GLOBAL SELECTION, LOCAL CREDIT: DECOUPLED TOPOLOGY SEARCH FOR ADAPTIVE TEST-TIME SCALING

Anonymous authors
Paper under double-blind review

ABSTRACT

Feedback-guided inference-time tree search is widely used to improve the reasoning of large language models on complex tasks. Existing adaptive methods use feedback to allocate a limited inference budget between generating new solutions and revising existing ones. However, they still traverse the tree level by level to choose a solution to revise, even though a complete solution can serve directly as the starting point. Consequently, whether a candidate receives further computation depends on both its estimated value and its position in the search tree, which can lead to inefficient allocation and limit search performance. To allocate inference compute more efficiently, we propose Decoupled Topology Search (DTS), an inference-time framework that searches over complete solutions. Before each model call, DTS uses the available feedback to decide globally whether to generate a new solution or revise an existing one. If it chooses revision, it selects directly from all valid candidate solutions based on their current quality estimates and uncertainty, without traversing the tree level by level. DTS retains the links through which candidates were generated to propagate feedback and continually update their value estimates. We evaluate DTS on code generation, mathematical reasoning, and molecular optimization. At the same candidate-call budget, DTS outperforms repeated sampling and existing tree-search baselines. On LiveCodeBench v6, DTS improves the Pass@1 of Qwen3-14B by up to 7.2 percentage points over repeated sampling.

1 INTRODUCTION

Recent work has shown that allocating more computation at inference time can improve the performance of large language models (LLMs) on complex tasks (Snell et al., 2025). This computation can be used for reasoning within a single response or for exploring multiple candidate solutions (Brown et al., 2024). We study the latter setting. Tree-search methods, including Monte Carlo Tree Search (MCTS) (Browne et al., 2012), conduct systematic breadth exploration and depth refinement in the candidate solution space. By balancing exploration and exploitation under a finite compute budget, they have substantially improved model performance on complex tasks (Inoue et al., 2025).

However, adaptively controlling search width and depth does not remove the dependence of refinement-parent selection on generation lineage. In complete-solution search, standard MCTS selects nodes along a path from the root. Adaptive Branching Monte Carlo Tree Search (AB-MCTS) uses feedback at each node along the selection path to decide whether to generate a new child or descend into an existing branch. As illustrated in Figure 1, reaching the complete solution x_7 requires selecting the branch leading to it at every ancestor. With stochastic local decisions, the reach probability is the product of the conditional selection probabilities along this path. Under the illustrative assumption of a 0.6 probability at each level, it decreases exponentially with depth, falling to 21.6% at x_7. Yet such a complete solution can be revised directly using the task description, the candidate itself, and its feedback. Generation lineage therefore imposes an access constraint that direct refinement does not require.

The same tree thus serves two roles: sharing feedback through parent–child relationships and following these relationships to select the next refinement target. These relationships help search use

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

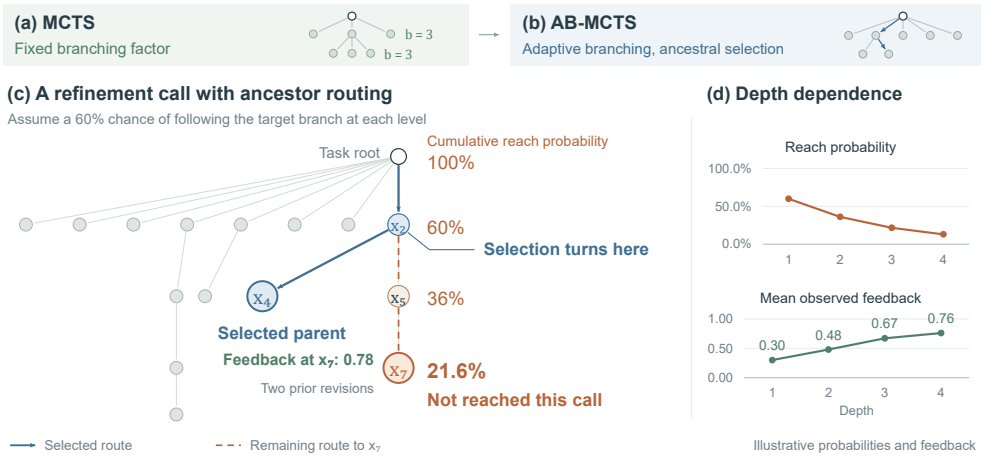


Figure 1: Ancestor routing in complete-solution tree search. (a,b) Both MCTS with fixed branching and AB-MCTS with adaptive branching retain selection along ancestral paths. (c) With conditional probability 0.6 per level, reaching x_7 has probability 21.6%; this call instead selects x_4. (d) Under this assumption, reach probability decays exponentially despite increasing feedback. Probabilities and feedback are illustrative.

the outcomes of related refinements, but a complete solution can serve directly as a starting point for refinement. Sharing feedback therefore does not require selecting refinement targets along ancestral paths. The core challenge is to decouple these roles while preserving the ability of feedback to guide search. When candidates from different branches and depths compete directly for budget, search must compare their value estimates and uncertainty while continually updating these estimates using new feedback from related candidates. We therefore need to combine global candidate selection with lineage-based feedback updates, allowing lineage to continue providing statistical information without constraining the paths through which refinement budgets reach candidates.

To address this challenge, we propose *Decoupled Topology Search* (DTS), which separates global refinement-parent selection from lineage-based feedback sharing. For each inference call, a global Bayesian controller uses observed feedback to choose between fresh generation (GEN) and refinement (CONT). When it chooses CONT, all valid complete candidates compete as refinement parents, regardless of their depth or branch. Candidate-level Beta distributions represent value estimates and uncertainty; a global acquisition rule uses them to select a parent directly. Expected improvement and a Pareto-based rule give rise to DTS-EI and DTS-Pareto, respectively. Meanwhile, parent–child links support weighted forward statistical inheritance and backward credit propagation. This allows feedback from related candidates to update the value estimates used for global selection (Section 3).

We evaluate DTS under matched candidate-call budgets, primarily on code generation, with additional studies in mathematical reasoning and molecular optimization. At $B = 128$, DTS outperforms repeated sampling and the evaluated tree-search baselines in final-answer accuracy (Pass@1) across the eight Qwen3/AReaL code settings. DTS also achieves a higher conversion rate from covered tasks to correct final outputs. We also observe a reversal in method rankings: in some settings, simple repeated sampling outperforms all evaluated feedback-guided search methods. We use low-budget offline probes on labeled instances to assess whether refinement may merit further calls. Diagnostics estimated from the first 16 candidates per pool or trajectory yield rankings across settings closely matching those obtained with 128 candidates (Spearman's $\rho \geq 0.927$; Section 5.4).

2 Related Work

Test-time scaling spans reasoning within responses and allocation across attempts (Zhang et al., 2025). Chain-of-thought prompting and budget forcing concern within-response reasoning (Wei et al., 2022; Muennighoff et al., 2025); self-consistency and repeated sampling exploit multiple

responses (Wang et al., 2022; Brown et al., 2024). Self-Refine and Reflexion use feedback to revise previous attempts (Madaan et al., 2023; Shinn et al., 2023). Snell et al. (2025) study difficulty-dependent compute allocation.

Feedback has been used to guide beam or tree search (Xie et al., 2023; Liu et al., 2023), supervise intermediate reasoning steps (Lightman et al., 2024; Wang et al., 2024), and support search-based training (Zhang et al., 2024a; Xie et al., 2024). These works apply feedback at different stages, from inference-time search to model training.

Structured reasoning methods organize intermediate thoughts or planning states (Yao et al., 2023; Besta et al., 2024; Yao et al., 2024; Hao et al., 2023), and tree search has also been applied to agent and code tasks (Zhou et al., 2023; Li et al., 2025; Antoniades et al., 2024). Bayesian methods guide program refinement (Tang et al., 2024), while FunSearch and Mind Evolution maintain evolving candidate populations (Romera-Paredes et al., 2024; Lee et al., 2025). Closer to complete-solution tree search, MCT Self-Refine and AB-MCTS revise complete candidates (Zhang et al., 2024b; Inoue et al., 2025). AB-MCTS uses feedback at each visited node to decide whether to generate a new child or enter an existing branch, and backs up scores through the tree (Inoue et al., 2025). Adaptive branching changes when the search widens or deepens, but reaching a particular complete candidate for refinement still depends on selections along its ancestral path.

3 Decoupled Topology Search

DTS separates two roles of the search tree: selecting a complete solution for refinement and sharing feedback across related solutions. It selects refinement parents from the eligible candidate pool, while retaining parent–child links to update candidate statistics. Figure 2 shows one search call: DTS chooses between fresh generation and refinement, selects a parent when refining, evaluates the new candidate, and updates its search statistics.

3.1 Candidates, Feedback, and Search Budget

For a task τ, each candidate x is a complete solution, such as a program, mathematical solution, or molecular proposal. We distinguish search-visible feedback $R_{\mathrm{obs}}(x)$ from the final evaluation objective $R_{\mathrm{eval}}(x)$. For code, these are public-test feedback and hidden-test correctness; they coincide under direct objective feedback.

After t calls, the ordered pool $\mathcal{P}_t = (x_1, \ldots, x_t)$ retains every generated entry, including repeated outputs. The eligible pool $\mathcal{A}_t \subseteq \mathcal{P}_t$ contains all candidates satisfying task-specific validity rules. GEN samples a fresh candidate from the task prompt; CONT revises a selected parent in $\mathcal{A}_t$ using available feedback, appending the child and retaining the parent. Each action consumes one call from the shared budget B.

At budget exhaustion, a selector using only search-visible information returns $\hat{x} = \mathcal{S}(\mathcal{P}_B, R_{\mathrm{obs}})$, evaluated by $R_{\mathrm{eval}}(\hat{x})$. Methods within each comparison share the final-selection rule. Section 4 and Appendix A specify the task protocols.

3.2 Adaptive Allocation between Generation and Refinement

Selecting refinement parents globally leaves a separate budget decision: should the next call generate a solution from the task prompt or revise an existing candidate? Neither action is always preferable, so DTS adapts their allocation using the observed scores of candidates produced by each action. The controller uses two-action Thompson sampling with a Beta state for each action $a \in \{\mathrm{GEN}, \mathrm{CONT}\}$, initialized to $(\alpha_a, \beta_a) = (\frac{1}{2}, \frac{1}{2})$. At call t, it chooses GEN if $\mathcal{A}_{t-1}$ is empty; otherwise, it independently samples both states:

$$\theta_a \sim \mathrm{Beta}(\alpha_a, \beta_a), \qquad a_t = \begin{cases} \mathrm{GEN}, & \theta_{\mathrm{GEN}} > \theta_{\mathrm{CONT}}, \\ \mathrm{CONT}, & \text{otherwise.} \end{cases} \tag{1}$$

The generated candidate x_t receives a normalized observed score $\tilde{r}(x_t) \in [0, 1]$, equal to $R_{\mathrm{obs}}(x_t)$ for binary feedback (score mappings: Appendix A). Only the chosen action is updated:

$$\alpha_{a_t} \leftarrow \alpha_{a_t} + \tilde{r}(x_t), \qquad \beta_{a_t} \leftarrow \beta_{a_t} + 1 - \tilde{r}(x_t). \tag{2}$$

A CONT update uses the child's score itself.

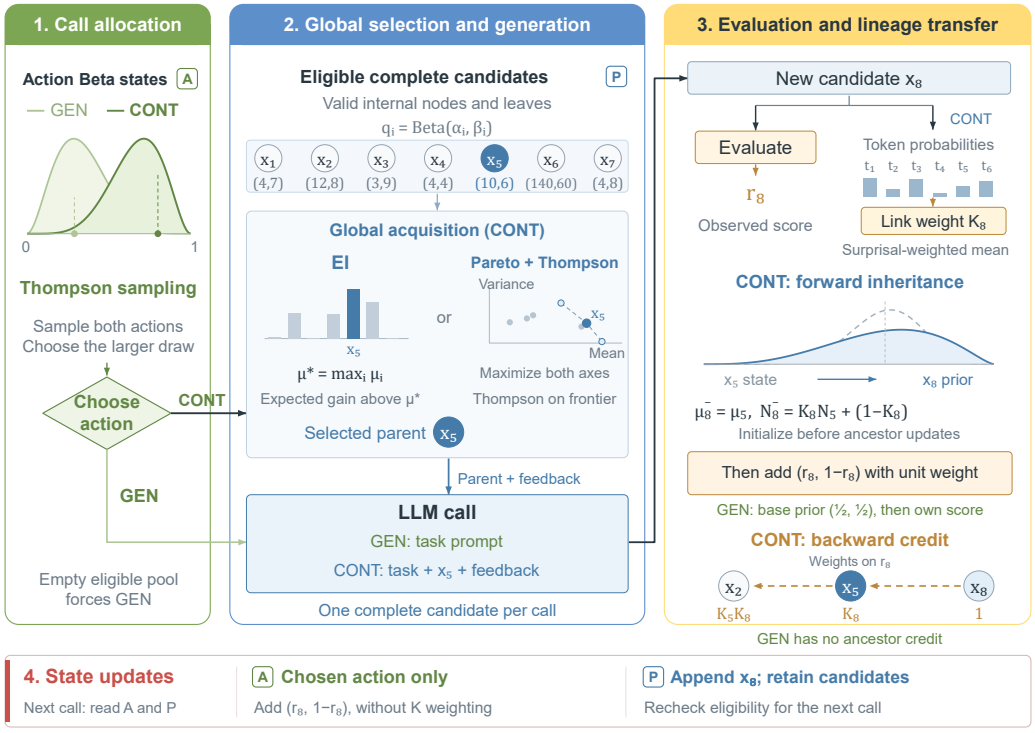


Figure 2: DTS method overview. (1) Thompson sampling chooses GEN or CONT. (2) For CONT, EI or Pareto–Thompson acquisition selects a parent globally from eligible complete candidates, including valid internal nodes and leaves; GEN samples from the task prompt.(3) Evaluation yields the observed score. For CONT, generated-token probabilities determine the link weight used for forward inheritance and backward credit; GEN uses a base prior. (4) The observed score updates the chosen action and new candidate states; for CONT, the link weight also controls updates along the lineage for the next call.

3.3 Global Selection of Refinement Parents

Once CONT is chosen, the search must decide which complete candidate to revise. DTS compares candidates across branches and depths using their feedback-updated score estimates and uncertainty. For CONT, acquisition considers all of $\mathcal{A}_{t-1}$, including valid candidates with existing children. Each candidate has a Beta state $q_x = \mathrm{Beta}(\alpha_x, \beta_x)$ with mean μ_x and variance σ_x^2 (Appendix B.2; Figure 4), updated through Section 3.4.

A single Thompson draw over the entire eligible pool provides a simple global selector, but it does not explicitly score exceedance over the current best estimate or screen candidates dominated in both mean and variance. DTS examines two acquisition rules for these choices: EI scores expected exceedance over the best current mean, while Pareto restricts sampling to the mean–variance frontier.

DTS-EI. With the common threshold $\mu^\star_{t-1} = \max_{x \in \mathcal{A}_{t-1}} \mu_x$, expected improvement scores candidates by

$$\mathrm{EI}(x) = \mathbb{E}_{\theta \sim q_x}\left[(\theta - \mu^\star_{t-1})_+\right], \qquad (z)_+ = \max(0, z). \tag{3}$$

We use $L = 1000$ independent draws per candidate and select $x_p = \arg\max_{x \in \mathcal{A}_{t-1}} \widehat{\mathrm{EI}}(x)$.

DTS-Pareto. The frontier Π_{t-1} is nondominated in posterior mean and variance (μ_x, σ_x^2), maximizing both coordinates (Appendix B.2). Larger variance provides an exploration signal. We draw

once per frontier candidate and select

$$\theta_x \sim q_x \quad (x \in \Pi_{t-1}), \qquad x_p = \arg\max_{x \in \Pi_{t-1}} \theta_x. \tag{4}$$

3.4 Lineage-Based Evidence Sharing

Each revision creates a new complete solution, so the child's score evaluates the child rather than its unchanged parent. Yet the child is generated from the parent and its feedback, making their observations potentially informative about one another. DTS transfers this evidence along lineage in both directions: forward inheritance initializes the child's Beta state using the parent's accumulated evidence before adding the child's own score, while weighted backward propagation lets that score inform ancestor states for future parent selection (Figure 5). A fresh GEN candidate starts from $(\frac{1}{2}, \frac{1}{2})$ and adds $(\tilde{r}(x), 1 - \tilde{r}(x))$. For a CONT child x_c of x_p, write $r_c = \tilde{r}(x_c)$ and apply the following updates.

Transfer weight. We use the model probabilities of generated refinement tokens to set a heuristic transfer weight for this indirect evidence.

For the generated refinement tokens with probabilities $P_1, \ldots, P_m$, the link weight $K_c = K(x_c, x_p) \in [0, 1]$ is

$$K_c = \frac{\sum_{j=1}^{m} P_j(-\log P_j)}{\sum_{j=1}^{m}(-\log P_j)}. \tag{5}$$

Here P_j is the model probability of the j-th generated token under the refinement prompt containing x_p and its feedback. We use the resulting K_c as a heuristic weight for transferring evidence along this parent–child link. This surprisal-weighted average of the child's own token probabilities regulates transfer strength in both directions.

Forward inheritance. Before updating ancestors, let $N_p = \alpha_p + \beta_p$ and $\mu_p = \alpha_p / N_p$. With baseline concentration $N_0 = 1$, the child inherits the parent mean at concentration N_c^- and adds its own score:

$$\begin{aligned} N_c^- &= K_c N_p + (1 - K_c) N_0, \\ (\alpha_c, \beta_c) &= (N_c^- \mu_p + r_c,\ N_c^-(1 - \mu_p) + 1 - r_c). \end{aligned} \tag{6}$$

Smaller K_c moves the inherited concentration toward N_0, increasing the relative weight of the child's observation.

Backward credit propagation. After updating the child's state, its score updates each visited ancestor x_a with weight $\lambda_{a,c}$, the product of link weights on the path from x_a to x_c:

$$\alpha_a \leftarrow \alpha_a + \lambda_{a,c} r_c, \qquad \beta_a \leftarrow \beta_a + \lambda_{a,c}(1 - r_c). \tag{7}$$

The direct parent receives weight K_c; propagation proceeds toward the lineage root. Algorithm 1 summarizes the search. Appendix B provides the full loop (Algorithm 2), subroutines, and computational costs.

Algorithm 1 DTS: global allocation with lineage-based evidence sharing

Require: Task τ, budget B, acquisition rule, final selector $\mathcal{S}$
1: Initialize empty pool $\mathcal{P}_0$ and both action states to $(\frac{1}{2}, \frac{1}{2})$
2: **for** $t = 1, \ldots, B$ **do**
3: Form valid pool $\mathcal{A}_{t-1}$; choose a_t by Section 3.2
4: Set $x_p = \perp$ for GEN; otherwise acquire x_p globally from $\mathcal{A}_{t-1}$
5: Generate x_t from τ or refine x_p; obtain observed score $r_t = \tilde{r}(x_t)$
6: Append x_t to $\mathcal{P}_{t-1}$ to form $\mathcal{P}_t$; record $p(x_t) = x_p$
7: Initialize its Beta state and add r_t, using Section 3.4
8: For CONT, propagate r_t to ancestors via Eq. equation 7
9: Update only the chosen action state via Eq. equation 2
10: **end for**
11: **return** $\mathcal{S}(\mathcal{P}_B, R_{\text{obs}})$

4 EXPERIMENTAL PROTOCOL

4.1 TASKS, MODELS, AND EVALUATION METRICS

Our primary evaluation uses subsets of 175 LiveCodeBench v6 (LCBv6) tasks (Jain et al., 2025) and 165 CodeContests (CCT) tasks (Li et al., 2022). On both benchmarks, we evaluate Qwen3-4B/8B/14B (Yang et al., 2025), Qwen2.5-7B-Instruct (Qwen et al., 2025), and AReaL-boba-2-14B (Fu et al., 2025). Search receives a binary reward for passing all public tests; hidden tests determine final correctness.

Supplementary studies use Qwen3-8B on 60 AIME 2024/2025 problems, with exact-correctness feedback (AIME-Base) or ReasonFlux-PRM feedback (AIME-RM) (Zou et al., 2025). Molecular optimization uses gpt-4o-mini on the five-objective PMO setting (Gao et al., 2022; Ran et al., 2025), with the objective itself providing feedback and final evaluation. Task protocols appear in Appendix A.

For code and AIME, we report final-answer accuracy (Pass@1) and oracle coverage: the fractions of tasks with a correct returned answer and with at least one correct generated candidate, respectively. For code, conversion $\mathrm{Conv} = \mathrm{Pass@1}/\mathrm{Coverage}$ measures the fraction of covered tasks yielding a correct final answer when coverage is nonzero; aggregation details appear in Appendix D. For PMO, we report the returned objective value, `top1_f`.

4.2 BASELINES AND CONTROLLED COMPARISONS

We compare DTS-EI and DTS-Pareto with repeated sampling (RS), sequential refinement (S), standard MCTS, and AB-MCTS-A/M (Inoue et al., 2025). RS generates independently; S follows one revision chain. Standard MCTS uses UCT with coefficient $\sqrt{2}$ and five children per expansion, truncating the last expansion to respect the budget. AB-MCTS uses its default configurations.

Within each setting, methods share tasks, model, prompts, feedback, decoding (temperature 0.6), budget, and final-selection rule. Each generation or revision counts as one candidate call, including invalid outputs. Budgets are $B = 128$ for code and PMO and $B = 64$ for AIME; results report means and standard deviations over three runs.

The default final selector returns the latest candidate with the highest observed feedback. We also keep each method's generated pools fixed and evaluate earliest or uniform selection among feedback maximizers (Appendix D).

Topology controls compare DTS and coupled-DTS on LCBv6 / Qwen3-14B and CCT / Qwen3-8B at $B = 128$, using both EI and Pareto. Coupled-DTS replaces global parent selection with tree traversal, retaining the controller, acquisition rule, and statistical updates (Appendix B.4). Both methods use the same final-selection rule in each comparison.

5 RESULTS AND ANALYSIS

We evaluate overall performance, analyze candidate coverage and final selection, test selection decoupling and component contributions, and examine feedback errors and local recovery with low-budget offline probes.

5.1 PERFORMANCE ACROSS MODELS AND SEARCH BUDGETS

At $B = 128$, DTS achieves the highest mean final-answer accuracy across all eight Qwen3/AReaL code settings (Table 1). On LCBv6 / Qwen3-14B, DTS-EI and DTS-Pareto improve accuracy over RS by 6.2 and 7.2 percentage points, respectively, and over the strongest non-DTS baseline, sequential refinement, by 2.8 and 3.8 points. RS ranks first in both Qwen2.5-7B settings; we examine this reversal in Section 5.4.

Figure 3 shows performance as the candidate-call budget increases. On LCBv6, DTS continues to improve at larger budgets, while RS shows smaller and less consistent gains. Appendix C.2 provides additional curves for AReaL and Qwen2.5.

Table 1: Final-budget results (three-run means; best in bold). Code and AIME report Pass@1 (%); PMO reports `top1_f`. DTS subscripts show differences from RS: percentage points for code and AIME, objective units for PMO. Standard deviations and additional metrics are in Appendices C.1 and C.4.

Model / Task	RS	S	MCTS	AB-MCTS		DTS	
				A	M	EI	Pareto
LiveCodeBench v6: Pass@1 (%), $B = 128$							
Qwen3-4B	55.7	60.2	58.7	61.1	57.7	$\mathbf{61.3}_{+5.6}$	$59.8_{+4.1}$
Qwen3-8B	55.6	60.8	60.2	58.5	59.8	$61.1_{+5.5}$	$\mathbf{63.0}_{+7.4}$
Qwen3-14B	64.4	67.8	65.7	65.1	64.2	$70.6_{+6.2}$	$\mathbf{71.6}_{+7.2}$
AReaL-14B	65.3	67.8	68.0	66.1	68.2	$70.3_{+5.0}$	$\mathbf{71.6}_{+6.3}$
Qwen2.5-7B	**29.7**	21.7	26.7	27.2	27.7	$28.8_{-0.9}$	$27.6_{-2.1}$
CodeContests: Pass@1 (%), $B = 128$							
Qwen3-4B	49.5	48.9	50.4	51.1	48.9	$\mathbf{52.3}_{+2.8}$	$49.4_{-0.1}$
Qwen3-8B	51.7	52.3	51.4	52.7	53.2	$54.3_{+2.6}$	$\mathbf{54.5}_{+2.8}$
Qwen3-14B	54.2	55.0	57.8	56.6	55.7	$57.8_{+3.6}$	$\mathbf{59.8}_{+5.6}$
AReaL-14B	59.2	60.6	60.8	60.2	59.4	$\mathbf{63.8}_{+4.6}$	$63.2_{+4.0}$
Qwen2.5-7B	**15.8**	5.2	10.3	14.9	14.3	$14.9_{-0.9}$	$15.4_{-0.4}$
AIME: Qwen3-8B, Pass@1 (%), $B = 64$							
AIME-Base	78.3	73.9	76.1	77.2	76.1	$\mathbf{78.9}_{+0.6}$	$75.6_{-2.7}$
AIME-RM	60.0	61.1	60.6	56.4	58.3	$\mathbf{63.9}_{+3.9}$	$60.6_{+0.6}$
PMO: gpt-4o-mini, objective score, $B = 128$							
PMO	3.7775	3.6566	3.7495	3.7723	3.7279	$\mathbf{3.8051}_{+0.0276}$	$3.7182_{-0.0593}$

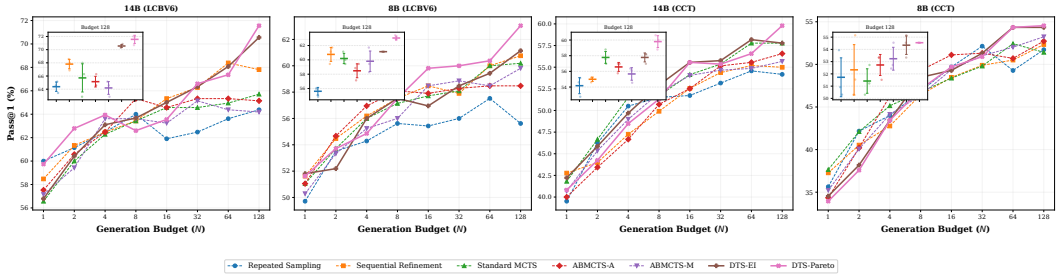


Figure 3: Final-answer accuracy versus candidate-call budget B (labeled N) for Qwen3-14B and Qwen3-8B on LCBv6 and CCT. Curves show three-run means under latest-maximizer selection; insets show variation at $B = 128$. ABMCTS-A/M denote AB-MCTS-A/M.

Beyond code generation, supplementary experiments provide further evidence of DTS's effectiveness on mathematical reasoning and molecular optimization (Table 1). DTS-EI improves accuracy over RS by 0.6 percentage points on AIME-Base and 3.9 points on AIME-RM. On PMO, DTS-EI improves the objective value from 3.7775 to 3.8051. Full results for both variants appear in Appendix C.4.

5.2 Candidate Coverage and Final-Answer Conversion

DTS converts a larger fraction of covered tasks into correct final outputs. On LCBv6 / Qwen3-14B, DTS-EI and DTS-Pareto reduce oracle coverage relative to RS by 2.9 and 3.2 percentage points, respectively, while increasing final-answer accuracy by 6.2 and 7.2 points (Table 2). The corresponding conversion rates rise from 80.3% for RS to 91.3% and 93.0%. A similar pattern appears on CCT / Qwen3-14B: DTS-Pareto raises accuracy from 54.2% to 59.8%, with coverage decreasing from 69.4% to 68.1% (Appendix C.1).

Table 2: Coverage and final selection at $B = 128$. (a) LCBv6 / Qwen3-14B: three-run means (%); conversion is their ratio. (b) Gains over RS (pp), averaged equally across ten code settings; parentheses count settings above RS. Both methods use the same final rule, with each method's pools fixed across rules.

(a) Coverage and conversion				(b) Final-selection rules		
Method	Coverage	Pass@1	Conv.	Rule	DTS-EI	DTS-Pareto
RS	80.2	64.4	80.3	Latest	$+3.41$ $(8/10)$	$+3.48$ $(7/10)$
DTS-EI	77.3	70.6	91.3	Uniform	$+2.53$ $(8/10)$	$+2.17$ $(9/10)$
DTS-Pareto	77.0	71.6	93.0	Earliest	$+0.41$ $(6/10)$	$+0.39$ $(5/10)$

Table 3: Effect of parent-selection topology at $B = 128$. Pass@1 (%) uses the default latest-maximizer rule. Values are three-run means $\pm$ standard deviations; Δ is the DTS-minus-coupled difference in percentage points, rounded from the reported means.

Setting	Acquisition	Coupled-DTS	DTS	Δ (pp)
LCBv6 / Qwen3-14B	EI	63.81 ± 1.35	$\mathbf{70.6} \pm 0.2$	$+6.8$
	Pareto	65.33 ± 0.97	$\mathbf{71.6} \pm 0.6$	$+6.3$
CCT / Qwen3-8B	EI	51.52 ± 0.49	$\mathbf{54.3} \pm 0.8$	$+2.8$
	Pareto	53.13 ± 0.76	$\mathbf{54.5} \pm 0.0$	$+1.4$

We further evaluate alternative final-selection rules while keeping each method's generated candidate pools fixed (Table 2b). Under uniform selection among candidates with the highest observed feedback, both DTS variants outperform RS in all eight Qwen3/AReaL settings, with mean gains of 3.28 and 2.79 percentage points for EI and Pareto, respectively (Table 13). Across all ten settings, earliest-maximizer selection yields smaller mean gains of approximately 0.4 points for both variants (Appendix D). Together, the higher conversion rates and gains under uniform selection show that DTS produces candidate pools that support more accurate final answers.

5.3 Selection Decoupling and Component Ablations

Matched comparisons with coupled-DTS show the benefit of global refinement-parent selection while retaining the same controller and lineage updates (Table 3). At $B = 128$, decoupling improves mean final-answer accuracy by approximately 6.8 and 6.3 percentage points for EI and Pareto on LCBv6 / Qwen3-14B, and 2.8 and 1.4 points on CCT / Qwen3-8B. Under uniform final selection, the mean gains remain positive in all four comparisons (Appendix D.4).

Lineage sharing remains important after parent selection is decoupled. On LCBv6 / Qwen3-14B at $B = 128$, the three lineage-update interventions reduce DTS-Pareto accuracy by 5.1–6.6 percentage points (Table 4). These results support retaining parent evidence and controlling its transfer strength during global search.

Adaptive action allocation and parent acquisition also improve results. A fixed 50/50 generation/refinement allocation reduces accuracy by 7.0 percentage points. Replacing Pareto acquisition with random parent selection or global Thompson sampling reduces accuracy by 4.9 or 4.6 points, respectively (Table 4).

5.4 Search Applicability and Low-Budget Diagnostics

Method rankings reverse in the two Qwen2.5-7B code settings: at $B = 128$, RS achieves higher mean final-answer accuracy than all evaluated feedback-guided search methods (Table 1). This reversal shows that the benefits of feedback-guided refinement depend on the model–task setting. We use low-budget offline probes to inform whether refinement merits further calls in a given setting.

The probes measure X_{trap}, the error rate among top-feedback RS candidates, and Y_{local}, the fraction of initially incorrect solutions recovered through sequential refinement. We estimate both quantities on labeled instances, using the first 16 candidates per sampling pool or refinement trajectory. At

Table 4: Ablations on LCBv6 / Qwen3-14B, $B = 128$. Pass@1: three-run mean $\pm$ SD (%); Δ: change from full DTS-Pareto (pp). Definitions and full results: Appendices B.6 and C.3.

Full DTS-Pareto: $\mathbf{71.6} \pm 0.6$					
Lineage updates			Allocation and parent selection		
Variant	Pass@1	Δ	Variant	Pass@1	Δ
No inheritance	65.5 ± 2.3	-6.1	Fixed 50/50	64.6 ± 2.2	-7.0
Static $K = 1$	66.5 ± 1.2	-5.1	Random parent	66.7 ± 1.2	-4.9
No K-aware transfer	65.0 ± 1.0	-6.6	Global TS	67.0 ± 1.5	-4.6

this budget, recovery rates are only 2.49–4.25% in the two Qwen2.5 settings, compared with 20.21–31.07% in the other eight settings. The Qwen2.5 settings also exhibit higher feedback-error rates. Thus, the short probes identify low-recovery settings that coincide with those where RS outperforms feedback-guided search at $B = 128$.

Across the ten code settings, diagnostic rankings estimated from 16 candidates closely match those obtained with 128 candidates: Spearman's ρ is 0.976 for X_{trap} and 0.927 for Y_{local} (Appendix E). These results show that short probes preserve the cross-setting ordering of feedback errors and local recovery, providing early evidence relevant to allocating further computation to refinement. DTS itself uses only search-visible feedback and does not access these offline diagnostic quantities.

6 Conclusion

We introduced Decoupled Topology Search (DTS), which allows complete candidates to compete globally for refinement while retaining their generation lineage for feedback sharing. At $B = 128$, DTS achieves higher final-answer accuracy across the eight Qwen3/AReaL code settings and converts a larger fraction of covered tasks into correct final outputs. Matched controls support the benefit of decoupling, and fixed-pool tests show that gains persist under uniform final selection. We also identify settings where repeated sampling outperforms feedback-guided search. Sixteen-candidate offline probes on labeled instances characterize feedback error and local recovery, providing early evidence for assessing whether refinement merits further computation.

AI Use Statement

We used ChatGPT to assist with literature review, drafting and revising parts of the manuscript, language editing, manuscript organization, and table formatting. We used AI coding tools to write portions of the research and analysis code and to assist with the analysis of results. All AI-assisted material was manually reviewed by the authors, who take responsibility for the final content of the paper.

Ethics Statement

This work studies inference-time search using existing models and benchmarks for code generation, mathematical reasoning, and molecular optimization. The findings concern the evaluated task and feedback settings. Benchmark scores alone do not establish the correctness or safety of outputs in downstream applications; in particular, molecular objective scores do not establish biological efficacy or safety. Practical use requires application-specific validation.

Reproducibility Statement

Section 4 specifies the benchmarks, models, baselines, candidate-call budgets, evaluation metrics, and final-selection rules. Appendix A describes task-specific feedback and score mappings. Appendix B provides the full search algorithm, acquisition and lineage-update rules, the coupled-DTS control, and ablation definitions. Appendices C–F provide complete result tables, final-selection analyses, diagnostic estimators, and behavioral metrics supporting the reported comparisons. We will make our code repository publicly available after removing sensitive information.

REFERENCES

Antonis Antoniades, Albert Örwall, Kexun Zhang, Yuxi Xie, Anirudh Goyal, and William Wang. Swe-search: Enhancing software agents with monte carlo tree search and iterative refinement. *arXiv preprint arXiv:2410.20285*, 2024.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35: 21342–21357, 2022.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, 2023.

Yuichi Inoue, Kou Misaki, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. Wider or deeper? scaling llm inference-time compute with adaptive branching tree search. *arXiv preprint arXiv:2503.04412*, 2025.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*, 2025.

Jierui Li, Hung Le, Yingbo Zhou, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Codetree: Agent-guided tree search for code generation with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3711–3726, 2025.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. *arXiv preprint arXiv:2309.15028*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Nian Ran, Yue Wang, Xiaoyuan Zhang, Zhongzheng Li, Qingsong Ran, Wenhao Li, and Richard Allmendinger. Exllm: Experience-enhanced llm optimization for molecular design and beyond. *arXiv preprint arXiv:2502.12845*, 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.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Hao Tang, Keya Hu, Jin P Zhou, Sicheng Zhong, Wei-Long Zheng, Xujie Si, and Kevin Ellis. Code repair with llms gives an exploration-exploitation tradeoff. *Advances in Neural Information Processing Systems*, 37:117954–117996, 2024.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36:41618–41650, 2023.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 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.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Yao Yao, Zuchao Li, and Hai Zhao. Got: Effective graph-of-thought reasoning in language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2901–2921, 2024.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024a.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394*, 2024b.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.

Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. Reasonflux-prm: Trajectory-aware prms for long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2506.18896*, 2025.

A Task Protocols and Feedback

A.1 Code Generation

We evaluate complete-program generation, a standard setting for LLM program synthesis (Austin et al., 2021; Li et al., 2022). Each candidate is a complete Python program. Fresh generation conditions on the task; refinement additionally receives the selected program and its public-test feedback. A revision is appended to the pool. Observable reward is binary: $R_{\text{obs}} = R_{\text{pub}} = 1$ exactly when all available public tests pass. Final evaluation uses the binary private-test reward $R_{\text{eval}} = R_{\text{priv}}$. Private tests are used only for evaluation and offline diagnostics. Unless specified otherwise, the returned candidate is the latest maximizer of public-test reward.

A.2 Mathematical Reasoning

We use Qwen3-8B on the 60 AIME 2024/2025 problems with $B = 64$. Each candidate is a complete solution with an extracted final answer. Exact answer correctness defines R_{eval}. AIME-Base exposes this same binary score to search, providing an oracle-feedback control. AIME-RM instead uses ReasonFlux-PRM (Zou et al., 2025). Explicit "Step k:" blocks are extracted when present; otherwise the response body forms one step. An identifiable final boxed-answer segment is separated and appended as the final step. Steps are joined with `<extra_0>`. Given step rewards $r_1, \ldots, r_m$, the search and final-selection score is

$$R_{\text{obs}}(x) = \text{clip}_{[0,1]}\left(0.5\frac{1}{m}\sum_{j=1}^{m} r_j + 0.3\min_j r_j + 0.2r_m\right). \tag{8}$$

Step rewards are also included in refinement feedback. Final AIME-RM accuracy is evaluated against the exact answer, independently of the PRM.

A.3 Molecular Optimization and Budget Accounting

PMO uses gpt-4o-mini, $B = 128$, and the five-objective setting of Ran et al. (2025). Candidates are SMILES strings; invalid molecules receive zero reward. The objective itself supplies feedback and evaluation, $R_{\text{obs}}(x) = R_{\text{eval}}(x) = f(x)$. For DTS score updates it is normalized as $\tilde{r}(x) = \min\{\max\{f(x), 0\}, 5\}/5$. Results retain the original objective units.

Every fresh generation or revision consumes one candidate call, including invalid outputs. Invalid candidates are excluded from future parent selection. Comparisons within a setting share decoding, feedback, and candidate-call budgets. All reported runs use temperature 0.6; means and standard deviations summarize three independent runs. Candidate-call matching does not equate token counts or wall-clock cost.

B Full Algorithm Specification

Algorithm 2 gives the complete DTS search loop. The following subsections specify its state, acquisition rules, lineage updates, and coupled-DTS parent selector. The global and coupled selectors share the controller, candidate-state representation, and lineage updates.

B.1 State and Update Order

Each candidate record stores its complete output, observed score, validity flag, Beta parameters, and parent pointer. A refinement child also stores the weight of its incoming edge. The ordered pool retains one record per generation event, including repeated outputs and invalid candidates. Invalid candidates receive zero observed reward under the task protocols and are excluded from refinement-parent selection. The implementation treats a missing validity flag as valid.

The implementation attaches all GEN candidates to a virtual root, which stores no complete solution and is never a refinement parent. We write $p(x) = \perp$ for these candidates, omitting the virtual root from candidate-level updates. The two action states are separate from this lineage structure and both start at $(\alpha_a, \beta_a) = (\frac{1}{2}, \frac{1}{2})$.

Algorithm 2 Decoupled Topology Search (DTS)

Require: Task τ, budget B, acquisition mode $m \in \{\mathrm{EI}, \mathrm{Pareto}\}$, samplers $G_{\mathrm{GEN}}, G_{\mathrm{CONT}}$, feedback R_{obs}, final selector $\mathcal{S}$

Initialize empty ordered pool $\mathcal{P}_0$ and lineage links
$(\alpha_a, \beta_a) \leftarrow (\frac{1}{2}, \frac{1}{2})$ for $a \in \{\mathrm{GEN}, \mathrm{CONT}\}$
for $t = 1, \ldots, B$ **do**
  $\mathcal{A}_{t-1} \leftarrow$ valid refinement candidates in $\mathcal{P}_{t-1}$
  $a_t \leftarrow$ GEN if $\mathcal{A}_{t-1} = \emptyset$; otherwise sample by Eq. equation 1
  if $a_t = \mathrm{GEN}$ **then**
    $x_t \sim G_{\mathrm{GEN}}(\tau)$; $p(x_t) \leftarrow \perp$
  else
    $x_p \leftarrow$ GLOBALPARENT$(\mathcal{A}_{t-1}, m)$ ▷ Section 3.3
    $x_t \sim G_{\mathrm{CONT}}(\tau, x_p, R_{\mathrm{obs}}(x_p))$; $p(x_t) \leftarrow x_p$
  end if
  Evaluate $R_{\mathrm{obs}}(x_t)$ and its normalized score $\tilde{r}(x_t)$
  Append x_t to $\mathcal{P}_{t-1}$ to obtain $\mathcal{P}_t$
  if $a_t = \mathrm{GEN}$ **then**
    $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\frac{1}{2} + \tilde{r}(x_t), \frac{1}{2} + 1 - \tilde{r}(x_t))$
  else
    LINEAGEUPDATE$(x_p, x_t, \tilde{r}(x_t))$ ▷ Algorithm 3
  end if
  Update only action a_t with $\tilde{r}(x_t)$ by Eq. equation 2
end for
return $\mathcal{S}(\mathcal{P}_B, R_{\mathrm{obs}})$

For CONT, parent selection uses the statistics available before the new call. After generating and evaluating the child, DTS initializes its state from the parent's current state, adds the child's own score with unit weight, propagates that score to ancestors, and finally updates the CONT action state. The child is initialized before its feedback changes any ancestor. Both success and failure feedback enter these updates through $(\tilde{r}, 1 - \tilde{r})$. The final selector operates on the complete ordered pool using observed feedback; the selector for each experiment is specified with its evaluation protocol.

B.2 Global Acquisition Rules

For $q_x = \mathrm{Beta}(\alpha_x, \beta_x)$, the moments used by both acquisition rules are

$$\mu_x = \frac{\alpha_x}{\alpha_x + \beta_x}, \qquad \sigma_x^2 = \frac{\alpha_x \beta_x}{(\alpha_x + \beta_x)^2(\alpha_x + \beta_x + 1)}. \tag{9}$$

For a nonempty eligible pool $\mathcal{A}$, GLOBALPARENT uses the current Beta states without changing them during selection. For EI, let $\mu^\star = \max_{x \in \mathcal{A}} \mu_x$. The implemented estimate is

$$\widehat{\mathrm{EI}}(x) = \frac{1}{L} \sum_{\ell=1}^{L} \max\{0, \theta_x^{(\ell)} - \mu^\star\}, \quad \theta_x^{(\ell)} \sim \mathrm{Beta}(\alpha_x, \beta_x), \quad L = 1000. \tag{10}$$

The vectorized computation uses an $L \times |\mathcal{A}|$ sample matrix and averages over its sample dimension, yielding one score per candidate. The selected parent maximizes this estimate over $\mathcal{A}$.

For Pareto selection, a candidate $x' \in S$ dominates $x \in S$ if $\mu_{x'} \geq \mu_x$ and $\sigma_{x'}^2 \geq \sigma_x^2$, with at least one strict inequality. For any nonempty candidate pool S, $\Pi(S)$ contains all candidates not dominated by another candidate in S. Candidates with identical mean–variance pairs do not dominate one another. DTS constructs $\Pi(\mathcal{A})$, draws one Beta value per frontier candidate, and returns the largest draw. Acquisition ties follow the fixed candidate ordering. These draws are internal numerical operations and do not consume additional candidate calls.

B.3 Lineage Updates and Boundary Cases

Algorithm 3 expands the update in Section 3.4. The weight K_c is computed from the generated child's token probabilities using Eq. equation 5. The implementation floors its denominator at a

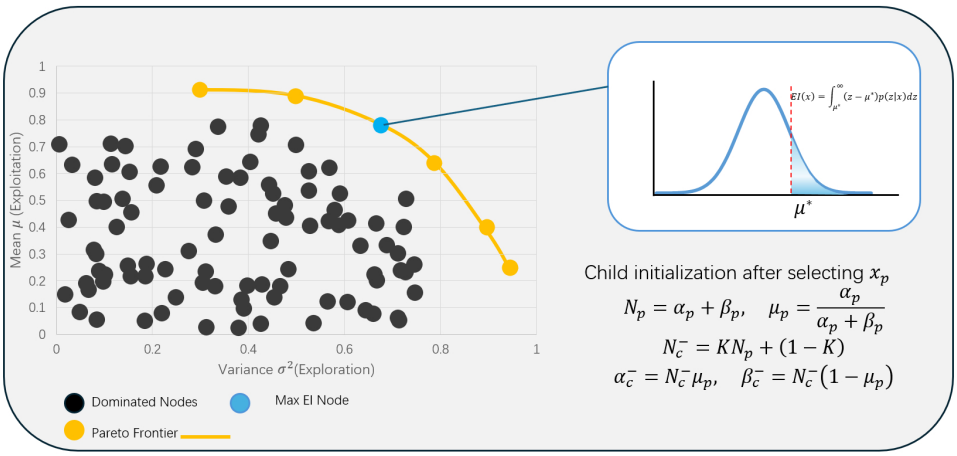


Figure 4: Schematic illustration of EI and Pareto acquisition in the global candidate pool. EI compares each candidate against the largest current Beta mean. Pareto selection retains nondominated mean–variance candidates and then applies Thompson sampling on the frontier. The right-hand formulas show the child's initialization before its own score is added.

small positive constant ε_{num} to avoid division by zero. DTS and coupled-DTS use the same computation. The generation record supplies this weight as `transition_K`; the search procedure reads it and uses $K_c = 0$ when it is missing. At $K_c = 0$, the inherited concentration is still $N_0 = 1$ and its mean remains μ_p; backward updates have zero weight. At $K_c = 1$, the child inherits the parent's full concentration.

Algorithm 3 LINEAGEUPDATE: inheritance and backward credit

Require: Parent x_p, child x_c with $p(x_c) = x_p$, normalized score $r = \tilde{r}(x_c)$
$K_c \leftarrow$ stored weight for $x_p \to x_c$, or 0 if missing
$N_p \leftarrow \alpha_p + \beta_p$; $\mu_p \leftarrow \alpha_p / N_p$; $N_c^- \leftarrow K_c N_p + (1 - K_c)$
$(\alpha_c, \beta_c) \leftarrow (N_c^- \mu_p + r,\ N_c^-(1 - \mu_p) + 1 - r)$
$z \leftarrow x_p$; $\lambda \leftarrow K_c$
while $z \neq \perp$ **do**
  $\alpha_z \leftarrow \alpha_z + \lambda r$; $\beta_z \leftarrow \beta_z + \lambda(1 - r)$
  if $\lambda < 10^{-4}$ or $p(z) = \perp$ **then**
    break
  end if
  $k_z \leftarrow$ stored weight for $p(z) \to z$, or 0 if missing
  $\lambda \leftarrow \lambda k_z$; $z \leftarrow p(z)$
end while

The cutoff is checked *after* updating the current ancestor. Thus, the first visited ancestor receiving weight below 10^{-4} still receives that weighted update, after which propagation stops. The virtual root receives no candidate-level credit; the action controller is updated separately in Algorithm 2.

B.4 COUPLED-DTS PARENT SELECTION

We evaluate this control on LCBv6 / Qwen3-14B and CCT / Qwen3-8B at $B = 128$, using both EI and Pareto. Appendix D.4 reports coverage and accuracy under three final-selection rules.

The coupled control replaces GLOBALPARENT in Algorithm 2 with Algorithm 4. All generation, evaluation, controller, and lineage updates are unchanged. Selection starts among valid children of the virtual root, each corresponding to a GEN candidate. At a visited candidate v, the local pool consists of v followed by its valid immediate children in their stored order. Selecting v stops traversal and refines it; selecting a child moves to that child and repeats the comparison. A valid internal node can therefore be the refinement parent, and a leaf stops automatically.

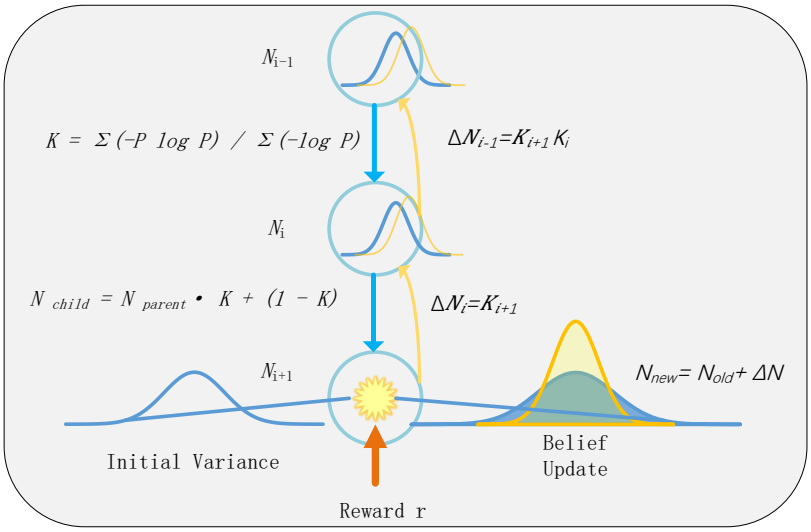


Figure 5: Schematic illustration of lineage-based evidence sharing. Blue arrows indicate forward inheritance; yellow arrows indicate backward propagation of the child's observed score. The inherited concentration is set before adding the child's own score; ancestor update weights multiply along the lineage. These links share evidence without constraining future parent selection.

For EI, scores are computed once for the full eligible pool, using its global maximum mean and $L = 1000$ draws per candidate. Every local comparison reuses these scores. For Pareto, one Thompson value is drawn for each eligible candidate before traversal. At each local pool S, the same dominance operator constructs $\Pi(S)$, and the candidate with the largest cached draw on that local frontier is selected. The frontier is local to the comparison; the draws are fixed throughout the traversal. Descending deeper introduces no additional acquisition draws.

Algorithm 4 Coupled-DTS: tree traversal with cached acquisition values

Require: Eligible candidates $\mathcal{A}$, virtual root v_0, acquisition mode m
if $m = \text{EI}$ **then**
  Cache $s_x \leftarrow \widehat{\text{EI}}(x)$ for all $x \in \mathcal{A}$ using Eq. equation 10
else
  Cache $s_x \sim \text{Beta}(\alpha_x, \beta_x)$ for all $x \in \mathcal{A}$
end if
$v \leftarrow \textsc{LocalSelect}(\text{ValidChildren}(v_0), m, s)$
loop
  $S \leftarrow [v] + \text{ValidChildren}(v)$
  $u \leftarrow \textsc{LocalSelect}(S, m, s)$
  if $u = v$ **then**
    return v
  end if
  $v \leftarrow u$
end loop
function $\textsc{LocalSelect}(S, m, s)$
  $F \leftarrow S$ if $m = \text{EI}$, otherwise $F \leftarrow \Pi(S)$
  return the first maximizer of s_x in F
end function

The implementation's maximum operation returns the first maximizer in the supplied comparison sequence. In the EI local pool this gives the current node priority over equally scoring children. In the Pareto case, tie order follows the frontier sequence returned by the frontier routine. An empty local pool raises an error. The virtual root is never itself included in a comparison pool.

B.5 Computation and Storage

Let $A_t = |\mathcal{A}_{t-1}|$ at call t, m_t be the number of refinement tokens used for K, and D_t the number of ancestors visited during credit propagation. Rebuilding the eligible pool costs $O(t)$ per call when validity flags are stored, or $O(B^2)$ over the run. The action controller and forward inheritance each take $O(1)$ work per call. Computing K and propagating credit take $O(m_t + D_t)$ per refinement call. Candidate statistics, parent pointers, and edge weights require $O(B)$ persistent storage, in addition to the candidate outputs and feedback records.

EI costs $O(LA_t)$ per refinement decision and $O(LB^2)$ in the worst case over the run, with $L = 1000$. Its vectorized sample matrix uses $O(LA_t)$ temporary storage. For Pareto selection, direct pairwise dominance checks cost $O(A_t^2)$; a two-dimensional sorting-and-scan construction can compute the same frontier in $O(A_t \log A_t)$, retaining tied non-dominated entries. Sampling on the resulting frontier costs $O(|\Pi(\mathcal{A}_{t-1})|)$.

The coupled selector first computes or samples cached values for the full eligible pool, then compares them along one descending path. EI adds the scans of the visited local pools. Pareto reconstructs a frontier for each visited local pool using cached draws. These costs are numerical bookkeeping; generating candidates and obtaining their task feedback are additional costs. Matching B therefore fixes the number of candidate calls, while token and wall-clock costs depend on the realized outputs and evaluations.

B.6 Ablation Definitions

The ablation runner exposes separate action-allocation, parent-selection, and credit modes. Unless stated otherwise, the reference configuration is adaptive allocation, Pareto selection, and dynamic credit. Every variant still incorporates the child's own score with unit weight and updates the selected action state with that score. An empty eligible pool forces GEN.

Allocation and acquisition. Fixed 50/50 replaces the controller's decision with an independent fair coin flip between GEN and CONT; it does not enforce exactly equal realized call counts. Random parent samples uniformly from the full eligible pool. Global TS draws once from every eligible candidate's Beta state and chooses the largest draw, without Pareto filtering. These acquisition variants retain adaptive allocation and dynamic lineage updates.

Credit modes. Let κ_f be the forward coefficient and λ_0 the initial backward weight assigned to the direct parent. The ablation implementation initializes the child's pre-observation state as

$$(\alpha_c^-, \beta_c^-) = \begin{cases} (\frac{1}{2}, \frac{1}{2}), & \kappa_f = 0, \\ \big(N^- \mu_p, N^-(1-\mu_p)\big), & \kappa_f > 0, \end{cases} \qquad N^- = \kappa_f N_p + (1 - \kappa_f). \tag{11}$$

Table 5: Credit-mode settings in the ablation implementation. The historical-edge multiplier is applied when moving to the next ancestor.

Mode	κ_f	λ_0	Historical-edge multiplier
Dynamic	K_c	K_c	Stored K
No K-aware transfer	0	1	1
Static $K = 1$	1	1	1
No inheritance	0	K_c	Stored K

The code mode `none` corresponds to "No K-aware transfer" in Table 5: backward propagation remains active, with weight one at every non-root ancestor. Static $K = 1$ inherits the parent's full pre-update Beta state. No inheritance is evaluated with both Pareto and EI acquisition. For dynamic propagation, the cutoff is checked after updating the current ancestor, as in Algorithm 3.

For refinement outputs with raw $K_c = 0$, the generator sets `is_valid=False`, excluding them from subsequent parent selection. Setting $\kappa_f = 0$ in an ablation leaves this validity flag unchanged: valid candidates remain eligible even though they receive no forward inheritance.

C Complete Experimental Results

C.1 Code Results at the Full Budget

Tables 6 and 7 report final-answer accuracy and oracle coverage at $B = 128$. All values are percentages, summarized as mean $\pm$ standard deviation over three runs. The final selector returns the latest candidate attaining the largest observable score. Model and method abbreviations follow Table 1.

Table 6: Complete LCBv6 results at $B = 128$. Bold marks the highest mean in each model–metric comparison.

Method	Qwen3-4B	Qwen3-8B	Qwen3-14B	AReaL-14B	Qwen2.5-7B
Final-answer accuracy (%)					
RS	55.7 ± 2.6	55.6 ± 0.5	64.4 ± 0.7	65.3 ± 1.0	$\mathbf{29.7} \pm 0.8$
S	60.2 ± 2.3	60.8 ± 0.8	67.8 ± 0.7	67.8 ± 1.4	21.7 ± 0.0
MCTS	58.7 ± 0.7	60.2 ± 0.7	65.7 ± 2.1	68.0 ± 1.6	26.7 ± 1.0
AB-A	61.1 ± 1.2	58.5 ± 1.0	65.1 ± 0.8	66.1 ± 0.3	27.2 ± 1.0
AB-M	57.7 ± 1.2	59.8 ± 1.4	64.2 ± 1.0	68.2 ± 1.5	27.7 ± 0.3
DTS-EI	$\mathbf{61.3} \pm 0.5$	61.1 ± 0.0	70.6 ± 0.2	70.3 ± 1.2	28.8 ± 2.1
DTS-Pareto	59.8 ± 2.2	$\mathbf{63.0} \pm 0.3$	$\mathbf{71.6} \pm 0.6$	$\mathbf{71.6} \pm 0.3$	27.6 ± 1.0
Oracle coverage (%)					
RS	$\mathbf{72.8} \pm 1.1$	$\mathbf{74.7} \pm 1.4$	$\mathbf{80.2} \pm 0.7$	$\mathbf{82.7} \pm 0.5$	$\mathbf{35.2} \pm 0.3$
S	67.0 ± 1.2	69.0 ± 0.3	76.2 ± 0.7	77.0 ± 0.5	23.0 ± 0.0
MCTS	69.0 ± 1.2	72.0 ± 0.9	77.1 ± 0.9	78.3 ± 1.2	29.9 ± 0.3
AB-A	71.7 ± 2.8	73.5 ± 0.3	78.7 ± 0.5	80.2 ± 1.2	32.2 ± 0.7
AB-M	71.1 ± 1.0	73.7 ± 0.7	78.7 ± 1.8	80.6 ± 0.9	32.6 ± 0.6
DTS-EI	70.3 ± 1.2	73.1 ± 1.2	77.3 ± 0.7	79.4 ± 0.5	33.0 ± 1.5
DTS-Pareto	69.0 ± 0.3	71.8 ± 0.5	77.0 ± 0.5	80.2 ± 0.7	31.8 ± 1.2

C.2 Additional Code Budget Curves

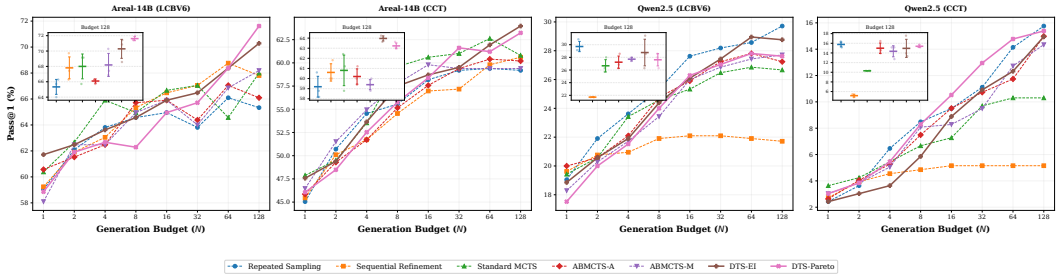


Figure 6: Additional budget curves for AReaL-boba-2-14B (Areal-14B) and Qwen2.5-7B-Instruct (Qwen2.5) on LCBv6 and CCT. Curves show three-run means under latest-maximizer selection. The horizontal-axis budget N denotes candidate calls B; insets show variation at $B = 128$.

Table 7: Complete CCT results at $B = 128$. Bold marks the highest mean in each model–metric comparison.

Method	Qwen3-4B	Qwen3-8B	Qwen3-14B	AReaL-14B	Qwen2.5-7B
Final-answer accuracy (%)					
RS	49.5 ± 3.2	51.7 ± 1.6	54.2 ± 1.0	59.2 ± 1.0	$\mathbf{15.8} \pm 0.5$
S	48.9 ± 2.9	52.3 ± 1.2	55.0 ± 0.3	60.6 ± 0.9	5.2 ± 0.3
MCTS	50.4 ± 0.6	51.4 ± 1.0	57.8 ± 0.8	60.8 ± 1.5	10.3 ± 0.0
AB-A	51.1 ± 1.6	52.7 ± 0.9	56.6 ± 0.6	60.2 ± 0.8	14.9 ± 1.1
AB-M	48.9 ± 2.5	53.2 ± 0.9	55.7 ± 0.8	59.4 ± 0.5	14.3 ± 1.1
DTS-EI	$\mathbf{52.3} \pm 0.8$	54.3 ± 0.8	57.8 ± 0.6	$\mathbf{63.8} \pm 0.3$	14.9 ± 1.9
DTS-Pareto	49.4 ± 0.9	$\mathbf{54.5} \pm 0.0$	$\mathbf{59.8} \pm 0.8$	63.2 ± 0.3	15.4 ± 0.3
Oracle coverage (%)					
RS	$\mathbf{61.6} \pm 0.6$	$\mathbf{63.1} \pm 1.1$	$\mathbf{69.4} \pm 0.5$	$\mathbf{73.9} \pm 0.9$	$\mathbf{18.8} \pm 1.0$
S	56.6 ± 2.6	58.9 ± 1.0	65.0 ± 0.6	68.5 ± 0.9	5.2 ± 0.3
MCTS	58.0 ± 0.3	60.3 ± 1.2	65.1 ± 0.8	71.1 ± 1.4	11.5 ± 0.0
AB-A	60.0 ± 1.0	62.0 ± 0.3	67.9 ± 0.9	70.9 ± 0.9	16.8 ± 1.1
AB-M	58.0 ± 1.0	62.5 ± 1.1	68.4 ± 1.4	71.5 ± 1.0	16.8 ± 0.8
DTS-EI	60.6 ± 0.5	60.6 ± 0.0	67.9 ± 1.0	71.1 ± 1.0	17.0 ± 1.8
DTS-Pareto	59.7 ± 0.7	61.4 ± 0.3	68.1 ± 1.2	70.7 ± 1.0	17.2 ± 1.0

C.3 Complete Ablation Results

The ablations use LCBv6 with Qwen3-14B, $B = 128$, and the default latest-maximizer final selector. The operations are specified in Appendix B.6. "No K-aware transfer" is the configuration previously labeled "No Credit"; it resets the child prior and retains unit-weight backward propagation. Table 8 preserves the reported summary values, and Table 9 reports the available three-run values rounded to one decimal place.

Table 8: Complete component-ablation results (%, mean $\pm$ standard deviation over three runs). Both full DTS variants are included as references.

Configuration	Final-answer accuracy	Oracle coverage
Full DTS-Pareto	71.6 ± 0.6	77.0 ± 0.5
Full DTS-EI	70.6 ± 0.2	77.3 ± 0.7
Global TS	67.0 ± 1.5	77.9 ± 0.3
Fixed 50/50	64.6 ± 2.2	79.6 ± 0.3
No K-aware transfer	65.0 ± 1.0	77.3 ± 0.3
Random parent	66.7 ± 1.2	78.7 ± 1.0
Static $K = 1$	66.5 ± 1.2	79.1 ± 0.7
No inheritance (Pareto)	65.5 ± 2.3	78.3 ± 1.2
No inheritance (EI)	65.9 ± 0.7	77.7 ± 0.9

C.4 Supplementary Tasks: AIME and PMO

Tables 10 and 11 report the complete final-budget results. AIME-Base exposes exact correctness, so oracle coverage and final-answer accuracy coincide. In AIME-RM, the reward-model selector can

Table 9: Available run-level ablation results (%). Each entry is final-answer accuracy / oracle coverage.

Configuration	Run 1	Run 2	Run 3
Global TS	66.3 / 78.3	69.1 / 77.7	65.7 / 77.7
Fixed 50/50	66.9 / 80.0	61.7 / 79.4	65.1 / 79.4
No K-aware transfer	66.3 / 77.1	64.0 / 77.1	64.6 / 77.7
Random parent	68.0 / 80.0	65.1 / 78.3	66.9 / 77.7
Static $K = 1$	66.3 / 78.9	68.0 / 80.0	65.1 / 78.3
No inheritance (Pareto)	62.3 / 77.7	67.4 / 80.0	66.9 / 77.1
No inheritance (EI)	65.1 / 76.6	65.7 / 78.9	66.9 / 77.7

fail even when the pool contains a correct answer. For example, AB-MCTS-A has 78.8% coverage and 56.4% accuracy, whereas DTS-EI has 74.4% coverage and 63.9% accuracy.

Table 10: AIME results with Qwen3-8B at $B = 64$ (mean $\pm$ standard deviation over three runs, %). Both protocols use the same 60 problems.

	AIME-Base	AIME-RM	
Method	Coverage / Accuracy	Coverage	Accuracy
RS	78.3 ± 1.4	78.3 ± 1.4	60.0 ± 3.6
S	73.9 ± 2.1	69.4 ± 0.8	61.1 ± 3.4
MCTS	76.1 ± 1.6	72.8 ± 0.8	60.6 ± 3.9
AB-MCTS-A	77.2 ± 1.6	78.8 ± 0.6	56.4 ± 2.2
AB-MCTS-M	76.1 ± 0.8	76.1 ± 0.8	58.3 ± 2.4
DTS-EI	78.9 ± 0.8	74.4 ± 0.8	63.9 ± 3.4
DTS-Pareto	75.6 ± 2.1	77.8 ± 0.8	60.6 ± 1.6

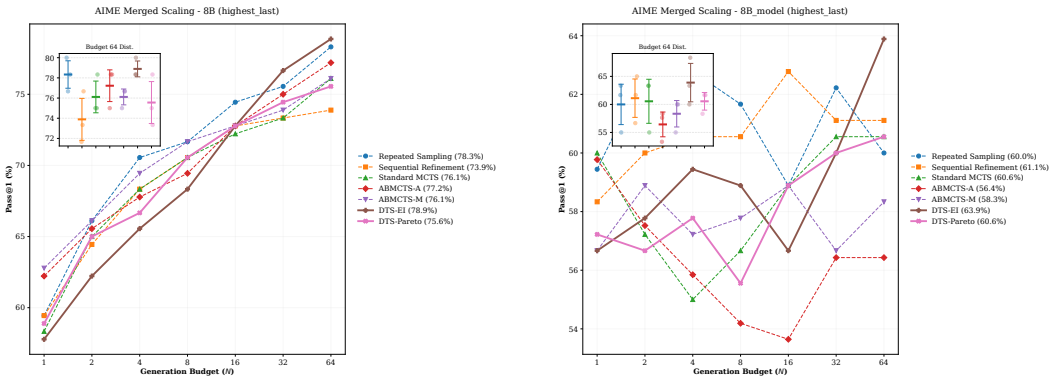


Figure 7: AIME final-answer accuracy across budgets under exact-correctness and reward-model feedback with Qwen3-8B. The horizontal generation budget is the candidate-call budget B used in this paper.

The PMO gain of EI over RS is 0.0276 objective units. The two DTS variants do not share the same ranking across these supplementary settings; the tables report them separately throughout.

Table 11: PMO results with gpt-4o-mini at $B = 128$ (three-run mean $\pm$ standard deviation, original objective units). `top1_f` is the final selected objective score; `top10_f` and `auc_top10` retain the reported frontier-quality and area-under-curve summaries.

Method	top1_f	top10_f	auc_top10
RS	3.7775 ± 0.0287	3.7070 ± 0.0091	3.6455 ± 0.0053
S	3.6566 ± 0.1146	3.5929 ± 0.0948	3.5506 ± 0.0885
MCTS	3.7495 ± 0.0247	3.7170 ± 0.0403	3.6582 ± 0.0371
AB-MCTS-A	3.7723 ± 0.0696	3.7041 ± 0.0210	3.6506 ± 0.0065
AB-MCTS-M	3.7279 ± 0.0197	3.6882 ± 0.0021	3.6409 ± 0.0182
DTS-EI	3.8051 ± 0.0384	3.7345 ± 0.0215	3.6724 ± 0.0057
DTS-Pareto	3.7182 ± 0.0352	3.6889 ± 0.0228	3.6421 ± 0.0249

D Final-Answer Selection Analysis

For binary correctness, let C be the event that the generated pool contains a correct candidate and A the event that the returned candidate is correct. Since $A \subseteq C$, $\Pr(A) = \Pr(C)\Pr(A \mid C)$ for $\Pr(C) > 0$. The ratios in Section 5.2 use the reported accuracy mean divided by the coverage mean; they are not averages of per-run ratios. The covered tasks can differ across methods, so these summaries do not compare identical task subsets.

D.1 Selection Rules and Counting

For a fixed ordered candidate pool $\mathcal{P}_B$, define $T = \{i : R_{\text{obs}}(x_i) = \max_j R_{\text{obs}}(x_j)\}$. Last-argmax returns $x_{\max T}$ and first-argmax returns $x_{\min T}$. Under uniform tie-breaking, the task-level expected correctness is

$$u(\mathcal{P}_B) = \frac{1}{|T|} \sum_{i \in T} \mathbf{1}\{x_i \text{ is correct}\}. \tag{12}$$

Each generation event contributes one entry, including duplicate outputs. Uniform selection averages correctness over these entries exactly, then over tasks; it introduces no additional tie-sampling noise. Means and standard deviations summarize three search runs. Changing the final tie rule leaves each run's generated pool and oracle coverage unchanged.

D.2 Per-Setting Results

Table 12 reports all ten code settings under the three final tie rules. The last-argmax rows match the accuracy entries in Tables 6 and 7.

Table 12: Final-answer accuracy (%, mean $\pm$ standard deviation over three runs) for all ten code settings at $B = 128$, under last, first, and uniform tie rules. Bold marks the highest mean within each setting and rule. AReaL-14B and Qwen2.5-7B abbreviate AReaL-boba-2-14B and Qwen2.5-7B-Instruct.

Tie rule	RS	S	MCTS	AB-A	AB-M	DTS-EI	DTS-Pareto
LCBv6 / Qwen3-4B							
Last	55.7 ± 2.6	60.2 ± 2.3	58.7 ± 0.7	61.1 ± 1.2	57.7 ± 1.2	$\mathbf{61.3} \pm 0.5$	59.8 ± 2.2
First	54.5 ± 0.5	56.4 ± 1.3	56.4 ± 1.3	$\mathbf{59.1} \pm 1.4$	56.8 ± 1.1	55.8 ± 1.0	55.8 ± 2.9
Uniform	55.4 ± 0.7	58.3 ± 0.9	58.1 ± 0.6	59.3 ± 1.5	57.6 ± 0.6	$\mathbf{59.9} \pm 0.8$	58.6 ± 0.3
LCBv6 / Qwen3-8B							
Last	55.6 ± 0.5	60.8 ± 0.8	60.2 ± 0.7	58.5 ± 1.0	59.8 ± 1.4	61.1 ± 0.0	$\mathbf{63.0} \pm 0.3$
First	57.1 ± 0.8	56.4 ± 1.9	57.7 ± 1.2	56.8 ± 2.3	57.0 ± 1.2	57.9 ± 0.7	$\mathbf{60.0} \pm 0.5$
Uniform	57.4 ± 0.4	60.1 ± 1.0	59.7 ± 0.8	58.8 ± 0.7	59.0 ± 0.3	60.2 ± 0.8	$\mathbf{60.8} \pm 0.8$

Continued on next page

Table 12: Final-answer accuracy under three tie rules (continued).

Tie rule	RS	S	MCTS	AB-A	AB-M	DTS-EI	DTS-Pareto
				LCBv6 / Qwen3-14B			
Last	64.4 ± 0.7	67.8 ± 0.7	65.7 ± 2.1	65.1 ± 0.8	64.2 ± 1.0	70.6 ± 0.2	$\mathbf{71.6} \pm 0.6$
First	65.0 ± 2.1	64.2 ± 1.8	63.0 ± 2.1	$\mathbf{65.3} \pm 1.9$	63.2 ± 1.1	62.7 ± 0.6	64.7 ± 1.4
Uniform	63.5 ± 1.0	67.5 ± 0.1	65.5 ± 0.3	65.3 ± 0.4	64.8 ± 0.5	$\mathbf{68.1} \pm 0.4$	67.7 ± 0.6
				LCBv6 / AReaL-14B			
Last	65.3 ± 1.0	67.8 ± 1.4	68.0 ± 1.6	66.1 ± 0.3	68.2 ± 1.5	70.3 ± 1.2	$\mathbf{71.6} \pm 0.3$
First	64.8 ± 2.2	63.4 ± 0.5	65.7 ± 0.8	65.9 ± 1.3	62.7 ± 1.9	$\mathbf{66.7} \pm 1.2$	62.7 ± 1.5
Uniform	65.2 ± 0.1	68.0 ± 0.3	66.6 ± 0.1	66.1 ± 0.4	65.9 ± 0.1	$\mathbf{68.3} \pm 1.4$	67.9 ± 0.2
				LCBv6 / Qwen2.5-7B			
Last	$\mathbf{29.7} \pm 0.8$	21.7 ± 0.0	26.7 ± 1.0	27.2 ± 1.0	27.7 ± 0.3	28.8 ± 2.1	27.6 ± 1.0
First	$\mathbf{29.5} \pm 0.7$	22.3 ± 0.8	26.9 ± 0.8	28.0 ± 0.8	28.0 ± 0.6	29.3 ± 1.9	28.2 ± 1.4
Uniform	$\mathbf{29.3} \pm 0.5$	22.0 ± 0.3	27.3 ± 0.7	28.0 ± 0.6	27.8 ± 0.0	28.8 ± 1.9	28.1 ± 0.9
				CCT / Qwen3-4B			
Last	49.5 ± 3.2	48.9 ± 2.9	50.4 ± 0.6	51.1 ± 1.6	48.9 ± 2.5	$\mathbf{52.3} \pm 0.8$	49.4 ± 0.9
First	47.9 ± 0.0	45.6 ± 2.3	49.4 ± 0.2	49.9 ± 0.6	48.4 ± 1.5	50.1 ± 1.9	$\mathbf{51.6} \pm 1.3$
Uniform	48.5 ± 1.2	49.0 ± 1.5	49.2 ± 0.4	50.8 ± 1.3	48.9 ± 0.9	$\mathbf{52.0} \pm 1.0$	50.7 ± 0.9
				CCT / Qwen3-8B			
Last	51.7 ± 1.6	52.3 ± 1.2	51.4 ± 1.0	52.7 ± 0.9	53.2 ± 0.9	54.3 ± 0.8	$\mathbf{54.5} \pm 0.0$
First	50.9 ± 1.3	49.7 ± 1.8	50.4 ± 2.4	$\mathbf{52.5} \pm 2.5$	51.4 ± 0.4	51.3 ± 2.3	50.2 ± 2.0
Uniform	51.2 ± 0.9	51.3 ± 0.5	51.8 ± 0.7	52.9 ± 1.2	52.5 ± 0.2	$\mathbf{54.3} \pm 0.3$	53.4 ± 0.9
				CCT / Qwen3-14B			
Last	54.2 ± 1.0	55.0 ± 0.3	57.8 ± 0.8	56.6 ± 0.6	55.7 ± 0.8	57.8 ± 0.6	$\mathbf{59.8} \pm 0.8$
First	55.2 ± 2.6	53.0 ± 1.5	55.4 ± 0.3	55.2 ± 1.8	55.1 ± 0.8	56.4 ± 0.5	$\mathbf{57.2} \pm 2.9$
Uniform	54.9 ± 0.5	55.3 ± 0.2	56.9 ± 0.5	56.2 ± 0.6	56.1 ± 0.7	57.8 ± 1.0	$\mathbf{58.0} \pm 1.1$
				CCT / AReaL-14B			
Last	59.2 ± 1.0	60.6 ± 0.9	60.8 ± 1.5	60.2 ± 0.8	59.4 ± 0.5	$\mathbf{63.8} \pm 0.3$	63.2 ± 0.3
First	$\mathbf{61.0} \pm 1.0$	59.2 ± 1.5	60.4 ± 0.6	60.0 ± 2.2	60.2 ± 2.1	60.9 ± 0.6	59.2 ± 0.8
Uniform	60.8 ± 0.7	60.2 ± 0.8	61.8 ± 0.8	61.0 ± 0.5	60.7 ± 0.2	$\mathbf{62.5} \pm 0.4$	62.1 ± 0.2
				CCT / Qwen2.5-7B			
Last	$\mathbf{15.8} \pm 0.5$	5.2 ± 0.3	10.3 ± 0.0	14.9 ± 1.1	14.3 ± 1.1	14.9 ± 1.9	15.4 ± 0.3
First	15.8 ± 1.5	4.8 ± 0.6	10.9 ± 0.0	14.2 ± 1.1	14.7 ± 2.0	14.7 ± 0.8	$\mathbf{16.0} \pm 1.0$
Uniform	15.1 ± 1.2	5.1 ± 0.3	10.3 ± 0.0	14.9 ± 0.8	14.6 ± 0.6	14.7 ± 1.8	$\mathbf{15.7} \pm 0.5$

D.3 Cross-Setting Summaries

Table 13 summarizes both all ten code settings and the eight Qwen3 and AReaL settings discussed in the main text. Each gain averages the DTS-minus-RS accuracy difference across settings with equal weight, using the same final tie rule for both methods. Counts record settings with a strictly higher mean than RS. Aggregates are computed from the displayed per-setting means.

Table 13: Accuracy gains over RS under the same final tie rule. Mean gains are in percentage points; counts give settings with a higher mean than RS. The eight-setting subset excludes Qwen2.5-7B-Instruct.

		All ten settings		Eight-setting subset	
Tie rule	Method	Mean gain (pp)	Above RS	Mean gain (pp)	Above RS
Last	DTS-EI	$+3.41$	$8/10$	$+4.49$	$8/8$
Last	DTS-Pareto	$+3.48$	$7/10$	$+4.66$	$7/8$
First	DTS-EI	$+0.41$	$6/10$	$+0.68$	$6/8$
First	DTS-Pareto	$+0.39$	$5/10$	$+0.63$	$4/8$
Uniform	DTS-EI	$+2.53$	$8/10$	$+3.28$	$8/8$
Uniform	DTS-Pareto	$+2.17$	$9/10$	$+2.79$	$8/8$

D.4 Coupled-DTS under Different Final-Selection Rules

Table 14 compares DTS with the coupled-DTS control on LCBv6 / Qwen3-14B and CCT / Qwen3-8B at $B = 128$, for both EI and Pareto. The coupled parent selector is specified in Appendix B.4. Each method keeps its own generated pools fixed when the final-selection rule changes. Last and First choose the latest and earliest feedback maximizers; Uniform reports the expected accuracy when choosing among maximizers with equal probability.

Table 14: Coupled and global parent selection at $B = 128$. Coverage and final-answer accuracy (Pass@1) are percentages, reported as mean $\pm$ standard deviation over three runs. Coverage is unchanged across final-selection rules for a given method. Bold marks the higher mean accuracy within each coupled/global pair under each rule.

		Final-answer accuracy (Pass@1)		
Method	Coverage	Last	Uniform	First
LCBv6 / Qwen3-14B				
Coupled-DTS-EI	77.90 ± 0.71	63.81 ± 1.35	65.05 ± 0.46	$\mathbf{63.81} \pm 0.97$
DTS-EI	77.3 ± 0.7	$\mathbf{70.6} \pm 0.2$	$\mathbf{68.1} \pm 0.4$	62.7 ± 0.6
Coupled-DTS-Pareto	77.90 ± 0.71	65.33 ± 0.97	65.09 ± 0.27	$\mathbf{64.76} \pm 0.97$
DTS-Pareto	77.0 ± 0.5	$\mathbf{71.6} \pm 0.6$	$\mathbf{67.7} \pm 0.6$	64.7 ± 1.4
CCT / Qwen3-8B				
Coupled-DTS-EI	61.41 ± 1.74	51.52 ± 0.49	52.70 ± 0.85	$\mathbf{52.32} \pm 1.59$
DTS-EI	60.6 ± 0.0	$\mathbf{54.3} \pm 0.8$	$\mathbf{54.3} \pm 0.3$	51.3 ± 2.3
Coupled-DTS-Pareto	61.41 ± 0.76	53.13 ± 0.76	52.89 ± 1.21	$\mathbf{52.53} \pm 1.25$
DTS-Pareto	61.4 ± 0.3	$\mathbf{54.5} \pm 0.0$	$\mathbf{53.4} \pm 0.9$	50.2 ± 2.0

Under Last, global selection improves mean accuracy by approximately 6.8 and 6.3 percentage points for EI and Pareto on LCBv6, and 2.8 and 1.4 points on CCT. These gains occur with comparable or slightly lower coverage. The gains remain positive under Uniform for both acquisition rules in both settings. Under First, the LCBv6 Pareto results are nearly tied, while global selection has lower mean accuracy in the other three comparisons. Thus, the gains extend beyond Last, but their size and direction depend on the final-selection rule.

E Offline Diagnostics of Search Utility

E.1 Feedback and Recovery Estimators

The diagnostics use two probing procedures: repeated sampling for feedback error, and sequential refinement for local recovery. The full diagnostic budget is $B_{\mathrm{diag}} = 128$ for code and PMO and

64 for AIME. These quantities characterize the evaluated task–model–feedback settings and are not inputs to DTS.

Error among top-feedback candidates. For instance i, let $\mathcal{P}_i^{\mathrm{RS}}$ be its repeated-sampling pool, and define

$$\mathcal{T}_i = \operatorname*{argmax}_{x \in \mathcal{P}_i^{\mathrm{RS}}} R_{\mathrm{obs}}(x), \tag{13}$$

$$\mathcal{I}_{\mathrm{head}} = \{i : \max_{x \in \mathcal{P}_i^{\mathrm{RS}}} R_{\mathrm{obs}}(x) \geq 0.5\}. \tag{14}$$

For settings with proxy feedback and nonempty $\mathcal{I}_{\mathrm{head}}$, we report

$$X_{\mathrm{trap}} = \frac{1}{|\mathcal{I}_{\mathrm{head}}|} \sum_{i \in \mathcal{I}_{\mathrm{head}}} \frac{1}{|\mathcal{T}_i|} \sum_{x \in \mathcal{T}_i} \mathbf{1}[R_{\mathrm{eval}}(x) < 0.1]. \tag{15}$$

Code uses binary public/private-test rewards; AIME-RM uses the normalized PRM score and binary answer correctness. Thus the indicator denotes an incorrect candidate. All tied maximizers contribute equally within an instance, and instances receive equal weight. This differs from applying the latest-maximizer final selector. AIME-Base and PMO use direct target feedback and are assigned $X_{\mathrm{trap}} = 0$ in the diagnostic summary.

Recovery from an incorrect seed. A diagnostic trajectory j is $(x_{0,j}, \ldots, x_{B_{\mathrm{diag}}-1,j})$, with fresh generation followed by sequential revisions. Write $s_{k,j} = R_{\mathrm{eval}}(x_{k,j})$ and $\mathcal{J}_{\mathrm{flawed}} = \{j : s_{0,j} < 1 - 0.01\}$. For binary correctness tasks,

$$\widehat{Y}_{\mathrm{local}} = \frac{1}{|\mathcal{J}_{\mathrm{flawed}}|} \sum_{j \in \mathcal{J}_{\mathrm{flawed}}} \left(\max_{0 \leq k < B_{\mathrm{diag}}} s_{k,j} - s_{0,j} \right). \tag{16}$$

This is the fraction of initially incorrect trajectories that reach a correct candidate at least once. It is defined as zero when the flawed set is empty. For PMO, the reported coordinate instead uses

$$Y_{\mathrm{local}}^{\mathrm{PMO}} = \frac{\mathrm{Top1}_{\mathrm{SR,final}} - \mathrm{Top1}_{\mathrm{init}}}{\mathrm{Top1}_{\mathrm{global\ oracle}} - \mathrm{Top1}_{\mathrm{init}} + 10^{-6}}. \tag{17}$$

Here the three terms are the best objective after sequential refinement, among initial diagnostic candidates, and over the full diagnostic pool, respectively. This coordinate is normalized objective improvement, not a binary recovery probability.

E.2 Observed Coordinates and Search Gains

Table 15 reports the coordinates used in Figure 8. The gain column is $\Delta_{\mathrm{complex}} = \max_{m \in \mathcal{M}} \mathrm{Score}(m) - \mathrm{Score}(\mathrm{RS})$, where $\mathcal{M}$ contains MCTS, AB-MCTS-A, AB-MCTS-M, DTS-EI, and DTS-Pareto. It summarizes the best evaluated complex search method in each setting, rather than a single algorithm chosen before evaluation. AReaL is derived from Qwen3-14B through RL post-training.

Table 15: Offline diagnostic coordinates (mean ± standard deviation) and search gains. Code and AIME scores are accuracy percentages and gains are percentage points; PMO uses objective units. The selected complex method is identified separately from the RS comparison.

Setting	X_{trap}	Y_{local}	RS	Best complex	Δ	Method
LCBv6 / Qwen3-4B	0.4055 ± 0.0101	0.3358 ± 0.0399	55.7	61.3	+5.6	EI
LCBv6 / Qwen3-8B	0.3917 ± 0.0124	0.3578 ± 0.0210	55.6	63.0	+7.4	Pareto
LCBv6 / Qwen3-14B	0.3308 ± 0.0198	0.4247 ± 0.0404	64.4	71.6	+7.2	Pareto
LCBv6 / Qwen2.5-7B	0.5395 ± 0.0156	0.0425 ± 0.0095	29.7	28.8	−0.9	EI
LCBv6 / AReaL-14B	0.3140 ± 0.0013	0.4344 ± 0.0151	65.3	71.6	+6.3	Pareto
CCT / Qwen3-4B	0.4174 ± 0.0429	0.3502 ± 0.0218	49.5	52.3	+2.8	EI
CCT / Qwen3-8B	0.4039 ± 0.0389	0.3438 ± 0.0266	51.7	54.5	+2.8	Pareto
CCT / Qwen3-14B	0.3775 ± 0.0205	0.3869 ± 0.0294	54.2	59.8	+5.6	Pareto
CCT / Qwen2.5-7B	0.5374 ± 0.0122	0.0249 ± 0.0060	15.8	15.4	−0.4	Pareto
CCT / AReaL-14B	0.3055 ± 0.0038	0.4223 ± 0.0084	59.2	63.8	+4.6	EI
AIME-Base / Qwen3-8B	0.0000 ± 0.0000	0.3567 ± 0.0428	78.3	78.9	+0.6	EI
AIME-RM / Qwen3-8B	0.4000 ± 0.0360	0.2619 ± 0.0568	60.0	63.9	+3.9	EI
PMO / gpt-4o-mini	0.0000 ± 0.0000	0.3811 ± 0.4415	3.7775	3.8051	+0.0276	EI

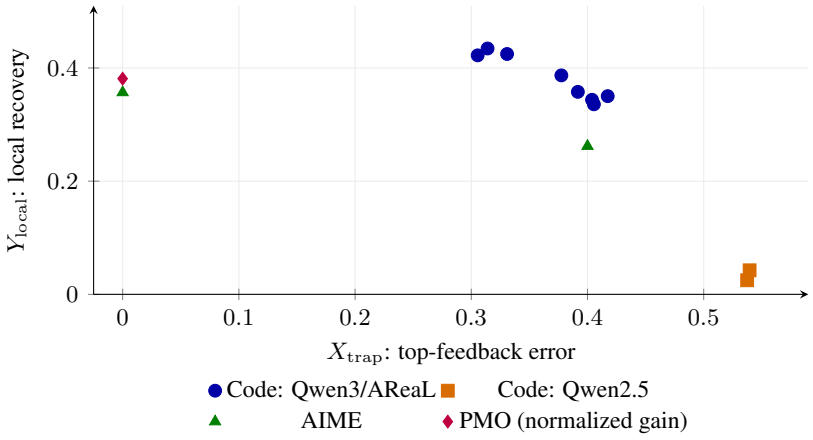


Figure 8: Observed diagnostic means. Symbols distinguish evaluated settings. PMO uses normalized objective improvement on the vertical axis. Table 15 reports uncertainty, including the large variation in the PMO estimate.

E.3 Reduced-Budget Estimation

To assess how early the diagnostic signals emerge, we use the first 16 candidates per repeated-sampling pool for X_{trap} and per sequential-refinement trajectory for Y_{local}. Both estimates aggregate across the evaluated labeled instances; 16 denotes the candidate budget per pool or trajectory, not the number of instances. Compared with 128 candidates, this uses one-eighth of each probing procedure's candidate budget. Table 16 shows that the resulting cross-setting rankings closely match those of the full-budget diagnostics. At 16 candidates, the two Qwen2.5 settings have

Table 16: Spearman correlation between 16- and 128-candidate diagnostic estimates across the ten code settings.

Diagnostic	Spearman correlation
X_{trap}	0.976
Y_{local}	0.927

$X_{\text{trap}} = 0.4398$–0.4808, versus 0.2544–0.4026 for the other eight, and $Y_{\text{local}} = 0.0249$–$0.0425$, versus 0.2021–0.3107. This early separation is consistent with the full-search results: at $B = 128$, neither DTS variant exceeds RS in the two low-recovery settings, whereas DTS-EI exceeds RS in the

other eight (Table 1). It supports low-budget, labeled assessment of whether refinement is promising in these settings. The reported correlations measure stability of diagnostic rankings, rather than accuracy in forecasting final scores. A calibrated rule for selecting algorithms on unseen settings requires separate validation.

F Additional Behavioral Analyses

F.1 Allocation and Branching Metrics

Metrics are computed from the realized candidate pool $\mathcal{P}_B$ and its parent links. Freshly generated candidates have depth 1; a refinement child x has $d(x) = d(p(x)) + 1$. Let $\mathcal{C} = \{x \in \mathcal{P}_B : d(x) \geq 2\}$ be the refinement events, and $A_d(x)$ the ancestor at depth d (including x itself when $d(x) = d$). Events are counted separately, including repeated outputs.

Allocation Quality (AQ). For each depth-1 root lineage ℓ, let $\mathrm{Desc}(\ell)$ exclude the root itself, and define

$$O(\ell) = \max_{x \in \mathrm{Desc}(\ell) \cup \{\ell\}} R_{\mathrm{eval}}(x) - R_{\mathrm{eval}}(\ell), \tag{18}$$

$$C(\ell) = \#\{x \in \mathcal{C} : A_1(x) = \ell\}, \qquad w(\ell) = \frac{C(\ell)}{\sum_{\ell'} C(\ell')}, \tag{19}$$

$$\mathrm{AQ} = \sum_{\ell} w(\ell) O(\ell). \tag{20}$$

For binary correctness, $O(\ell) = 1$ exactly when the root is incorrect and a descendant is correct. AQ therefore counts the fraction of refinement calls assigned to such lineages, including calls both before and after recovery. Correct roots have zero recovery opportunity under this definition. AQ describes realized recovery allocation, rather than absolute lineage quality or the ability to predict recovery in advance.

Root and conditional entropies. Let $U = A_1(X)$ and $V = A_2(X)$ for an event X sampled uniformly from $\mathcal{C}$. The root breadth and within-root branching metrics are

$$H_1 = H(U), \qquad H_{2|1} = H(V \mid U), \tag{21}$$

where H uses base-2 logarithms and empirical event frequencies. For $H_{3|2}$, sample uniformly from $\mathcal{C}_{\geq 3} = \{x \in \mathcal{C} : d(x) \geq 3\}$ and compute $H(A_3(X) \mid A_2(X))$. Thus $(H_1, H_{2|1}, H_{3|2})$ is a multiscale profile; the different depth-conditioned populations mean that its components should not be added as a single total-entropy decomposition.

Sibling entropy. For a parent p, write $n_p = |\mathrm{Ch}(p)|$ for its number of refinement children. The event-weighted local branching measure is

$$H_{\mathrm{sibling}} = \sum_{p : n_p > 0} \frac{n_p}{\sum_q n_q} \log_2 n_p. \tag{22}$$

It is zero for a serial chain and positive when parents receive multiple children. When there are no refinement events, AQ and these entropies are undefined; RS is therefore shown as a no-continuation reference.

F.2 Observed Allocation and Topology

Table 17 gives LCBv6 statistics, and Figures 9 and 10 show the three-panel analyses. In both benchmarks, DTS places more refinement weight on lineages exhibiting recovery than the explicit-tree baselines while showing lower root-lineage entropy. Serial refinement is a useful single-lineage reference: its high AQ does not imply broad exploration or the highest final accuracy.

Table 17: Behavioral statistics on LCBv6 / Qwen3-14B at $B = 128$. AQ is reported as mean $\pm$ standard deviation (%); entropy columns contain reported means in bits.

Method	AQ (%)	H_1	H_{sibling}	$H_{2\|1}$	$H_{3\|2}$
S	17.71 ± 2.47	0.00	0.00	0.00	0.00
MCTS	9.55 ± 0.32	2.00	2.30	2.13	2.22
AB-MCTS-A	7.57 ± 0.86	3.65	1.42	1.62	0.59
AB-MCTS-M	5.87 ± 0.54	4.94	0.64	0.69	0.25
DTS-EI	14.81 ± 1.15	0.50	1.09	0.47	0.27
DTS-Pareto	13.47 ± 1.16	0.90	0.58	0.69	0.49

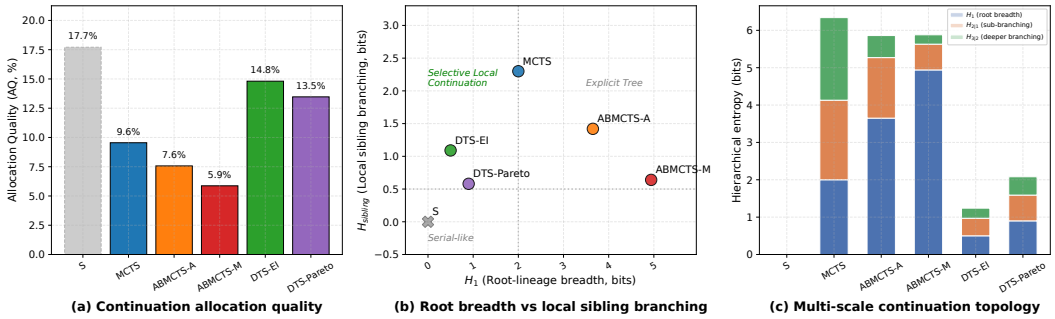


Figure 9: Allocation and topology on LCBv6 / Qwen3-14B. The panels show AQ, root versus sibling entropy, and the multiscale entropy profile. The last panel compares components with different depth-conditioned event sets; its stacked height is not a total entropy. S is the single-lineage reference; RS has no refinement events. ABMCTS-A/M denote AB-MCTS-A/M.

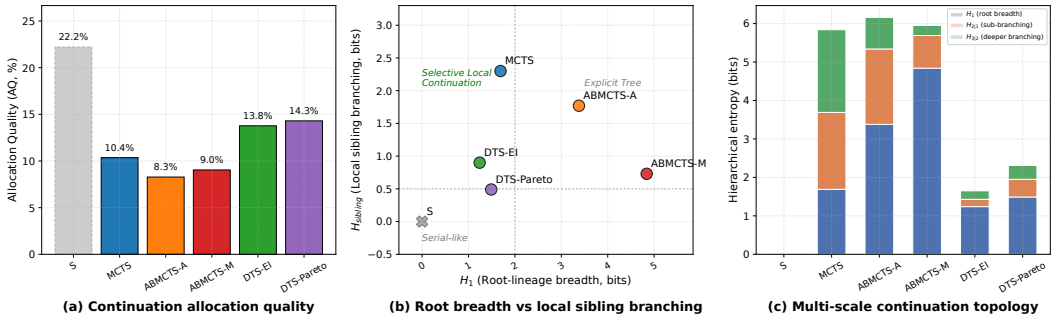


Figure 10: Allocation and topology on CCT / Qwen3-14B. The panels show AQ, root versus sibling entropy, and the multiscale entropy profile. The last panel compares components with different depth-conditioned event sets; its stacked height is not a total entropy. S is the single-lineage reference; RS has no refinement events. ABMCTS-A/M denote AB-MCTS-A/M.

F.3 Budget Trajectories

Figure 11 relates final-answer accuracy to the coarse width–depth balance as the budget grows. The horizontal axis is $\log_2(\text{mean depth}/\text{mean width})$; larger values denote a deeper relative shape. This geometric summary complements the event-based entropies and is not an optimization target.

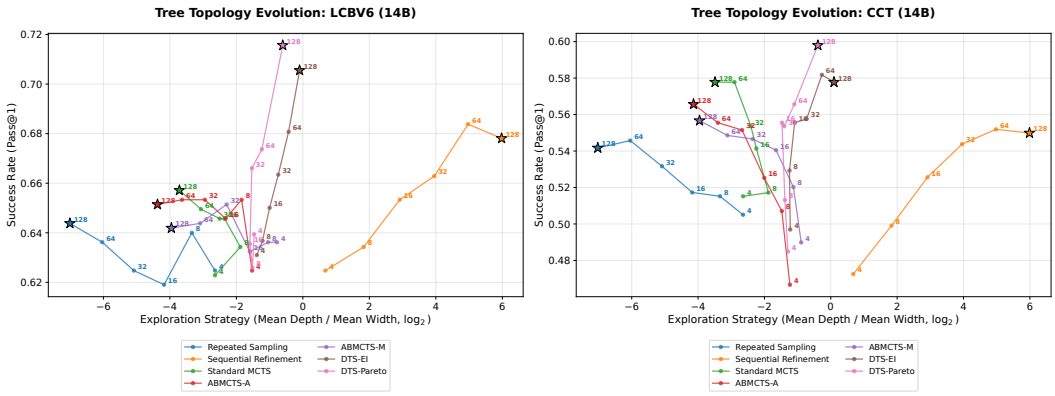


Figure 11: Width–depth and accuracy trajectories on LCBv6 and CCT with Qwen3-14B. Point labels denote candidate-call budgets.