erdős	disabled	—	—	disabled
AHC039	disabled	—	—	disabled
Denoising	disabled	—	—	disabled

The CP26 comparator of Tab. 1 also had streaming enabled and fired on $41/384$ (10.7%) of its rollouts, saving $51{,}582$ tokens in total; the AC1, AC2 and Erdős baselines ran with streaming off, as did both arms of every AHC039 and denoising pair.

Paired AC2/Erdős comparison (epochs 3–5). We forked AC2 from a shared epoch-2 checkpoint and ran the remaining three epochs with and without streaming; on Erdős we ran the two configurations as separate full runs and aligned epochs 3–5 for the comparison.

	$R_{\max}$ (stream)	$R_{\max}$ (no stream)	Mean length (stream)	Mean length (no stream)
AC2	0.8415	**0.8563**	5,312	8,711
Erdős	2.6150	**2.6167**	5,255	8,720

On AC2 the streamed arm (0.8415) also falls below the baseline (0.8532), so the Tab. 1 AC2 gain depends on the streaming-off setting selected from this pilot; we disclose the selection rather than remove it, and the seed-2/3, AHC039 and denoising rows of Tab. 2, like the AHC039 and denoising rows of Tab. 1, run streaming off on both arms.

Length–reward diagnostic. We test whether the within-run rank correlation between rollout length and execution reward separates the four tasks on which streaming was calibrated (it was never enabled on AHC039 or denoising). Restricting to correct rollouts and parsing the `</think>` marker to split thinking from code tokens (streamed rollouts use the logged `phase1_tokens` / `phase2_tokens` fields directly), we report Spearman ρ for epochs 0–2 and for all epochs:

	Epochs 0–2			All epochs		
Task	ρ_{think}	ρ_{code}	ρ_{total}	ρ_{think}	ρ_{code}	ρ_{total}
AC1	-0.30	$+0.52$	-0.19	-0.51	$+0.48$	-0.43
CP26	$+0.16$	-0.10	$+0.15$	-0.34	-0.14	-0.41
AC2	-0.28	$+0.39$	-0.21	-0.18	$+0.36$	-0.11
Erdős	-0.18	$+0.12$	-0.16	-0.11	$+0.07$	-0.10

The thinking-phase correlation is not significantly positive on any problem and is significantly negative on AC1; the code-phase correlation is positive on AC1 and AC2, weakly positive on Erdős, and null on CP26. No single test on length predicts whether streaming will help or hurt $R_{\max}$ on a given problem; the per-problem calibration in this paper is therefore empirical (paired pilot on AC2/Erdős) rather than diagnosable in advance from a length–reward test.

What streaming-truncated rollouts look like in practice. On AC1 and CP26 the rollouts that achieve the Table 1 $R_{\max}$ are themselves streaming-truncated: AC1 $R_{\max}$=0.6406 from a rollout of length $5{,}116$ (`streaming_mi_stopped`=1); CP26 $R_{\max}$=2.6360 from a rollout of length $5{,}318$ (also truncated). On AC2, Erdős, AHC039 and denoising the Tab. 1 entries are produced from no-streaming runs (on AC2 and Erdős, the two pilot tasks of this appendix, the $R_{\max}$ rollouts have lengths $5{,}642$ and $7{,}316$). We report this asymmetry directly because it is recoverable from the trajectory logs.

I Program motifs and their observed adoption

Table 9 lists, for three mathematical tasks, the motif used by the highest-reward UG-TTT rollout, how many of each arm's 30 highest-reward rollouts contain it, and its adoption among all correct rollouts during training. The motifs are established techniques rather than new mathematics: a closed-form target sum $\sqrt{2n}$ motivated by a Cauchy–Schwarz bound on AC1, a library differential-evolution solver on AC2, and a Fourier initialisation $h(x) = a + b\sin(kx)$ on Erdős. Each is rare or absent among the baseline's correct rollouts (whole-run adoption 3.0%, 0.9% and 0%) and common under UG-TTT. This is consistent with the exploration hypothesis, but it does not show that the disagreement bonus caused the adoption: the arms also differ in ensemble sampling, and per-rollout MI does not predict which rollouts win (App. O). On CP26 the best rollouts of both arms belong to the dominant `hex_grid` family, so no row is given.

Table 9: **Program motifs and their adoption.** For each task, the motif of the highest-reward UG-TTT rollout (a family of the released classifiers), the number of each arm's 30 highest-reward rollouts that contain it, and its adoption (motif uses divided by correct rollouts over training). CP26 is omitted because both arms' best rollouts share the dominant family; AHC039 and denoising have no validated taxonomy (App. M).

		Top-30 (count)		Adoption (%)	
Task	Motif	Baseline	UG-TTT	Baseline	UG-TTT
AC1	`target_sum` $= \sqrt{2n}$ (Cauchy–Schwarz)	1/30	18/30	3.0	28.0
AC2	`scipy` differential evolution ∪ gradient-based	0/30	13/30	0.9	21.6
Erdős	Fourier initialisation $h(x) = a + b\sin(kx)$	0/30	6/30	0	7.0

J CASE STUDY: THE BEST AC2 PROGRAMS

We compare the programs behind the best AC2 construction of each arm (Tab. 1, $+0.0031$). For the baseline this is its highest-reward rollout. For UG-TTT, the highest-reward construction ($R = 0.856348$) first appears at epoch 3 in a multi-start SLSQP program; two later rollouts (epochs 4 and 5), differential-evolution programs whose initial population contains their parent's construction, return the same construction to within 10^{-12} and are the nominal maximum only by rounding. We therefore show the SLSQP program that produced it. Both listings reproduce the algorithmic core of the model-emitted program, abridged where boilerplate (clipping, sum repairs, timers, logging) repeats.

Problem statement (AC2)

Construct a sequence $h \in \mathbb{R}^{1000}_{\geq 0}$ of step-function heights whose autocorrelation lower bound C_2 (returned by the closed-form checker `evaluate_sequence`) is as large as possible, subject to $\sum_i h_i \geq 0.01$ and $h_i \in [0, 1000]$. Reward equals C_2 for a valid sequence and 0 otherwise. The harness exposes the parent state's construction to the program as the global `height_sequence_1`. Higher is better.

TTT-DISCOVER (best of 384 rollouts, $R_{\max} = 0.8532$, hand-written genetic algorithm)

```
POPULATION_SIZE = 200; MAX_GENERATIONS = 500
MUTATION_RATE_INITIAL, MUTATION_RATE_FINAL = 0.25, 0.05

# half: perturb height_sequence_1; half: random uniform over [0,
    200]
population = generate_initial_population()

for gen in range(MAX_GENERATIONS):
    fitness = [evaluate_sequence(s) for s in population]
    if max(fitness) > best_value:
        best_value, best_seq, stamina = max(fitness), argmax_seq, 0
    else:
        stamina += 1
    if stamina > 80:                          # restart on stagnation
        population = generate_initial_population(); stamina = 0

    parents = [tournament_selection(population, fitness, size=5)
               for _ in range(POPULATION_SIZE)]
    new_pop = [parents[0]]                    # elitism
    while len(new_pop) < POPULATION_SIZE:
        p1, p2 = random.sample(parents, 2)
        w = random.uniform(0.3, 0.7)          # blend crossover
        child = [p1[i]*w + p2[i]*(1-w) for i in range(SEQ_LENGTH)]
```

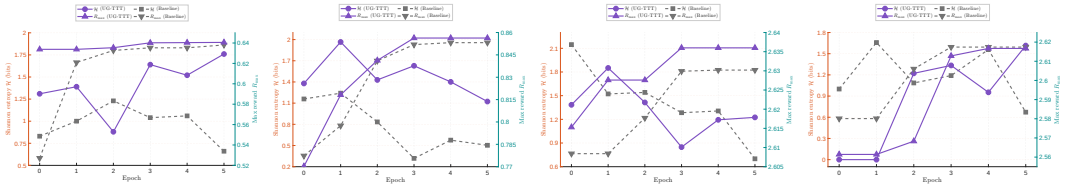
```
        rate = MUTATION_RATE_INITIAL - (gen/MAX_GENERATIONS)*(
                MUTATION_RATE_INITIAL - MUTATION_RATE_FINAL)
        for i in range(SEQ_LENGTH):
            if random.random() < rate:
                child[i] = clip(child[i] + random.gauss(0, 1.5))
        new_pop.append(child)

    best_seq = local_search(best_seq)       # +/- delta sweep on
        elite
    population = new_pop
```

Algorithmic family. Pure-Python evolutionary loop: tournament selection of size 5, blend crossover with weight in $[0.3, 0.7]$, Gaussian mutation with rate decaying linearly from 0.25 to 0.05, restart after 80 generations without improvement, and a coordinate-wise sweep on the elite. No library optimiser is called. Library optimisers (differential evolution or gradient-based) appear in 2 of the 220 correct baseline rollouts and in none of its 30 best (Tab. 9).

(epoch 3; first rollout to reach 0.856348, multi-start SLSQP)

```
from scipy.optimize import minimize

bounds = [(0.0, 1000.0)] * 1000
constraints = [{'type': 'ineq', 'fun': lambda x: np.sum(x) - 0.01}]

def objective(x):                           # maximise C_2
    try:
        return -evaluate_sequence(x.tolist())
    except:
        return -float('inf')

initial_guesses = [
    np.random.rand(1000) * 0.1,                                    #
        uniform
    np.random.rand(1000) * 0.2,                                    #
        uniform
    np.sort(np.clip(np.random.normal(0.025, 0.05, 1000), 0, 1000)),
         # sorted
]
if 'height_sequence_1' in globals():        # the parent's
    construction
    initial_guesses.append(np.clip(np.array(height_sequence_1), 0,
        1000))

best_score, best_seq = -float('inf'), None
for guess in initial_guesses:               # one local search per
    start
    result = minimize(objective, guess, method='SLSQP', bounds=
        bounds,
                      constraints=constraints, tol=1e-5,
                      options={'maxiter': 500, 'ftol': 1e-6, 'gtol':
                          1e-6})
    seq = np.clip(result.x, 0.0, 1000.0)
    score = evaluate_sequence(seq.tolist())
    if score > best_score:
        best_score, best_seq = score, seq
return best_seq.tolist()
```

Algorithmic family. Gradient-based local search: four SLSQP runs with box bounds and the sum constraint, started from two uniform random vectors, one sorted Gaussian vector and the parent's construction; the best result is returned. The program did not log which start produced the returned construction. Library optimisers account for 13 of the 30 best UG-TTT AC2 rollouts (Tab. 9).



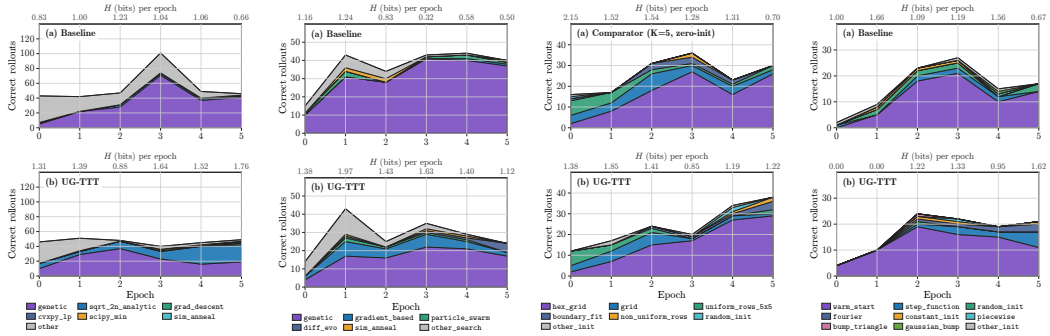
(a) AC1 (b) AC2 (c) CP26 (d) Erdős

Figure 2: **Training dynamics on the four mathematical tasks** (seed 42). Purple solid lines: UG-TTT; gray dashed lines: comparator (the $K{=}5$ zero-initialised control on CP26, the $K{=}1$ baseline elsewhere). Circles and squares: per-epoch family entropy H of correct rollouts (left axis, released classifiers); triangles: best reward so far (right axis). Each comparator ends with lower family entropy than UG-TTT; UG-TTT ends with the higher $R_{\max}$ on AC1, AC2 and CP26 and 0.0007 lower on Erdős.

Differences between the selected AC2 programs

The baseline program searches with a hand-written genetic algorithm; the UG-TTT program refines several starting points, including its parent, with a library gradient-based optimiser. These are different algorithms, and the comparison does not isolate which difference produces the $+0.0031$ gap; the construction also passes through the PUCT buffer, so later UG-TTT rollouts re-submit it rather than improve on it. Library optimisers appear among UG-TTT's best AC2 rollouts and not among the baseline's, whose family distribution concentrates on genetic algorithms during training (Fig. 3b). We did not measure token-level disagreement at the positions where the programs diverge, and per-rollout MI does not predict which rollouts win (App. O), so we do not attribute the choice of optimiser to the disagreement bonus.

K TRAINING DYNAMICS, SEED ROBUSTNESS, AND ENSEMBLE SIZE



(a) AC1 (b) AC2 (c) CP26 (d) Erdős

Figure 3: **Family composition of correct rollouts per epoch on the four mathematical tasks** (seed 42, released classifiers). In each panel the top plot is the comparator (the $K{=}1$ baseline; on CP26 the zero-initialised $K{=}5$ control) and the bottom plot is UG-TTT; the per-epoch family entropy H (bits) is printed above each plot. Vocabularies are task-specific and shared by both arms. Final-epoch H is 1.76/1.12/1.22/1.62 bits under UG-TTT and 0.66/0.50/0.70/0.67 under the comparator (Tab. 10).

Seed-level entropy. Table 10 accompanies the six CP26/AC1 rows of Tab. 2 with final-epoch and whole-run family entropy and adds the AC2 and Erdős seed-42 pairs; the AHC039 and denoising rows are ranked by the rule-free measure (Tab. 12). On both measures the entropy gap favours UG-TTT in four of six seed pairs and reverses in two (CP26 seed 2 and AC1 seed 3), so, as with reward, we make no cross-seed significance claim. Figure 4 visualises the reward pairs.

Table 10: Family entropy H (bits, correct rollouts) per seed: final epoch (top) and whole run over epochs 0–5 (bottom). All values use the released classifiers (App. K); the seed-42 final-epoch values are those discussed in §3.

	CP26			AC1			AC2	Erdős
Seed	2	3	42	2	3	42	42	42
Baseline (ep. 5)	1.92	0.94	0.70	0.44	1.90	0.66	0.50	0.67
UG-TTT (ep. 5)	0.91	1.24	1.22	1.53	1.07	1.76	1.12	1.62
Baseline (whole run)	1.82	1.64	1.62	0.82	1.89	1.20	0.84	1.43
UG-TTT (whole run)	1.47	1.94	1.64	1.39	1.06	1.74	1.79	1.30

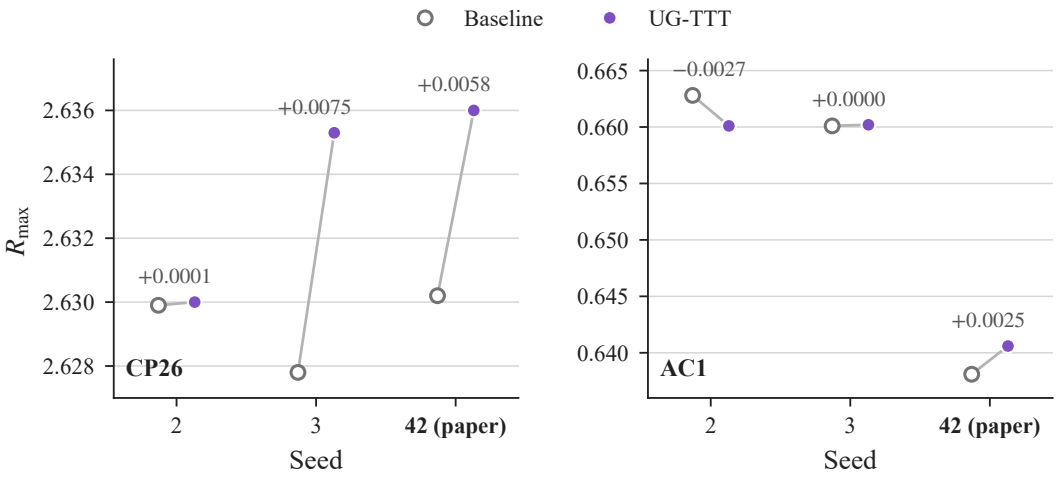


Figure 4: **Seed robustness of** $R_{\max}$ on CP26 (left) and AC1 (right): per-seed baseline–UG-TTT pairs; the AHC039 and denoising pairs are drawn in Fig. 7. Values as in Tab. 2.

Family classifiers and counts. Each mathematical task uses one released rule-based classifier throughout the paper: `ac1_tag_family` (AC1), `tag_search` (AC2) and the task-specific `tag_init` classifiers (CP26, Erdős); the AC2 script counts a rollout when it is correct and its reward is at least 0.5, which removes one rollout per arm relative to the correctness filter of the other tasks. The fallback family `other` covers 33.5%/20.1% of correct AC1 rollouts (comparator/UG-TTT), 7.8%/17.1% on AC2, 0.7%/2.8% on CP26 and 4.3%/0.0% on Erdős; AC1's vocabulary covers its solution space least completely, so AC1 family entropies carry that caveat. Correct-rollout counts over the six epochs (comparator vs. UG-TTT) are AC1 328/448 vs. 279/384, AC2 220 vs. 171, CP26 153 vs. 145 and Erdős 93 vs. 100. The best reward within epoch 0, among the 64 rollouts drawn before the first optimiser step, is AC1 0.5271 vs. 0.6338, AC2 0.7773 vs. 0.7702, CP26 2.6084 vs. 2.6153 and Erdős 2.5801 vs. 2.5614; each run's $R_{\max}$ is first attained at epoch 5/5 (AC1), 4/3 (AC2), 4/3 (CP26) and 3/4 (Erdős). Every run ends above its epoch-0 best, but further sampling alone gives the best-so-far more chances to improve, and without a frozen-ensemble search control this does not isolate the effect of the updates. The two arms' epoch-0 family entropies already differ by 0.48 (AC1), 0.22 (AC2), 0.76 (CP26) and 1.00 bits (Erdős). These gaps reflect finite-sample variation and the fact that UG-TTT's first epoch is already sampled from five distinct adapters; they are why comparisons that hold the initial policies fixed across arms matter.

Ensemble size. At seed 2 on CP26 (full configuration), $K{=}2/3/5$ reach $R_{\max} = 2.6343/2.6307/2.6300$ at this one seed, with $K{=}2$ highest; we do not read an ordering into one seed, and our results establish $K \leq 5$, not that $K{=}5$ is optimal (§3.2). The sweep ends with final-epoch $H = 1.27/1.28/0.91$ (Fig. 5). The $K{=}2/3$ runs used a smaller generation cap (32,000 tokens); the cap never binds (longest observed rollout 20,271 tokens, no rollout within 3% of the cap), so the comparison is clean.

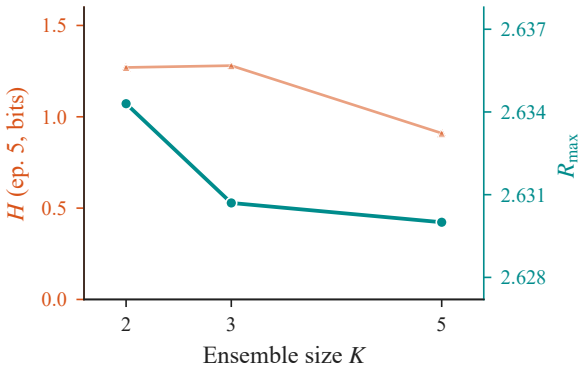


Figure 5: **Ensemble-size sweep on CP26** (seed 2, full configuration): $R_{\max} = 2.6343/2.6307/2.6300$ at K=2/3/5; one run per K, no ordering read into one seed.

L Component pilot: score definitions and diversity detail

Scores. All arms of Tab. 3 score each rollout's sequence through the K members. The full method uses the full-vocabulary MI of Eq. (1) and the top-7% mean of Eq. (2). The two alternative arms use only the members' log-probabilities of the sampled token o_t and average over all scored positions: the token-entropy arm uses $-\bar{p}_t(o_t)\log\bar{p}_t(o_t)$ with $\bar{p}_t(o_t) = \frac{1}{K}\sum_k p_{k,t}(o_t)$, a sampled-token term of the mixture's entropy, and the member-variance arm uses the unbiased variance across members of $\log p_{k,t}(o_t)$. Every score is then standardised and shaped identically (Eqs. (3)–(5)). The alternatives therefore differ from the MI bonus in three ways at once: the functional, the restriction to the sampled token, and the aggregation. For small member deviations $p_{k,t} = \bar{p}_t + \delta_{k,t}$, Eq. (1) is approximately $\frac{1}{2K}\sum_{k,v}\delta^2_{k,t,v}/\bar{p}_{t,v}$ (nats), a squared disagreement weighted inversely by the mixture probability of each token, whereas the variance score measures disagreement in log-probability at the sampled token only; neither score is aleatoric by construction, and the pilot does not show which property, if any, explains the observed ordering.

Family entropy. Table 11 reports per-epoch family entropy for the six arms, computed with the released CP26 classifier (App. K). No arm ends in a single family (all epoch-5 $H \geq 0.35$ bits), and at this reduced context every arm except the variance arm (1.37) and the no-NNM arm (0.82), the full method included, ends at $H \leq 0.59$ bits. Over the whole run, the variance arm is the most diverse ($\Delta H = +0.61$ bits against the full method; rollout-permutation $p = 0.001$, a descriptive statistic, App. M), and every other arm is within 0.16 bits of the full method. The pilot therefore gives the MI bonus no diversity advantage over the other scores; its observed advantage is on reward (Tab. 3).

Table 11: Per-epoch and whole-run family entropy H (bits, correct rollouts) for the Tab. 3 arms, with whole-run differences from the full method ($\Delta H = H_{\text{arm}} - H_{\text{full}}$) and descriptive rollout-permutation p-values.

Arm	ep. 0	ep. 1	ep. 2	ep. 3	ep. 4	ep. 5	whole	ΔH	p
Full UG-TTT	1.35	0.68	0.70	0.34	0.53	0.35	0.69	—	—
Token entropy	1.41	0.90	0.44	0.37	0.20	0.59	0.66	−0.03	0.86
Member variance	1.35	0.98	0.83	1.21	1.08	1.37	1.30	+0.61	0.001
α=0	1.85	1.07	0.35	0.46	0.84	0.45	0.85	+0.16	0.36
No NNM	1.41	0.64	0.00	0.40	0.50	0.82	0.68	−0.01	0.95
Streaming on	1.41	0.00	0.66	0.35	0.59	0.37	0.69	+0.00	1.00

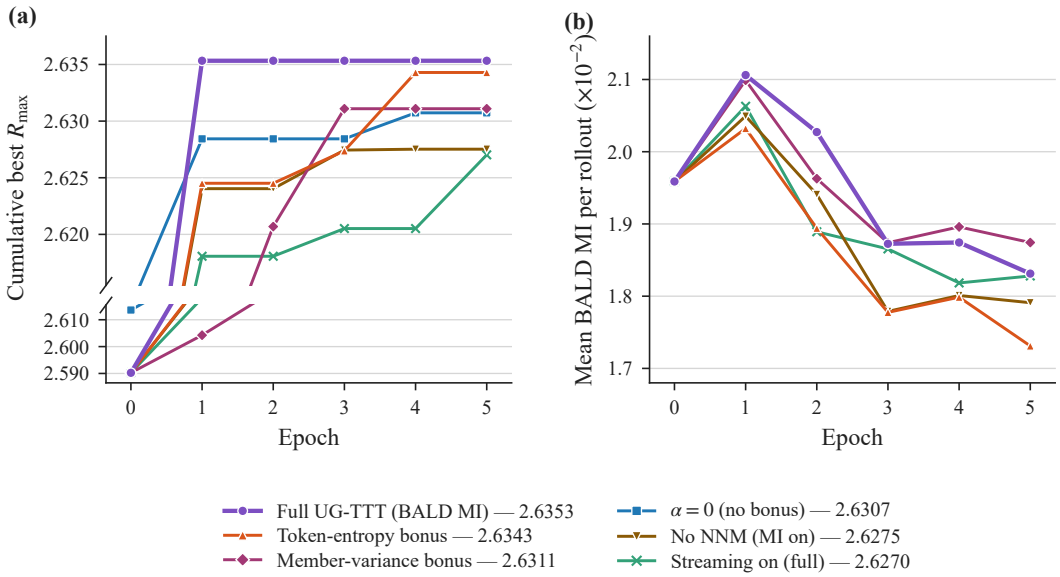


Figure 6: **Ablation-arm dynamics** (matched configuration of Tab. 3). Left: cumulative best $R_{\max}$ per epoch for the six arms. Right: per-epoch mean ensemble MI for the five arms that compute it (the α=0 arm skips the computation); the no-NNM arm retains MI throughout this six-epoch horizon, and both it and the full arm decline $\approx$13% from their epoch-1 peaks.

MI trajectories at full configuration. In the Tab. 1 configuration, the zero-initialised no-NNM comparator's per-epoch mean MI runs from 2.5×10^{-9} (epoch 0, before any gradient step) to 6.6×10^{-6} (epoch 5), while the full method holds 1.9–2.2×10^{-2} across all epochs (epoch-5 means 6.6×10^{-6} versus 2.2×10^{-2}): the 3.5-order gap is present before any update, which is why §3.3 attributes it to the initialisation rather than to NNM. Because Eq. (3) standardises within each group, that numerically tiny MI still enters the comparator's advantage as an $O(1)$ term: the comparator is a degenerate-ensemble control running the full objective, not an α=0 run, and it differs from the K=1 baseline of the other pairs. In the streamed Tab. 1 CP26 runs, streaming fired on 50.0% of the full method's rollouts versus 10.7% of the comparator's (App. H).

M RULE-FREE STRUCTURAL DIVERSITY ON EVERY TASK

Measure and statistics. Each correct rollout's program is embedded by an AST fingerprint (Python) or a TF–IDF lexical n-gram profile (C++, AHC039); both arms' embeddings are clustered jointly by k-means at $k \in \{6, 8, 10\}$ (clustering seed 0), and each arm's cluster entropy is computed over the whole run. This whole-run measure was fixed before the Qwen3-14B pair was run and is the primary rule-free endpoint. Table 12 lists all twelve run pairs. Next to each difference we print a rollout-level permutation p-value, obtained by permuting arm labels within the pair. Rollouts share parents, policy checkpoints and an adaptively updated buffer, so they are not exchangeable; these p-values neither measure run-to-run variability nor test the algorithms, and we treat them, like the reseed counts, as descriptive measures of how clearly the two arms' samples separate. Pair-level statements in the paper rest on signs and magnitudes only.

Robustness of the measure. On the three Qwen3-8B AHC039 and denoising pairs, all 45 reseed cells (five clustering reseeds $\times$ three k) are positive, and re-embedding the Qwen3-8B denoising pair with the lexical pipeline gives ΔH between $+0.43$ and $+0.81$. On the Qwen3-14B pair (108/135 correct programs, baseline/UG-TTT) all three k are negative ($-0.38/-0.36/-0.30$; per-arm H at k=8 2.75/2.39), 14 of the 15 reseed cells are negative, the lexical embedding agrees in sign at every k, and the sign persists under every structural variant we tried (k = 4–20, Ward clustering, clustering-free pairwise and nearest-neighbour distances, equal sample sizes). Successful-program counts differ between arms, so we also recompute every cell at a common sample size, subsampling the larger arm to the smaller arm's count (500 draws, clustering fixed): no cell changes sign, no cell

moves by more than 0.022 bits, and UG-TTT remains the more diverse arm in 7 of 12 pairs, as it does when all 24 arms are cut to one common size of 69 programs. The pair count does depend on the clustering seed: it is 7 of 12 at four of five clustering seeds and 8 of 12 at the fifth, where the Erdős pair changes sign.

No taxonomy for AHC039 and denoising. A ten-family rule taxonomy written for these two tasks after inspecting their rollouts gives whole-run family entropies of $2.65/1.92$ (AHC039 seed 2), $2.12/2.22$ (AHC039 seed 42), $2.45/2.05$ (denoising, Qwen3-8B) and $2.01/2.20$ (denoising, Qwen3-14B) for baseline/UG-TTT, the opposite sign to the structural measure on three of the four pairs. With as few as 18–22 correct rollouts at epoch 5 (denoising, Qwen3-8B) against ten families, we consider it too coarse to rank the arms; Tab. 12 reports it (‡) but the paper does not use it.

Table 12: Whole-run structural-entropy difference ΔH (bits, UG-TTT minus comparator; correct rollouts, epochs 0–5; clustering seed 0) with descriptive rollout-permutation p-values, the number of positive cells over five clustering reseeds $\times$ three k, and the whole-run family-entropy difference under the released classifiers (mathematical tasks) or, marked ‡, under the rejected ten-family taxonomy. The seed-1 denoising row is the Qwen3-14B pair; the CP26 seed-42 comparator is the degenerate-ensemble control of Tab. 1. Python programs are embedded by AST fingerprint, C++ (AHC039) by TF–IDF lexical n-grams.

Task	Seed	k=6	k=8	k=10	Pos. cells	Family ΔH
AC1	2	-0.04 (0.53)	-0.48 (<0.001)	-0.25 (0.020)	1/15	$+0.57$
AC1	3	$+0.13$ (0.005)	$+0.43$ (<0.001)	$+0.38$ (<0.001)	15/15	-0.83
AC1	42	-0.24 (<0.001)	-0.60 (<0.001)	-0.46 (<0.001)	0/15	$+0.55$
AC2	42	$+0.89$ (<0.001)	$+0.50$ (<0.001)	$+0.47$ (<0.001)	15/15	$+0.94$
CP26	2	$+0.05$ (0.35)	$+0.10$ (0.035)	$+0.15$ (0.026)	14/15	-0.35
CP26	3	-0.51 (<0.001)	-0.72 (<0.001)	-0.15 (0.042)	1/15	$+0.29$
CP26	42	$+0.47$ (<0.001)	$+0.04$ (0.58)	$+0.47$ (<0.001)	14/15	$+0.02$
Erdős	42	-0.11 (0.39)	$+0.11$ (0.35)	-0.27 (0.023)	5/15	-0.13
AHC039	2	$+0.22$ (0.008)	$+0.13$ (0.20)	$+0.42$ (<0.001)	15/15	$-0.73^{‡}$
AHC039	42	$+0.36$ (<0.001)	$+0.38$ (<0.001)	$+0.32$ (<0.001)	15/15	$+0.10^{‡}$
Denoising	42	$+0.67$ (<0.001)	$+0.46$ (0.017)	$+0.35$ (0.001)	15/15	$-0.41^{‡}$
Denoising	1 (14B)	-0.38 (<0.001)	-0.36 (<0.001)	-0.30 (0.006)	1/15	$+0.19^{‡}$

Family and structural measures disagree. On the mathematical pairs the structural measure favours UG-TTT on AC2, on CP26 at seeds 42 and 2 and on AC1 at seed 3, and the comparator on AC1 at seeds 42 and 2, on Erdős and on CP26 at seed 3; it agrees in sign with whole-run family entropy on only three of the eight mathematical pairs (AC2, CP26 seed 42 and Erdős). Family entropy counts algorithmic families (on AC1, for instance, whether a rollout uses the closed-form target or a genetic search), whereas structural entropy counts structurally distinct implementations, so an arm that emits many variants of one family can be structurally diverse while concentrated in one family, and conversely. Neither is a validated measure of useful exploration or of novelty, which is why §3.4 reports both and draws no causal conclusion from either.

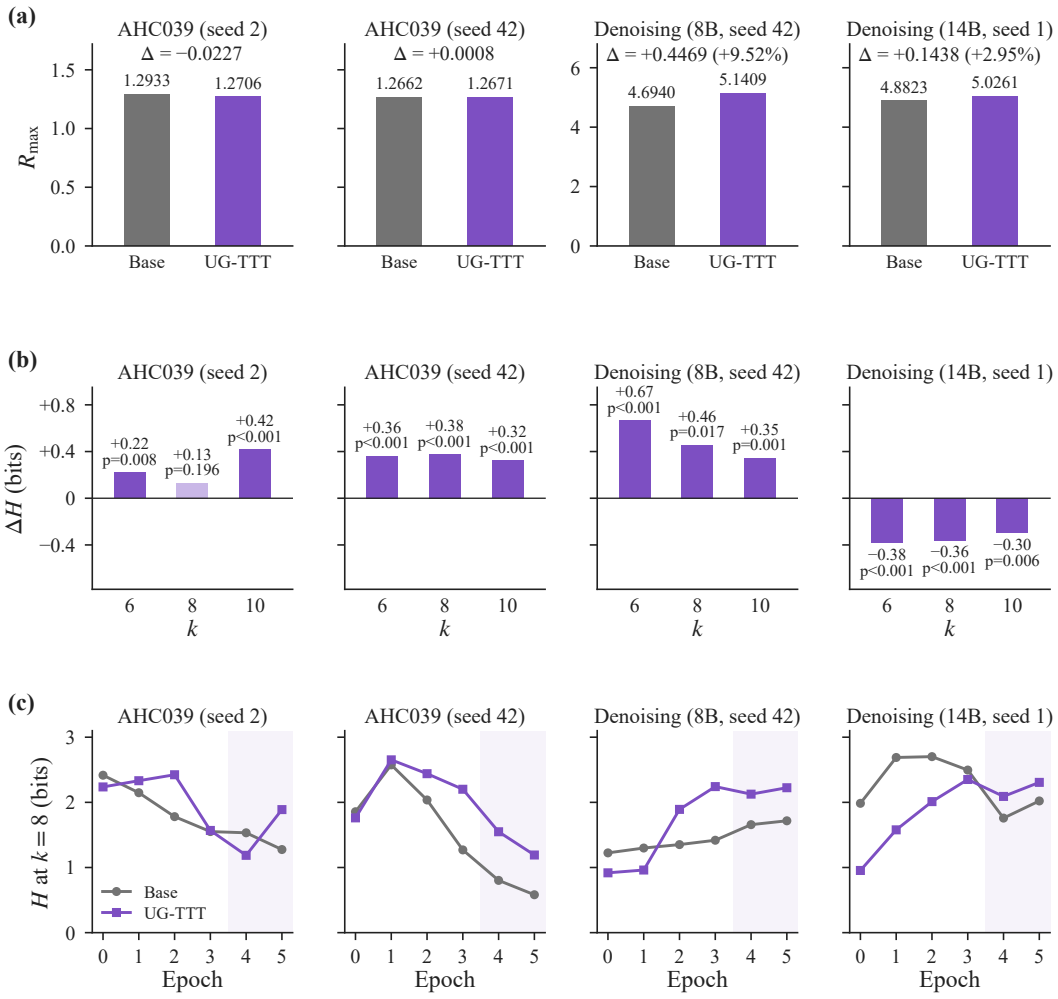


Figure 7: **Reward and structural diversity on the AHC039 and denoising pairs.** **(a)** Per-arm $R_{\max}$ for the four pairs (the AHC039 and denoising rows of Tab. 2; the AHC039 difference changes sign across seeds; the seed-1 denoising pair runs on Qwen3-14B). **(b)** Whole-run structural-entropy difference versus k with descriptive permutation p-values: UG-TTT is the more diverse arm in all nine cells of the three Qwen3-8B pairs and the less diverse one in all three cells of the Qwen3-14B pair. **(c)** Per-epoch structural entropy at k=8 for both arms (8–70 correct programs per arm and epoch); the shaded epochs 4–5 are the late window of Tab. 13.

Epoch-window view. The hypothesis of §1 predicts that a diversity gap opens late in training, so we split each pair's joint clustering into epochs 0–2 and 4–5 (Tab. 13, all twelve pairs). The split was computed after the whole-run analysis and the Qwen3-14B reversal; the whole-run measure therefore remains the conservative primary endpoint, and the window is reported for every pair whatever its sign. Over epochs 4–5, UG-TTT is the more structurally diverse arm in 27 of 36 pair×k cells (12 of 12 on the four AHC039 and denoising pairs, 15 of 24 on the mathematical pairs), and in 9 of 12 pairs by the sign of the median over k (whole run: 7 of 12). The Qwen3-14B reversal is concentrated in epochs 0–2 ($\Delta H = -0.88$ at k=8) and changes sign over epochs 4–5 ($+0.25/+0.32/+0.11$ at $k = 6/8/10$). On AC1 seed 42 and on Erdős the comparator remains the more structurally diverse arm late in training, whereas the family classifier favours UG-TTT on AC1 seed 42 over epochs 4–5 (+0.76 bits) and is nearly tied on Erdős (+0.03). On denoising, UG-TTT's per-epoch structural entropy (at k=8) starts low and rises ($0.92 \to 2.22$ at Qwen3-8B, $0.95 \to 2.31$ at Qwen3-14B) while the baseline's does not fall ($1.22 \to 1.72$; $1.99 \to 2.02$); on AHC039 the baseline's falls at both seeds ($2.42 \to 1.28$; $1.86 \to 0.58$). These per-epoch values rest on 8–70 programs per arm and epoch and are descriptive.

Table 13: Epoch-window view: structural-entropy difference ΔH (bits, UG-TTT minus comparator; descriptive within-window permutation p) over epochs 0–2 and 4–5 for every run pair of Tab. 12, from the same joint clustering; computed after the Qwen3-14B whole-run reversal and reported for every pair whatever its sign.

		Epochs 0–2			Epochs 4–5		
Task	Seed	k=6	k=8	k=10	k=6	k=8	k=10
AC1	2	−0.72 (<0.001)	−1.18 (<0.001)	−1.05 (<0.001)	+0.47 (<0.001)	+0.26 (0.008)	+0.52 (<0.001)
AC1	3	+0.20 (0.24)	+0.44 (0.026)	+0.37 (0.082)	+0.49 (<0.001)	+0.64 (<0.001)	+0.78 (<0.001)
AC1	42	+0.02 (0.85)	−0.38 (0.001)	−0.36 (<0.001)	−0.25 (0.035)	−0.27 (0.054)	−0.42 (0.007)
AC2	42	+0.68 (<0.001)	+0.27 (0.001)	+0.57 (<0.001)	+0.66 (<0.001)	+0.53 (0.011)	−0.07 (0.70)
CP26	2	+0.09 (0.50)	0.00 (0.99)	+0.08 (0.50)	+0.36 (<0.001)	+0.51 (<0.001)	+0.49 (<0.001)
CP26	3	−0.54 (<0.001)	−0.52 (0.003)	−0.25 (0.008)	−0.06 (0.68)	−0.39 (0.040)	+0.16 (0.38)
CP26	42	+0.39 (0.023)	+0.20 (0.26)	+0.37 (0.041)	+0.95 (<0.001)	+0.65 (<0.001)	+0.76 (<0.001)
Erdős	42	+0.11 (0.70)	+0.52 (<0.001)	+0.01 (0.95)	−0.11 (0.48)	−0.37 (0.015)	−0.79 (<0.001)
AHC039	2	+0.23 (0.002)	+0.10 (0.24)	+0.25 (0.004)	+0.06 (0.69)	+0.13 (0.56)	+0.57 (0.001)
AHC039	42	+0.07 (0.19)	−0.01 (0.88)	+0.24 (0.002)	+1.02 (<0.001)	+0.65 (0.001)	+0.11 (0.60)
Denoising	42	−0.06 (0.82)	−0.23 (0.54)	−0.14 (0.64)	+0.19 (0.21)	+0.45 (0.033)	+0.57 (0.015)
Denoising	1 (14B)	−0.55 (0.006)	−0.88 (<0.001)	−0.68 (0.008)	+0.25 (0.021)	+0.32 (0.078)	+0.11 (0.59)

N COUNTERFACTUAL ANALYSES OF THE SHAPED UPDATE

Run registry. Every statistic in this appendix uses a de-duplicated registry of trained groups: epochs 0–5 with a logged optimiser step; for an interrupted and resumed epoch, the last logged pass of each group; logs that re-log another run's epochs are excluded; and groups whose eight programs repeat an already registered group (the shared epoch-0 groups of the reduced-context CP26 arms) are counted once. Reconstructing each group from the logged execution rewards and per-rollout scores with the training code reproduces every logged per-step statistic (number of groups and of trained rollouts exactly; mean β, mean γ_{eff} and mean shaped advantage to within 2×10^{-6}). The primary set is the twelve reported UG-TTT arms of Tabs. 1 and 2 (576 trained groups). In these runs the per-rollout score used in training equals the logged top-7% full-vocabulary MI on every rollout.

β-coupling. For every trained group we compute the shaped advantage of Eq. (5) twice on the same rollouts and the same deployed score: with the coupled coefficient (α=0.1, $\gamma_{\max}$=10) and with a strength-matched constant (α=0.37, the realised mean γ_{eff} of a reference run over epochs 0–3, cap ratio 1). Groups whose execution rewards all tie are skipped, since both updates then vanish. Table 14 reports the within-group cosine between the two shaped-advantage vectors and the fraction of the 28 rollout pairs whose order changes. On the twelve reported arms (526 groups) the median cosine is 0.993 (minimum 0.68), 3.0% of pairs change order on average, and 40.5% of groups have at least one change; the difference is bounded by $3\,|\gamma_{\text{c}} - \gamma_{\text{d}}|\,\overline{|A|}$. Of the 576 trained groups, 68.1% lie below the cap ($\beta < \beta_{\text{ref}}\gamma_{\max} = 20$), 31.9% are pinned at $\gamma_{\text{eff}} = \alpha\gamma_{\max}$, 9.9% have β at the solver ceiling, and 8.7% have all execution rewards tied, where the bonus is numerically zero. These are counterfactuals of the update signal on identical rollouts, not decoupled training runs. One decoupled training run exists: with cap ratio 1 and α=0.37 in the reduced-context configuration of Tab. 3, it reaches $R_{\max} = 2.6271$ after four epochs (256 rollouts), within 0.0008 of the median coupled arm of Tab. 3 at the same budget; because the cap removes only the upward branch of Eq. (4), its realised advantage-weighted γ was 0.287 against 0.353 for the coupled reference, so it is a low-power check rather than a strength-matched test.

Top-7% aggregation (surrogate diagnostic). The all-position mean of the deployed full-vocabulary MI was not logged, so the aggregation choice can only be compared on a logged surrogate: the sampled-token Jensen gap $\text{JG}_t = \log \bar{p}_t(o_t) - \frac{1}{K}\sum_k \log p_{k,t}(o_t)$, logged at both aggregations. If o_t is drawn from the mixture, $\mathbb{E}[\text{JG}_t] = \frac{1}{K}\sum_k \text{KL}(\bar{p}_t \,\|\, p_{k,t})$, the reverse-KL counterpart of Eq. (1); the two are different divergences, so this comparison says nothing directly about the deployed update. On the twelve reported arms, replacing the top-7% mean of JG with its all-position mean gives a median within-group cosine of 0.999 (minimum 0.87), 5.1% of pairs reordered, and at least one reordering in 70.3% of groups. The two pilot arms whose deployed score is itself an all-position mean, and whose top-7% version is logged, allow a comparison on a deployed score: median cosines 0.998 (token entropy) and 0.9999 (member variance) over 39 groups each.

Table 14: Counterfactual comparisons of the shaped advantage on identical rollouts (de-duplicated registry; groups with all execution rewards tied excluded). Cosine: within-group cosine between the logged and the alternative shaped-advantage vectors; flips: mean fraction of rollout pairs whose order changes; any flip: share of groups with at least one change.

Comparison	Run set	Groups	Median cos	Mean flips	Any flip
β-coupling vs. constant γ (deployed score)	twelve reported arms	526	0.993	0.030	40.5%
	all 24 ensemble runs with a bonus	974	0.992	0.032	41.9%
Top-7% vs. all-position mean (JG surrogate)	twelve reported arms	526	0.999	0.051	70.3%
	all 23 ensemble runs with a bonus	927	1.000	0.051	71.6%

O DISAGREEMENT DIAGNOSTICS AND THEIR LIMITATIONS

Two properties of the logged per-rollout score bear on what the bonus can and cannot be claimed to do (Fig. 8). Both use the 14 independent ensemble runs with a live disagreement signal on the four mathematical tasks (continuation logs excluded, App. N); the AHC039 and denoising ensembles are not included.

Ensemble disagreement differs from predictive entropy. Across the 13 of these runs that log all three scores, per-rollout MI is uncorrelated or negatively correlated with the sampled-token predictive-entropy score of App. L (median Spearman $\rho = -0.13$; positive in $1/13$ runs) and moderately to strongly correlated with the member-variance score (median $\rho = +0.60$, range $+0.42$ to $+0.90$). The scores are therefore not simple monotone transforms of one another. This distinguishes the estimators; it does not validate the disagreement score as calibrated epistemic uncertainty.

Per-rollout MI does not predict which rollouts win. Across the 14 runs, MI–reward rank correlation is positive in $8/14$ runs (median epoch-adjusted $\rho = 0.02$, Wilcoxon $p = 0.63$), the 30 highest-reward rollouts do not carry higher MI than the rest (median $\Delta z = -0.005$, $p = 1.0$), and each run's $R_{\max}$ rollout sits at a mean MI percentile of 0.55 ($p = 0.39$ against 0.5). Correct rollouts tend to carry *lower* MI than incorrect ones ($3/14$ runs positive; median $\Delta z = -0.20$, binomial $p = 0.057$), so part of what the bonus rewards is disagreement on programs that fail, which is why we report validity alongside reward and diversity. An exploratory proposal need not succeed immediately, so these results neither support nor refute a population-level exploration effect; establishing one would require an intervention such as the controls listed in the Limitations. Mean MI is higher in the code phase than in the reasoning phase in all seven runs where both phases are logged (all reduced-context runs, where rollouts reach the phase-1 cap).

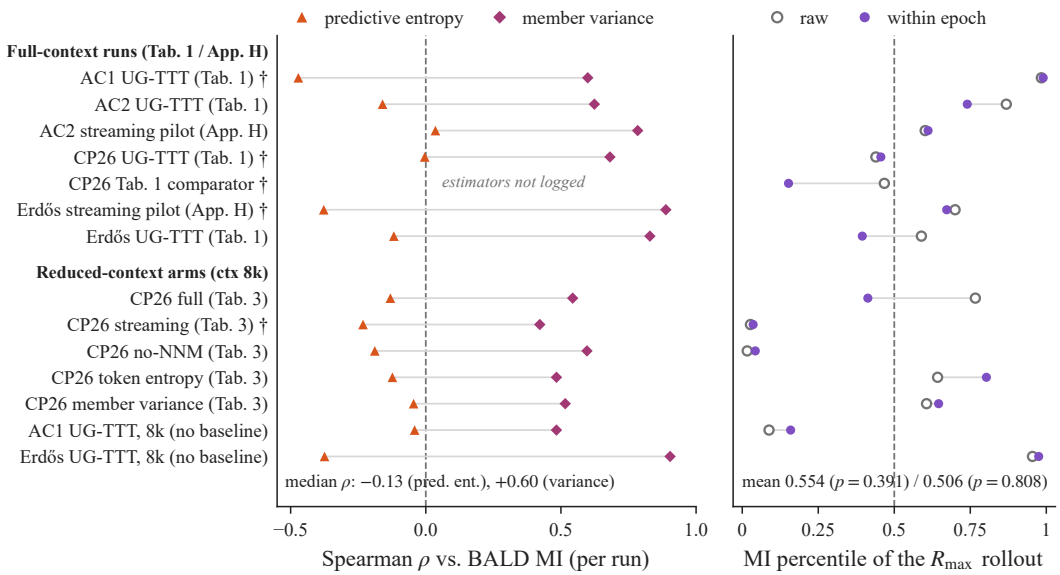


Figure 8: **Disagreement diagnostics.** Rows: the full-context ensemble runs of Tab. 1 (the four mathematical-task UG-TTT arms and the CP26 $K{=}5$ comparator), the App. H pilots, the reduced-context (8,192) arms of Tab. 3, and two single UG-TTT runs on AC1 and Erdős at that context, which enter no reward comparison; † marks streaming on. Left: per-run Spearman correlations of per-rollout MI with the sampled-token predictive-entropy and member-variance scores. Right: MI percentile of each run's $R_{\max}$ rollout. The disagreement score is not a monotone transform of predictive entropy, and it does not predict per-rollout winners.