WHERE TO RECOMPUTE: LEARNING RECOVERABILITY FOR COMPUTE-EFFICIENT LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time computation strengthens large language model reasoning only when the additional work changes the answer that the system ultimately returns, so deciding where and how much to recompute is central to compute-efficient inference. Existing methods face two challenges in making this decision. First, they target candidate correctness, continuation progress, or search coverage, yet a frozen selector can reject a repair or let a higher-scoring error replace a correct answer, so better generation does not by itself yield a better returned answer. Second, question-level allocation does not directly estimate the post-selection value of retaining a particular prefix and assigning it a continuation budget. To address the first challenge, we propose Recoverability-Aware Value of Computation (RAVEC), which learns the post-selection repair and damage of each prefix intervention from randomized same-parent interventions with a frozen generator and selector, and we prove a selector-gating identity under which changing only the selector can reverse the optimal intervention. To address the second challenge, we develop finite-horizon control over reopening positions, suffix budgets, and stopping, prove that path information improves on the best coarser policy exactly when its decision value exceeds estimation regret, and sharpen within-parent action ordering through parent-centered fitting without additional rollouts. Across 14 open models and 12 benchmarks, RAVEC improves core macro accuracy over adapted persistent-pool search (PB-SMC) by 1.34 points on average at equal expected operations, and at matched accuracy it reduces computation by 20.8% on MATH-500 and 25.4% on LiveCodeBench.

1 INTRODUCTION

Additional inference computation allows large language models to revisit difficult reasoning problems through longer generation, candidate search, and adaptive allocation (Snell et al., 2024; De Sabbata et al., 2024; Callaway et al., 2017; Muennighoff et al., 2025). Its practical value, however, is determined by the answer the system finally returns. A complete attempt also contains reusable work: a missing domain check calls only for a revised conclusion, whereas an incorrect substitution requires restarting from an earlier point. Deciding *where* to recompute, and with how much budget, therefore couples reasoning quality with the cost of preserving or replacing a trajectory.

Two challenges make this decision difficult. **Challenge 1: generation quality differs from selected-answer improvement.** Existing test-time methods mainly improve what is generated: best-of-N (BoN) ranks sampled candidates with a verifier (Lightman et al., 2023), and self-consistency aggregates sampled answers by voting (Wang et al., 2022), while process verifiers and PAV score step quality or continuation success (Lightman et al., 2023; Setlur et al., 2024). Once a frozen selector chooses the returned answer, a correct continuation can lose to an incorrect incumbent, and a higher-scoring error can displace a correct answer; larger coverage prevents neither event. **Challenge 2: the value of computation depends on the current path and the proposed intervention.** Difficulty-based allocation, metareasoning, and budget forcing regulate inference effort (Snell et al., 2024; De Sabbata et al., 2024; Muennighoff et al., 2025), and backtracking methods revisit prefixes guided by process scores or search access (Rohatgi et al., 2025; Tran et al., 2026). These signals do not directly estimate the post-selection value of a specific prefix intervention and its continuation budget; budget and difficulty confounds in trajectory analyses (Bulut, 2026) further require comparing actions within the same parent state.

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.

These challenges motivate two corresponding principles. **Motivation 1: measure computation by its effect after selection.** An intervention should be valued by the change it causes in the selected answer, with repair of an incorrect incumbent and damage to a correct one estimated separately because the selector gates both; randomized interventions cloned from the same parent hold the starting state fixed and thus isolate the action's effect. **Motivation 2: let the observed trajectory decide where and how much to recompute.** Already paid-for path information can value each eligible prefix and suffix budget, including subsequent decisions. For example, a trace solving $\sqrt{x+2} = x$ may correctly obtain $x = 2, -1$ but omit the domain check $x \geq 0$; reopening before that check preserves the valid algebra, provided the repaired answer survives selection (Appendix K.1). Such information justifies its cost only when it changes the best action and the controller recovers that value.

Following these principles, we make three contributions. **Contribution 1 (for Challenge 1): a selection-conditioned intervention target.** Recoverability-Aware Value of Computation (RAVEC) learns an *interventional recoverability surface*, namely the post-selection repair and damage probabilities of every retained position and suffix budget, from randomized same-parent interventions with a frozen generator and selector (Sections 3–4). A selector-gating identity (Proposition 1) shows that the preferred intervention can reverse when only the selector changes, even on unchanged candidate distributions. **Contribution 2 (for Challenge 2): path-informed computation control.** A finite-horizon controller chooses a reopening position and suffix budget, or STOP, under a fully charged operation ledger. An information-value theorem (Theorem 1) separates the benefit of path information from estimation regret, and an additional parent-centered fit sharpens within-parent action contrasts without extra rollouts (Sections 4–5). **Contribution 3: a controlled large-scale evaluation.** A study of 14 open models from four families on 12 mathematical, coding, and knowledge benchmarks combines matched-operation comparisons with a search–reward factorial, natural-parent position controls, fixed-bank selector exchange, and 15 component and replay ablations, each designed to test one of the two technical contributions (Sections 6–7).

The evidence follows this structure. For Contribution 1, training on selected outputs instead of coverage raises Qwen3-8B's MATH-500/LiveCodeBench accuracy by 1.7/2.8 points within the same incumbent search domain; against adapted PB-SMC, the 1.5/2.0-point selected-accuracy differences comprise 0.4/1.1 points of higher coverage and 1.1/0.9 points of reduced coverage–selection loss, and rescoring a fixed candidate bank with a different selector changes the preferred position for 23.1% of diagnostic parents. For Contribution 2, a root-only controller with identical inputs, supervision, and STOP rule trails one-step RAVEC by 1.7/1.9 points on natural math/code parents. Overall, RAVEC improves core macro accuracy over adapted PB-SMC by 1.34 points on average across the 14 models at equal expected operations and lowers matched-accuracy computation by 20.8% on MATH-500 and 25.4% on LiveCodeBench.

2 Related work

Adaptive computation and metareasoning. Compute-optimal inference allocates work according to estimated difficulty (Snell et al., 2024), rational metareasoning trades expected success against computation (De Sabbata et al., 2024; Callaway et al., 2017), and budget forcing extends reasoning (Muennighoff et al., 2025). General metareasoning selects computations, including allocation and stopping; our setting instantiates this principle for a frozen generator and selector. RAVEC decides *where* computation enters the current trajectory: its control object is a retained prefix together with a continuation budget, and its value is the resulting change in the selected answer. Theorem 1 makes the distinction exact: path-conditioned control beats the best coarser allocation precisely when its information dividend, zero unless optimal actions vary within coarse strata, exceeds estimation regret.

Process evaluation and prefix search. Process verification scores reasoning steps (Lightman et al., 2023); PAV measures the change in a prover's future success (Setlur et al., 2024); VGB studies backtracking with imperfect process verifiers (Rohatgi et al., 2025); and Subpool and Power Backtrack SMC (PB-SMC) revisit historical prefixes (Tran et al., 2026). These methods mainly focus on step quality, continuation progress, or access to historical prefixes. RAVEC instead learns the *post-selection effect of an intervention*, charging accepted errors as damage alongside accepted repairs. The difference is operational: by Proposition 1, changing only the selector can reverse the

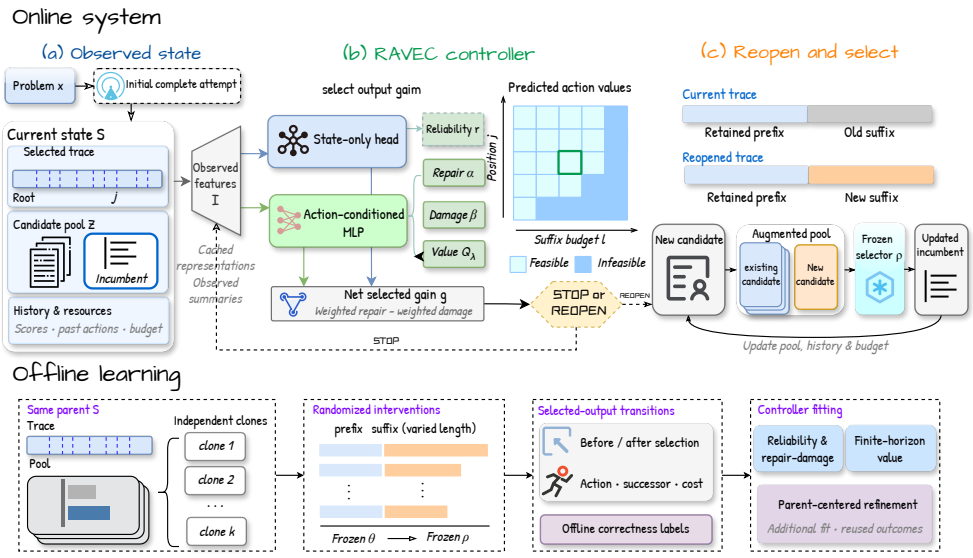


Figure 1: The RAVEC control loop. After a complete initial attempt, one batched value-model pass scores every feasible reopening position and suffix budget by its predicted post-selection repair, damage, and finite-horizon value; only the maximizing reopening is executed and the frozen selector rescores the pool. STOP, chosen when no action has positive value, returns the incumbent directly.

best reopening while the generated candidates remain unchanged, whereas a selector-independent continuation-success target assigns both selectors the same action values. The factorial therefore holds search access fixed when testing this target.

Positioning. BoN (Lightman et al., 2023) and self-consistency (Wang et al., 2022) select or aggregate independently sampled candidates, and analyses of search limits and overthinking (Chen et al., 2024; Qin et al., 2025; Rezazadeh & Davoodi, 2026) show that additional reasoning is not uniformly beneficial, motivating task-dependent evaluation. Existing methods thus answer three different questions: whether a new candidate is likely to be correct, whether a step improves continuation success, and which historical prefixes a search can revisit; metareasoning provides the broader framework for valuing computations. Our specific target is the selected-output effect of a prefix intervention from a fixed parent. RAVEC contributes five linked ingredients: randomized same-parent interventions, separate repair and damage probabilities, an explicit selector gate, a finite-horizon value that charges replay and controller calls, and a centered objective for within-parent action ordering. Complementary controls examine search access, supervision targets, and information content; the factorial and matched-feature root comparison isolate individual factors. Table 31 summarizes each adaptation's supervision, search, and accounting rules.

3 Problem: The Value of a Prefix Intervention

Let x be a problem, θ a frozen generator, $\mathcal{Z}$ the paid-for candidate pool, and ρ a frozen selector. The full history $S = (x, \mathcal{Z}, \mathcal{P}, \mathcal{H})$ records checkpoints and previous actions, and $I = \psi(S)$ is the compressed representation available to the controller. The selected answer is $\hat{z}(S) = \rho(x, \mathcal{Z})$, with correctness $Y(S) \in \{0, 1\}$. Reference answers and hidden tests supply offline labels only and are never online inputs.

The system first generates a complete attempt. Thereafter, STOP returns the incumbent, while $a = (j, \ell) = \text{REOPEN}(j, \ell)$ retains the original token prefix through position j and generates at most ℓ suffix tokens; root reopening is independent resampling. The discarded suffix, already paid for, remains visible to the controller but is hidden from the generator. A valid candidate joins the pool, after which ρ selects again. Invalid suffixes incur cost without replacing a valid incumbent, and an invalid initial attempt remains an explicit failure candidate, so no difficult question leaves the denominator.

For a common parent-information stratum I, define

$$\begin{aligned} r(I) &= \mathbb{P}(Y_0 = 1 \mid I), \\ \alpha(I, a) &= \mathbb{P}(Y_1 = 1 \mid Y_0 = 0, I, \mathrm{do}(a)), \\ \beta(I, a) &= \mathbb{P}(Y_1 = 0 \mid Y_0 = 1, I, \mathrm{do}(a)). \end{aligned} \tag{1}$$

These quantities separate current reliability r from the action-dependent repair probability α and damage probability β. Together they define the recoverability surface and its net selected-correctness gain:

$$\mathcal{G}_I(j, \ell) = (\alpha, \beta, g, \bar{c}), \qquad g(I, a) = (1 - r)\alpha - r\beta = \mathbb{E}[Y_1 - Y_0 \mid I, \mathrm{do}(a)]. \tag{2}$$

Here $\bar{c}$ is the expected operation cost. Unlike candidate correctness or continuation success, g is defined on the *selected* answer and therefore credits accepted repairs while charging accepted errors. All predictions condition on deployment-visible information; each labeled parent contributes to its supported initial-correctness branch, and α, β are learned across parent strata. The decision process is defined on the full history S, whereas the controller implements I.

The objective is $\max_\pi \mathbb{E}[Y_T]$ under an average operation allowance and a per-instance hard cap; a price $\lambda \geq 0$ yields the utility $\mathbb{E}[Y_T - \lambda C]$. The ledger charges initial generation, actual prefix replay or cache restoration, every suffix, terminal and process scoring, and controller calls, so preserved text is never credited as free computation. Hybrid recurrent models need recurrent-state checkpoints besides KV caches; otherwise replay is charged.

4 Learning and control

Action representation. Actions cross the root and legal boundaries near 25%, 50%, and 75% of the current trace with suffix caps $\{512, 1024, 2048\}$, merging duplicates, and at most six reopenings occur. Math boundaries are step or paragraph ends; code boundaries are newlines in the unmodified token stream. Final-answer completion uses a common 128-token reserve within each cap. Appendix C specifies format, initialization-failure, and cache rules.

Generator-specific hidden states at the question end, the checkpoint, the incumbent end, and pooled candidate endpoints are projected to 256 dimensions. A width-256 two-layer action MLP receives these vectors together with 20 observed score, position, history, and resource features and predicts α, β, and Q_λ; a separate state-only head predicts r independently of the proposed action. No future rollout enters as an input. Each generator has a separately fitted controller, and the generator and selector remain frozen. Figure 1 summarizes the control loop; exact parameter counts appear in the appendix.

From the two challenges to the controller. For Challenge 1, the repair and damage heads replace the coverage and continuation-success objectives of BoN, process verifiers, and PAV with the selected-output effect, while remaining directly comparable with them under a shared architecture. For Challenge 2, the position–budget grid and finite-horizon value replace question-level allocation: confidence supplies no within-trace ranking, and an immediate gain/cost rule omits the value of a useful successor state. Same-parent collection holds the starting question and outcome fixed, a direct-ΔY head offers an equal-information alternative to the factorization, and centered fitting refines relative ordering while absolute losses preserve the reference required for STOP. Table 24 pairs each choice with its alternative and the controlled comparison that distinguishes them.

Intervention data. Each training question provides two independent initial parents, and every position–budget action is expanded twice from independently cloned parents. Three training-state aggregation rounds cover the states reached by the learned policy. Transitions record (S, a, S', Y_0, Y_1, c), where labels are the answers *actually selected* by ρ and current features exclude successor information. All trajectories of a question stay within one train, calibration, validation, or test partition. The probability loss masks unsupported branches:

$$\mathcal{L}_{\mathrm{prob}} = \mathrm{BCE}(r, Y_0) + (1 - Y_0)\,\mathrm{BCE}(\alpha, Y_1) + Y_0\,\mathrm{BCE}(\beta, 1 - Y_1). \tag{3}$$

Reliability is counted once per parent. All-action collection weights parents equally, and subsampled actions require positive assignment probabilities to preserve the parent distribution. Calibration uses separate questions. Appendix B proves identification under common-support requirements.

Training-to-deployment contract. The generator θ, terminal selector ρ, checkpoint tokenizer, and action mask are frozen before controller fitting. Collection uses randomized actions with recorded assignment probabilities, calibration and controller selection use disjoint questions, and the final test policy is frozen before any benchmark score is read. Deployment exposes only $I = \psi(S)$, incumbent and pool summaries, and resource counters available before the proposed suffix is generated, never successor tokens, reference answers, or post-action scores. This contract prevents target leakage and fixes the estimand across all comparisons.

Finite-horizon control. The reward $R_\lambda = Y_1 - Y_0 - \lambda c$ telescopes:

$$\sum_{t<T} R_{\lambda,t} = Y_T - Y_0 - \lambda \sum_{t<T} c_t. \tag{4}$$

Consequently, an intermediate repair cannot receive repeated credit for remaining correct. With remaining operations b, remaining decisions h, feasible actions $\mathcal{A}_b(S)$, and $V^*(S, b, 0) = 0$,

$$Q^*_\lambda(S, b, h, a) = \mathbb{E}[Y_1 - Y_0 - \lambda c + V^*_\lambda(S', b - c, h - 1) \mid S, a], \tag{5}$$

$$V^*_\lambda(S, b, h) = \max\{0, \max_{a \in \mathcal{A}_b(S)} Q^*_\lambda(S, b, h, a)\}. \tag{6}$$

STOP has zero additional value at an already scored decision. Initialization pays the first controller call, continuation pays the next call whenever a next decision exists, and every call, including one that selects STOP, is charged once.

Fitted backups proceed in increasing h with fixed target heads and a Huber loss; at $h = 1$, a weight-0.1 consistency term aligns Q with $\hat{g} - \lambda\hat{c}$. Budget relabeling changes compatible resource inputs but not the generator's (x, prefix, ℓ) distribution, and the next-controller charge is recomputed for terminal transitions. Online, one batched pass scores all feasible actions; the system stops if their maximum value is nonpositive and otherwise executes the maximizer. Hard feasibility first reserves the current controller call and then bounds intervention and next-decision costs, whereas value targets use incurred cost. After calibration, shared features are frozen and only the linear value readout is refit, preserving the calibrated probability heads. No online speculative grid expansion occurs.

Learning action contrasts. An additional fit reuses existing same-parent outcomes to isolate action ordering. For A actions, let d_a denote the mean sampled $Y_1 - Y_0$ and $C_A = \mathrm{Id} - \mathbf{1}\mathbf{1}^\top/A$. We add

$$\mathcal{L}_{\mathrm{ctr}} = A^{-1}\|C_A\hat{\boldsymbol{g}} - C_A\boldsymbol{d}\|_2^2 \tag{7}$$

with weight 0.2. Centering cancels any common parent offset, including Y_0, without additional generator calls, and Proposition 2 shows that its population optimum preserves true action contrasts. Absolute-value losses remain necessary for STOP. The centered objective is evaluated separately from the primary controller, which uses absolute-value supervision without the contrast loss.

5 Theory: when path information has decision value

The theoretical statements use the protocol assumptions in Appendix A: A1 fixes intervention and cost semantics, A2 supplies common-support randomized assignment, A3 gives a finite full-history control problem with charged computation, and A4 is invoked only for the conditional value-error guarantee. Under A1–A3, the repair–damage surface is identified from same-parent outcomes; under A4, its approximation error translates into a finite-horizon control bound. The theory yields three testable predictions: selector exchange should change the preferred action while leaving candidate coverage fixed; path-aware policies should gain only when action optima conflict within coarse strata; and centered fitting should improve within-parent ordering while absolute supervision remains necessary for STOP.

Proposition 1 (Selection gates the value of generation)**.** *Suppose ρ selects the highest fixed per-candidate score $s(x, z)$, breaking ties in favor of the incumbent with score M. For a new candidate of correctness Y_a, let $K_a = \mathbf{1}\{s(x, Z_a) > M\}$, and $K_a = 0$ for invalid generation. Then*

$$g(I, a) = \mathbb{E}[(Y_a - Y_0)K_a \mid I, \mathrm{do}(a)]. \tag{8}$$

A strictly increasing transformation of all selector scores preserves this gain for a fixed parent/candidate distribution. Changing score ordering *can reverse optimal actions while generated-candidate coverage stays identical.*

An unselected candidate changes nothing, whereas a selected candidate replaces Y_0 by Y_a. This identity separates the target from continuation success and step advantage: the usefulness of an action depends on which correct and incorrect candidates cross the incumbent's selection threshold, not on candidate correctness alone. Appendix B.1 factors repair and damage through this gate and constructs an exact action reversal.

Theorem 1 (Path information dividend and estimation regret)**.** *Let $D = f(I)$ be a coarser question/resource/confidence statistic, with a common finite feasible action set measurable from D and STOP included. Define $u(I, a) = g(I, a) - \lambda \bar{c}(I, a)$ and*

$$V_I = \mathbb{E} \max_a u(I, a), \quad V_D = \mathbb{E} \max_a \mathbb{E}[u(I, a) \mid D], \quad \Delta_{I:D} = V_I - V_D. \tag{9}$$

Then $\Delta_{I:D} \geq 0$, with equality exactly when a common maximizing action exists almost surely within almost every D stratum. For a learned I-policy $\hat{\pi}$,

$$J(\hat{\pi}) - V_D = \Delta_{I:D} - \mathcal{R}(\hat{\pi}), \qquad \mathcal{R}(\hat{\pi}) = \mathbb{E}[\max_a u(I, a) - u(I, \hat{\pi}(I))]. \tag{10}$$

If $\|\hat{u} - u\|_\infty \leq \varepsilon$ and $\hat{\pi}$ is greedy, $\mathcal{R} \leq 2\varepsilon$. Thus $\Delta_{I:D} > 2\varepsilon$ suffices to improve on the best D-only policy, including stopping.

The theorem yields a two-part decision criterion: path information must change the best action, and the controller must recover more of that value than it loses to estimation regret (proof in Appendix B.2). The equality condition motivates within-question action-rank tests and the regret term motivates centered fitting; pooled correctness AUROC establishes neither requirement.

Structural consequences. Appendix B further establishes the decomposition $g = \mathbb{E}\Delta U - \mathbb{E}\Delta L_\rho$, a finite-budget blocked-prefix bound, and Theorem 2: when safe prefixes preserve success and deeper retention reduces cost, the latest prefix that removes the blocker is optimal. Theorem 3 bounds H-step utility regret by $2H\varepsilon$ under uniform optimal-Q error, and a margin refinement bounds one-step regret by $2\varepsilon\mathbb{P}(\Gamma \leq 2\varepsilon)$ for unique-optimum gap Γ. None of these results assumes a single irreversible error per trace or a Markov compressed representation.

6 Evaluation setting

Breadth. The evaluation spans Qwen3 4B/8B/14B/32B, Qwen3.5 9B/27B, Qwen3.6-27B, Qwen3.8-27B, R1-Distill-Qwen 7B/14B/32B, Gemma 3 12B/27B, and Llama 3.1 8B (Yang et al., 2025; Qwen Team, 2026a;b;c; DeepSeek-AI et al., 2025; Gemma Team et al., 2025; Grattafiori et al., 2024). It covers GSM8K, MATH-500, AMC23, AIME24/25, Minerva Math, OlympiadBench, HumanEval+, MBPP+, LiveCodeBench, GPQA Diamond, and MMLU-Pro (Cobbe et al., 2021; Hendrycks et al., 2021b; Lewkowycz et al., 2022; He et al., 2024; Liu et al., 2023; Jain et al., 2024; Rein et al., 2023; Wang et al., 2024); Appendix D fixes the slices and model provenance.

Common comparison. Initial answers use 4,096 tokens, temperature 0.6, and top-p 0.95. All multi-candidate methods share the initial attempts, a frozen Qwen3-0.6B outcome selector, completion rules, and hard limits of six reopenings, 16,384 generated tokens, and $16C_0$ operations, where C_0 is the model–task cost of one attempt plus selection; the headline operating point is $2.4C_0$. Policies are chosen on validation questions, and process-scoring overhead is charged. Longer-initial-budget controls raise the token cap separately.

Comparators. The comparison set comprises long-chain reasoning, BoN, root-adaptive, question-only and confidence-only controllers, PAV, VGB, Subpool, and PB-SMC, grouped into matched *search-access*, *information*, and *objective* controls. Every adaptation uses the common selector and cost ledger (Appendix E). The search–reward factorial holds horizon and architecture fixed, and the centered extension reuses its counterpart's generated supervision and inference allowance.

7 Results

7.1 Accuracy and compute efficiency across models and tasks

Table 1 reports 144 task scores whose generators, from Llama-3.1-8B to Qwen3.8-27B, span both large headroom and near saturation. On Qwen3-8B, RAVEC reaches 73.18% core macro accuracy, compared with 71.90% for PB-SMC and 70.98% for BoN (Table 32). Its gains over PB-SMC

Table 1: Selected accuracy (%) at $2.4C_0$ (Single: C_0) for six generators from four model families on all eight core benchmarks. Bold marks the better of PB-SMC and RAVEC within each model–task pair; RAVEC attains the higher core macro accuracy for every listed generator. The appendix reports the complete 14-model × 12-task × 4-method matrix and the 12-method Qwen3-8B comparison. †: harmonized adaptation.

Model	Method	M500	A24	A25	Olym.	HE+	MB+	LCB	GPQA	Core
Qwen3-4B	Single	93.1	61.4	54.8	49.5	80.6	68.9	45.6	51.1	63.13
	PB-SMC†	97.2	68.1	62.5	55.0	83.1	**75.0**	51.2	**53.8**	68.24
	RAVEC	**98.6**	**71.9**	**66.0**	**58.3**	**83.2**	74.9	**54.8**	53.3	**70.13**
Qwen3-8B	Single	94.6	65.8	59.1	56.7	86.2	75.8	49.4	56.7	68.04
	PB-SMC†	97.3	70.9	65.5	60.8	88.3	78.8	54.4	**59.2**	71.90
	RAVEC	**98.8**	**73.0**	**67.3**	**62.5**	**88.7**	**80.0**	**56.4**	58.7	**73.18**
Qwen3.8-27B	Single	97.7	83.6	78.9	79.3	96.8	92.2	79.7	83.8	86.50
	PB-SMC†	97.8	86.8	82.1	81.3	**97.2**	**94.0**	82.5	85.4	88.39
	RAVEC	**99.3**	**89.3**	**84.2**	**84.2**	97.0	93.7	**85.0**	**85.5**	**89.78**
R1-Distill-14B	Single	91.9	61.4	40.8	52.1	86.5	75.0	47.7	54.6	63.75
	PB-SMC†	95.5	**68.7**	47.9	57.1	88.6	**79.3**	53.0	**58.3**	68.55
	RAVEC	**97.4**	68.5	**52.1**	**60.6**	**90.6**	78.7	**56.6**	58.2	**70.34**
Gemma-3-12B	Single	82.3	18.6	15.4	34.6	79.5	66.1	26.9	40.6	45.50
	PB-SMC†	87.2	27.2	25.4	40.5	82.4	70.8	**36.2**	44.4	51.76
	RAVEC	**90.1**	**28.1**	**26.1**	**43.2**	**83.1**	**73.3**	35.7	**46.2**	**53.23**
Llama-3.1-8B	Single	50.2	6.4	3.1	16.8	69.8	56.8	16.7	31.5	31.41
	PB-SMC†	56.9	12.5	8.1	22.3	**76.8**	**62.7**	**26.1**	34.0	37.43
	RAVEC	**59.4**	**16.0**	**11.0**	**25.1**	76.6	62.4	25.9	**35.7**	**39.01**

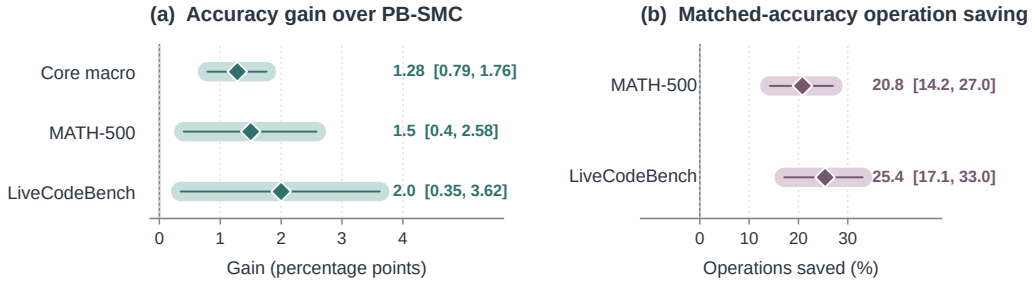


Figure 2: Uncertainty of the Qwen3-8B results. Diamonds mark point estimates and rounded bands the nested 95% intervals for (left) the accuracy gain of RAVEC over PB-SMC and (right) the matched-accuracy operation saving. The protocol uses 64 paired decoding replicates per question–method cell, question-cluster resampling, and five independently initialized controller fits.

are 1.5 points on MATH-500, 1.7 on OlympiadBench, and 2.0 on LiveCodeBench, whereas HumanEval+ gains only 0.4 point and GPQA decreases from 59.2 to 58.7; the gain is therefore task-dependent. The 1.28-point core gain has interval $[0.79, 1.76]$, with $1.5\,[0.40, 2.58]$ on MATH-500 and $2.0\,[0.35, 3.62]$ on LiveCodeBench, and fit-level deviations of 0.16, 0.18, and 0.22 points.

The improvement generalizes across all 14 models: the core gain over PB-SMC averages 1.34 points (median 1.36), ranges from 0.86 to 1.89 points, reaches 1.39 on Qwen3.8-27B, and has nested model-level intervals excluding zero for 12 of the 14 models (Table 22); the complete 168-cell-per-method matrix appears in Appendix M. Excluding both AIME sets leaves Qwen3-8B a 1.05-point gain, and Table 21 and appendix sweeps vary aggregation, selector capacity, collection size, training domain, decoding temperature, initial budget, replay cost, and latency, with cost always normalized within model and task.

The accuracy gains translate into computation savings. Interpolating the operation frontiers (Figure 3) at 97.0% MATH-500 accuracy gives $1.78C_0$ for RAVEC versus $2.25C_0$ for PB-SMC, a 20.8% saving with interval $[14.2, 27.0]\%$; at 55.0% LiveCodeBench accuracy, the corresponding costs are $2.09C_0$ and $2.80C_0$, a 25.4% saving with interval $[17.1, 33.0]\%$. Each bootstrap draw rebuilds the paired accuracy–cost curves and repeats monotone interpolation, and both comparisons lie within overlapping curve ranges. To test whether truncated initial thinking creates all of the room for repair, a longer initial cap is also evaluated: Qwen3-8B then reaches 98.9% on MATH-500 and 60.6% on LiveCodeBench versus 97.8% and 59.4%, and Qwen3.8 retains a 1.1-point coding gain (Table 20).

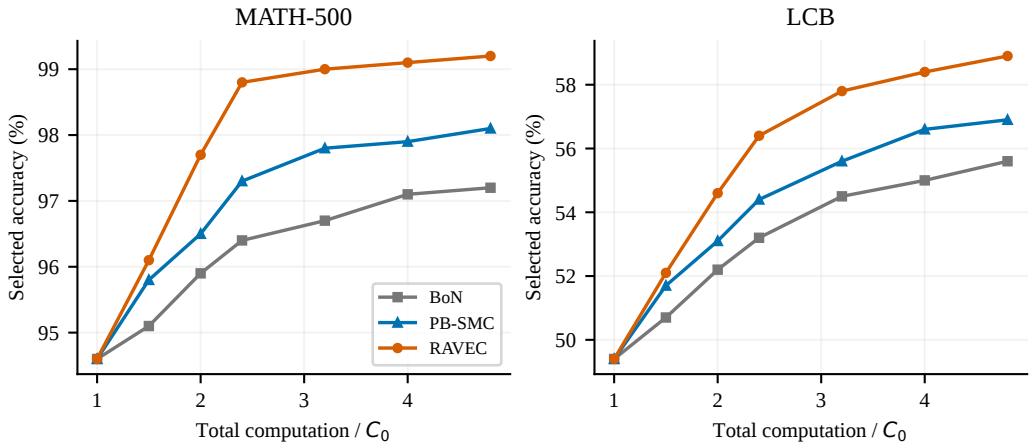


Figure 3: Accuracy versus complete operation cost, including initial generation and selection/control costs. Lines interpolate the operating points; at matched accuracy, RAVEC requires 20.8% fewer operations than PB-SMC on MATH-500 and 25.4% fewer on LiveCodeBench. Latency sensitivity is analyzed separately in Appendix K.6.

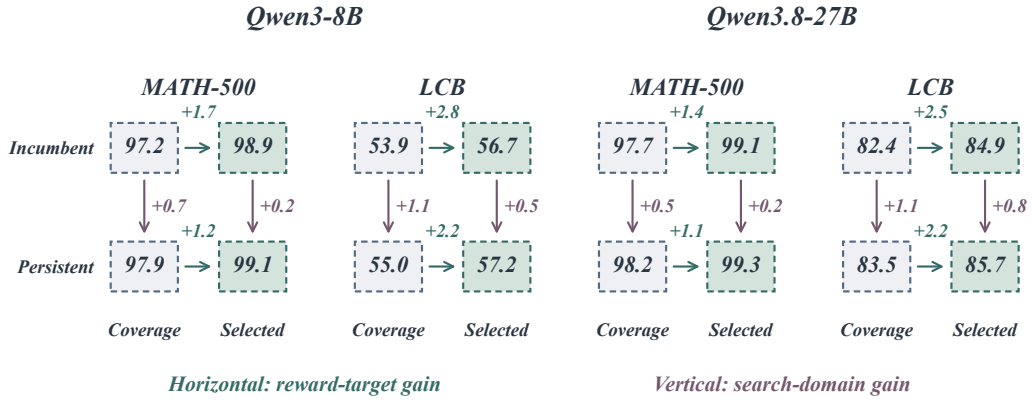


Figure 4: Search–reward interaction at $2.4C_0$. Nodes report selected accuracy (%); horizontal and vertical arrows show changes in percentage points. Replacing the coverage reward with the selected-correctness reward improves accuracy in both the incumbent and persistent search domains. Both targets share six-step backups, architecture, selector, and collection allowance; Persistent/Selected is fitted separately.

7.2 Controlled ablations: search domain and learned target

Figure 4 isolates Contribution 1 by crossing incumbent-only versus persistent-prefix search with coverage versus selected-correctness rewards under the same six-step architecture and backups. For Qwen3-8B, switching to the selected-correctness reward raises MATH-500/LiveCodeBench accuracy from 97.2/53.9 to 98.9/56.7 in the incumbent domain and from 97.9/55.0 to 99.1/57.2 in the persistent domain. The learned target thus improves both search domains, while historical prefixes supply an additional increment, and Qwen3.8 shows the same pattern at smaller magnitudes. Selection-conditioned value therefore complements richer search rather than depending on a weak search comparator.

All factorial cells share equal offline collection operations within a generator, a common union of collection parents, and the same selector; coverage labels are used only in training, and evaluation always returns the actual selected answer (Appendix J.1). Expanded-search controls test sensitivity to search configuration; wider PB-SMC exceeds RAVEC on AIME24 (74.4% versus 73.0%; Table 5). Beyond this crossed design, Table 12 audits 14 matched-cost component variants and one fixed-policy replay-cost check, including the damage term, direct-ΔY supervision, greedy versus

finite-horizon value, root-only and random positions, randomized collection, reached-state aggregation, calibration, and replay accounting.

7.3 Natural-parent controls distinguish reuse from location

The natural-parent controls test Contribution 2 on 512 held-out math and 512 code questions with two naturally sampled, unfiltered parents each. With fixed 1,024-token suffix caps, learned positions outperform root reopening by 2.9 and 3.1 points, a comparison that includes the benefit of retained work. Under a common total-operation envelope in which root receives the longest suffix it can afford, the gains remain 2.2 and 2.5 points. A root-only value controller reaches 83.2/59.5%, versus 84.9/61.4% for one-step RAVEC; it receives the same trace, pool, confidence, and resource inputs, supervision, network width, and STOP rule, and differs only in being restricted to root positions. The resulting 1.7/1.9-point differences therefore isolate the value of choosing a non-root position within this action grid, and they remain 1.8/2.0 points under full replay (Table 17), which rules out a free-cache explanation.

Doubling root's *additional* allowance yields 84.8/61.5%, within 0.1 point of one-step RAVEC on both tasks. RAVEC thus matches the tested root configuration to within 0.1 point using half its additional allowance, demonstrating efficient allocation through learned reopening positions. Artificial blocker-location surfaces are reported separately (Figure 6).

7.4 The selector changes which computation is valuable

Rescoring the same candidate bank with a 4B instead of a 0.6B ORM changes the preferred position for 23.1% of diagnostic parents, and refitting position choice regains 1.7 math and 2.4 code points over retaining the old choices. Because the generated candidates and their coverage are identical, only the selection map changes, which is the reversal predicted by Proposition 1; a strictly monotone score-temperature transform correspondingly yields exactly zero rank reversals and zero output gain. The diagnostic is generator-free, with selector operations charged separately (Table 18).

The endpoint decomposition is consistent with the primary table: RAVEC's 4.2-point MATH-500 gain over Single equals 4.53 repair points minus 0.33 damage points, and its 7.0-point LiveCodeBench gain equals 7.84 minus 0.84. Coverage-only training reaches 97.2% and 53.9%, and removing the damage penalty reaches 97.8% and 55.0% (Table 12), both below the full target, because generating more correct candidates differs from selecting more correct outputs.

Appendix K provides the corresponding evidence bridge. Against PB-SMC, the math output gain decomposes into 0.4 coverage points plus 1.1 points of reduced selection loss, and the code gain into 1.1 plus 0.9. In the action diagnostic, RAVEC recovers 6.5 of the 8.7-point math gap between confidence/length ranking and the finite reference. Table 25 further factors natural-parent repair and damage through candidate correctness and selector acceptance; on code, better repair coexists with slightly more damage.

8 Discussion and conclusion

RAVEC learns which affordable prefix intervention improves the answer that is ultimately selected. Selection-conditioned repair and damage target the selected answer rather than generation, and path-informed control chooses position, budget, and stopping. Across 14 models and 12 benchmarks, the core gain over adapted PB-SMC averages 1.34 points, and Qwen3-8B requires 20.8% and 25.4% fewer operations at matched math and code accuracy; historical-prefix search and parent-centered fitting (Appendix J.4) are separate extensions. The controller uses a complete initial answer and generator- and selector-specific values over a finite action grid; deployment with a new generator or selector calls for refitting. Under the overhead specification in Table 29, operation savings become latency savings when the effective service rate remains above 80.8% of PB-SMC's on math and 76.1% on code; Table 30 accounts for incremental offline cost. The matched-feature root and fixed-bank selector controls identify complementary roles for computation placement and selection-aware valuation. Within these conditions, path information improves control when it changes optimal actions, and the learned controller recovers enough of this value to outweigh estimation regret and computation cost.

AI USE STATEMENT

AI-assisted tools supported language editing, mathematical exposition, reference-metadata retrieval, implementation of the controller and training/evaluation scripts, and generation of tables and figures. The authors determined the research questions, technical claims, experimental design, quantitative analysis, and final manuscript content. The authors reviewed all AI-assisted work and take responsibility for the final content of this paper, including AI-assisted text, code, tables, and figures.

ETHICS STATEMENT

RAVEC studies computation allocation for language-model reasoning. More efficient inference can broaden access to reasoning systems, while retaining the underlying models' risks of incorrect, biased, or harmful outputs. Deployment should preserve model safeguards and apply task-appropriate validation; generated code should be evaluated in an isolated execution environment. Use of model weights and benchmark data remains subject to their original licenses and access conditions.

REPRODUCIBILITY STATEMENT

Section 4 specifies the learning and control procedure, and Appendices B–E provide proofs, implementation details, benchmark definitions, and comparison protocols. Appendix I details computation accounting. The supplementary package includes the controller implementation, training and evaluation scripts, and configurations. Model weights and benchmark inputs must be obtained from their original sources.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, et al. Program Synthesis with Large Language Models, 2021. URL https://arxiv.org/abs/2108.07732.

Yigit Utku Bulut. It's the problem, not the path: Budget and difficulty confounds in llm reasoning trajectories, 2026. URL https://arxiv.org/abs/2609.03436.

Frederick Callaway, Sayan Gul, Paul M. Krueger, Thomas L. Griffiths, and Falk Lieder. Learning to select computations, 2017. URL https://arxiv.org/abs/1711.06892.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, et al. Evaluating Large Language Models Trained on Code, 2021. URL https://arxiv.org/abs/2107.03374.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3=? on the overthinking of o1-like llms, 2024. URL https://arxiv.org/abs/2412.21187.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, et al. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168.

C. Nicolò De Sabbata, Theodore R. Sumers, Badr AlKhamissi, Antoine Bosselut, and Thomas L. Griffiths. Rational metareasoning for large language models, 2024. URL https://arxiv.org/abs/2410.05563.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025. URL https://arxiv.org/abs/2501.12948.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, et al. Gemma 3 Technical Report, 2025. URL https://arxiv.org/abs/2503.19786.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, et al. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017. URL https://arxiv.org/abs/1706.04599.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, et al. OlympiadBench: A Challenging Benchmark for Promoting AGI with Olympiad-Level Bilingual Multimodal Scientific Problems, 2024. URL https://arxiv.org/abs/2402.14008.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, et al. Measuring Coding Challenge Competence With APPS, 2021a. URL https://arxiv.org/abs/2105.09938.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset, 2021b. URL https://arxiv.org/abs/2103.03874.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Naman Jain, King Han, 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, 2024. URL https://arxiv.org/abs/2403.07974.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, et al. Solving Quantitative Reasoning Problems with Language Models, 2022. URL https://arxiv.org/abs/2206.14858.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, et al. TACO: Topics in Algorithmic COde generation dataset, 2023. URL https://arxiv.org/abs/2312.14852.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023. URL https://arxiv.org/abs/2305.01210.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017. URL https://arxiv.org/abs/1711.05101.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Fei-Fei Li, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Tian Qin, David Alvarez-Melis, Samy Jelassi, and Eran Malach. To backtrack or not to backtrack: When sequential search limits model reasoning, 2025. URL https://arxiv.org/abs/2504.07052.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-27b.

Qwen Team. Qwen3.8-27B model card, August 2026c. URL https://huggingface.co/Qwen/Qwen3.8-27B.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Navid Rezazadeh and Arash Gholami Davoodi. The shape of overthinking: Backtracking bursts in long reasoning traces, 2026. URL https://arxiv.org/abs/2605.27965.

Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan J. Foster. Taming imperfect process verifiers: A sampling perspective on backtracking, 2025. URL https://arxiv.org/abs/2510.03149.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for llm reasoning, 2024. URL https://arxiv.org/abs/2410.08146.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Dao Tran, Duc Anh Le, Ngoc Luu, Quan Pham, Tung Pham, and Hung Bui. Beyond the frontier: Stochastic backtracking for efficient test-time scaling, 2026. URL https://arxiv.org/abs/2605.25143.

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, 2022. URL https://arxiv.org/abs/2203.11171.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, et al. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, et al. Qwen3 Technical Report, 2025. URL https://arxiv.org/abs/2505.09388.

The appendix is organized around the three contributions. Notation and proofs (Appendices A–B) formalize the selection-conditioned target (Contribution 1) and path-informed control (Contribution 2). Implementation, benchmark, and baseline specifications (Appendices C–E) and controls and computation accounting (Appendices F–I) fix the evaluation protocol (Contribution 3). Mechanism comparisons (Appendices J–K) link contributions to controlled evidence; Appendix M collects complete results.

A Notation and assumptions

Table 2: Notation used throughout the theory, controller specification, and evaluation.

Symbol	Meaning
x, θ, ρ	Problem, frozen generator, frozen terminal selector.
$S, I = \psi(S)$	Full observed history state and its compressed controller representation.
$\mathcal{Z}, \mathcal{P}, \mathcal{H}$	Candidate pool, checkpoints, action history.
$Y(S), U(\mathcal{Z}), L_\rho$	Selected correctness, oracle pool coverage, and selection loss $U - Y$.
$a = (j, \ell)$	Reopen at retained checkpoint j with suffix token cap ℓ.
b, h, λ	Remaining operation budget, remaining decisions, computation price.
r, α, β	Current reliability, conditional repair probability, conditional damage probability.
$g, \bar{c}$	Expected correctness change and expected operation cost of an intervention.
Q^*, V^*	Optimal remaining net utility; STOP has value zero.
C_0	Expected single-attempt operation count for a specified model–task pair.

A1–A2 make the target of Contribution 1 identifiable, A3 defines the charged control problem of Contribution 2, and A4 enters only the conditional regret guarantee.

A1: well-defined interventions. Each action fully specifies the retained token prefix, generation distribution, termination rule, candidate-pool update, and frozen selector. Conditional on the same parent, repeated samples draw generation randomness independently, and cloned branches do not interfere. The generator and selector remain fixed throughout data collection for a given controller.

A2: supported randomized assignment. For an action comparison in stratum I, either every parent in the target stratum admits that action or the comparison is restricted to the common eligible population. Assignment follows a recorded $\mu(a \mid S) > 0$ and, conditional on S, is independent of potential outcomes. Mandatory coverage of all actions prevents the training set from becoming a PRM-selected subset. When a parent has fewer distinct legal positions, duplicated positions are not counted as additional evidence.

A3: observable costs and finite control. Costs are nonnegative, and their hard upper bounds determine admissibility. The full history constitutes a valid finite-horizon decision state with $H \leq 6$. The reward and the budget ledger apply the same convention to controller and selector charges. No Markov property is imposed on the compressed representation I.

A4: approximation condition, when invoked. The control theorem states performance under a uniform error bound relative to the *optimal* full-history Q^*. This explicit condition isolates the approximation requirement for the guarantee, while the later statistical bounds quantify estimation error for repeated actions at fixed parents.

B Proofs and additional theoretical results

B.1 Selection gates and an exact action-order reversal

Assumptions and proof of Proposition 1. This subsection gives the formal basis of Contribution 1. The selector scores each candidate separately with a fixed function, retains the incumbent on score ties, and rejects an invalid new candidate; pool-dependent rescoring and randomized tie breaking are thus excluded. Every realized intervention then has one of two outcomes: the incumbent survives ($K_a = 0$) or the new candidate becomes the selected answer ($K_a = 1$). Consequently,

$$Y_1 = (1 - K_a)Y_0 + K_a Y_a, \qquad Y_1 - Y_0 = K_a(Y_a - Y_0). \tag{11}$$

Taking the conditional interventional expectation proves Eq. (8). If s is replaced by a strictly increasing function $f(s)$, then $f(s_a) > f(M)$ exactly when $s_a > M$, including under the original tie rule, so every selected output in the fixed pool is unchanged. This invariance concerns selection among the same candidates only; a controller that reads absolute scores can still change its online actions after an uncalibrated score transformation, an effect outside the proposition. The general selected-gain identity in Proposition 3 remains valid even when the maximum-score assumption fails.

To separate candidate correctness from selector acceptance, let

$$q_0 = \mathbb{P}(Y_a = 1 \mid Y_0 = 0, I, \mathrm{do}(a)), \qquad \kappa_+ = \mathbb{P}(K_a = 1 \mid Y_a = 1, Y_0 = 0, I, \mathrm{do}(a)), \tag{12}$$

$$q_1 = \mathbb{P}(Y_a = 1 \mid Y_0 = 1, I, \mathrm{do}(a)), \qquad \kappa_- = \mathbb{P}(K_a = 1 \mid Y_a = 0, Y_0 = 1, I, \mathrm{do}(a)). \tag{13}$$

Thus q_0, q_1 describe generation, whereas κ_+, κ_- describe acceptance of a correct repair or an incorrect replacement. Gates on zero-probability events can be assigned arbitrarily because their factors vanish. Then

$$\alpha = q_0\kappa_+, \qquad \beta = (1 - q_1)\kappa_-, \qquad g = (1 - r)q_0\kappa_+ - r(1 - q_1)\kappa_-. \tag{14}$$

Candidate correctness identifies only q_0, q_1, not the acceptance factors, so changing the terminal selector can alter the optimal action without changing either correctness probability. Pool coverage is likewise insufficient when correct candidates can remain unselected. The repair and damage heads therefore target α and β, which include both factors.

Constructive example. The following construction realizes the reversal in Proposition 1. Consider equal-cost Early and Late interventions from a one-candidate initial pool with $r = 0.75$, with selector-independent generation quantities $(q_0, q_1) = (0.60, 0.70)$ for Early and $(0.45, 0.65)$ for Late. Under a weak selector, their (α, β) pairs are $(0.30, 0.055)$ and $(0.36, 0.015)$; under a stronger selector, $(0.55, 0.012)$ and $(0.40, 0.010)$. All pairs are feasible: $0 \le \alpha \le q_0$ and $0 \le \beta \le 1 - q_1$. Only the gates change; Early's repair gate rises from $0.30/0.60 = 0.50$ to $0.55/0.60 = 11/12$, while its damage gate falls from $0.055/0.30 = 11/60$ to $0.012/0.30 = 0.04$.

An explicit joint construction assigns each correctness-conditioned candidate type an independent uniform latent variable and places its score above the incumbent's precisely on an interval of length κ_+ or κ_-. Scores can be $+1$ or -1 with incumbent score zero, and encoding the latent types in candidate text keeps both selectors fixed per-candidate functions. No correctness label enters the scoring function; correctness and score are jointly distributed properties of that text. Both selectors act on identical candidate draws, and their gains are

$$(g_E^{\mathrm{weak}}, g_L^{\mathrm{weak}}) = (0.03375, 0.07875), \qquad (g_E^{\mathrm{strong}}, g_L^{\mathrm{strong}}) = (0.12850, 0.09250). \tag{15}$$

Late wins under the weak selector and Early under the stronger one, whereas coverage, $r + (1-r)q_0$, is selector-independent: 0.90 for Early and 0.8625 for Late. A coverage target thus prefers Early under both selectors and misorders the actions under the weak one. Table 19 and Figure 8 visualize this analytical construction, which establishes an exact reversal at fixed candidate quality; the fixed-bank selector-exchange experiment separately measures reversals in natural parent states.

B.2 Information value, equality conditions, and margin refinements

Proof of Theorem 1. This subsection gives the formal basis of Contribution 2. Assume each $u(I, a)$ is integrable and the common feasible action set within a D stratum is finite. When eligibility varies across parents, D includes the legal-action mask, so compared policies share the same executable options; otherwise a feasibility difference would be mistaken for informational value. Fix an ordering that resolves maximizing ties measurably, and let

$$a_D \in \arg\max_a \mathbb{E}[u(I, a) \mid D], \qquad a_I \in \arg\max_a u(I, a). \tag{16}$$

The value of any D-measurable randomized policy is a convex combination of conditional action values and cannot exceed that of a_D, so V_D is the best attainable coarse-policy value. By the tower property,

$$V_I - V_D = \mathbb{E}\big[\max_a u(I, a) - u(I, a_D)\big] \ge 0. \tag{17}$$

Since the integrand is nonnegative, its mean is zero if and only if a_D is a maximizing action for conditional-almost-every I within almost every D stratum. Conversely, if an action maximizes for

conditional-almost-every I in a stratum, finiteness permits choosing it measurably, and it attains both the coarse and fine conditional values. This proves the equality characterization, including nonunique optima. Different actions among tied maxima therefore do not establish a positive dividend.

For $J(\hat{\pi}) = \mathbb{E}u(I, \hat{\pi}(I))$, adding and subtracting V_I gives Eq. (10) exactly: the advantage over the best coarse policy equals the dividend minus estimation regret. If $\hat{\pi}$ maximizes $\hat{u}$ and $\sup_{I,a} |\hat{u} - u| \leq \varepsilon$, then pointwise

$$u(I, a_I) - u(I, \hat{\pi}(I)) \leq [u(I, a_I) - \hat{u}(I, a_I)] + [\hat{u}(I, \hat{\pi}(I)) - u(I, \hat{\pi}(I))] \tag{18}$$

$$\leq 2\varepsilon. \tag{19}$$

The omitted middle greedy term is nonpositive. Taking expectations proves the regret bound and the strict-advantage condition. Treating STOP as an exact-zero action is compatible with this argument; its availability makes every optimal additional value nonnegative, although an imperfect learned policy need not attain nonnegative true utility.

Corollary 1 (A positive dividend from conflicting optima)**.** *Within a fixed D stratum, suppose two types with probabilities p and $1 - p$ have distinct unique optimal actions a_1, a_2. On each type, its optimum exceeds every other action by at least $m > 0$. Then the conditional information dividend is at least $m \min\{p, 1 - p\}$.*

Proof. A coarse policy choosing a_1 loses at least m on type 2, one choosing a_2 loses at least m on type 1, and any other action loses at least m on both. Every fixed action thus has expected regret at least $m \min\{p, 1 - p\}$, and randomization cannot improve on the best fixed action. Equation (17) completes the proof. □

Here I is pre-action information shared across the action grid; revealing the proposed action reveals no post-intervention outcome. A restricted head implementation need not use all of I.

As an instance of Corollary 1, two equally frequent path types with utilities $(0.12, 0.02)$ and $(0.02, 0.12)$ for two actions, and STOP utility zero, have $V_I = 0.12$, $V_D = 0.07$, and dividend 0.05; this exact construction is independent of any performance table. Confidence can be identical on the two types, so a confidence-only allocator captures none of this dividend: the useful information is which action succeeds on the current path, not which parent is globally easier, which is the distinction behind Challenge 2.

Corollary 2 (Action margins reduce estimation regret)**.** *Suppose the optimal action is unique almost surely and let $\Gamma(I) > 0$ be its utility gap to the second-best feasible action. Under the uniform error assumption above,*

$$\mathcal{R}(\hat{\pi}) \leq 2\varepsilon \mathbb{P}(\Gamma \leq 2\varepsilon). \tag{20}$$

If $\mathbb{P}(\Gamma \leq t) \leq Kt^{\kappa}$ for $0 < t \leq t_0$ and $2\varepsilon \leq t_0$, then $\mathcal{R}(\hat{\pi}) \leq K(2\varepsilon)^{\kappa+1}$.

Proof. When $\Gamma > 2\varepsilon$, the true best action remains strictly better under every allowed perturbation, so the greedy action is correct with zero regret. Otherwise, the preceding proof bounds regret by 2ε. Integrating gives the first inequality, and substituting the assumed margin distribution gives the second. For a single feasible action, set $\Gamma = +\infty$. □

These are single-decision statements about expected utility conditional on I. In the full-history multistep problem, the optimal action also values future information and interventions; Theorem 3 supplies the separate $2H\varepsilon$ guarantee for optimal-Q approximation. Neither an action-ranking correlation nor the diagnostic regret in Figure 11 proves the required uniform bound, so the connection to practice is directional: within-parent action ordering is a relevant learning objective, whereas pooled outcome prediction can improve without changing any decision. This motivates the centered fit below.

B.3 Why a parent-centered loss learns action contrasts

Proposition 2 (Population target of centered fitting)**.** *Fix a parent S with $A \geq 2$ eligible actions. Let $\boldsymbol{g}_S$ be their true mean selected-correctness changes and let $\boldsymbol{d}$ be unbiased sampled means, with*

conditional covariance Σ_S. For a fixed prediction vector $\hat{\boldsymbol{g}}$ and $C_A = \mathrm{Id} - \mathbf{1}\mathbf{1}^\top/A$,

$$\mathbb{E}[\mathcal{L}_{\mathrm{ctr}} \mid S] = \frac{\|C_A(\hat{\boldsymbol{g}} - \boldsymbol{g}_S)\|_2^2}{A} + \frac{\mathrm{tr}(C_A \Sigma_S C_A)}{A}. \tag{21}$$

The loss identifies all pairwise action contrasts and leaves a common additive offset unidentified. Independence between action labels is unnecessary.

Proof. Write $\boldsymbol{d} = \boldsymbol{g}_S + \boldsymbol{\xi}$, where $\mathbb{E}[\boldsymbol{\xi} \mid S] = 0$ and $\mathbb{E}[\boldsymbol{\xi}\boldsymbol{\xi}^\top \mid S] = \Sigma_S$. Expanding the squared norm eliminates the mean-zero cross term, and the remaining noise term equals $\mathrm{tr}(C_A \Sigma_S C_A)$ because $C_A = C_A^\top = C_A^2$. Moreover, for any vector $\boldsymbol{e}$,

$$\frac{1}{A}\|C_A \boldsymbol{e}\|_2^2 = \frac{1}{2A^2}\sum_{a,b}(e_a - e_b)^2. \tag{22}$$

Applying this identity to $\boldsymbol{e} = \hat{\boldsymbol{g}} - \boldsymbol{d}$ shows that the loss is a normalized average of pairwise contrast errors. Its noise-free component vanishes exactly when predictions and true gains differ by a common parent constant, which identifies contrasts but not the offset. □

The proposition supports the parent-centered refinement of Contribution 2, reported separately from the primary controller. It is a population-risk statement with parameters fixed before a fresh target is drawn, not an in-sample noise formula for a model trained on those draws. If the predictor observes only $I = \psi(S)$ and I determines the eligible action set, averaging Eq. (21) over $S \mid I$ adds a prediction-independent representation-variance term, and the optimal centered prediction is $C_A \mathbb{E}[\boldsymbol{g}_S \mid I] = C_A \boldsymbol{g}(I)$. Otherwise, one must condition further on the eligible action set rather than mix incompatible action vectors.

Because $d_a = \bar{Y}_{1,a} - Y_0$, subtracting the within-parent mean cancels Y_0 exactly, and adding any common parent offset to labels or predictions leaves the loss unchanged. The fit therefore reuses existing same-parent outcomes without additional generator calls. Centering removes the shared parent component but not action-dependent representation and sampling errors, and the covariance term Σ_S accommodates correlated labels, so no per-action variance reduction is assumed.

Because the offset is unidentified, deciding whether to act at all requires a calibrated absolute target. Two actions with gains $(-0.02, -0.04)$ and $(0.03, 0.01)$ have identical contrasts, but only the latter pair beats zero-cost STOP before other costs. We therefore retain the masked probability losses, finite-horizon value targets, and one-step consistency term. The objectives are compatible: under realizability, supported branches, and sufficiency of each head's input summary for its stated conditional probability, the true probability heads minimize the proper BCE losses and their implied gains simultaneously minimize the centered population loss. With finite capacity or imperfect optimization, the added term is a fitting preference whose benefit is tested by the separate ablation; consistency of a target does not guarantee improved test accuracy.

B.4 Structural and finite-horizon results

Proposition 3 (Selected gain and coverage loss)**.** *For any common parent distribution and feasible intervention a, Eq. (2) holds. Let $U(\mathcal{Z})$ indicate whether the pool contains a correct answer and let $L_\rho = U - Y$ be selection loss. Then*

$$g(I, a) = \mathbb{E}[\Delta U \mid I, \mathrm{do}(a)] - \mathbb{E}[\Delta L_\rho \mid I, \mathrm{do}(a)]. \tag{23}$$

Even though $\Delta U \geq 0$ for a growing pool, g may be negative.

Argument. Only wrong-to-right and right-to-wrong transitions contribute to $Y_1 - Y_0$; substituting $Y = U - L_\rho$ gives the second identity. The decomposition fixes the supervision target of Contribution 1: the selected-output change equals the coverage change minus the selection-loss change, so larger coverage can coexist with lower selected accuracy.

Proposition 4 (Finite-budget obstruction)**.** *Consider initially unsuccessful parents with a latent prefix type Z. Let $\mathbb{P}(Z = 1) = q$; on type $Z = 0$, every one of m same-prefix completions has success probability at most ϵ. The probability of generating at least one correct new candidate is at most*

$$q + (1 - q)\min\{1, m\epsilon\}. \tag{24}$$

No independence assumption is needed. The bound also upper-bounds selected correctness when the initial pool contains no correct candidate.

A union bound over the m completions caps the blocked type at $\min\{1, m\epsilon\}$; the other type contributes probability at most one. With no correct initial candidate, selected correctness requires a correct new one, so the bound transfers. For positive ϵ, the result is thus a finite-budget obstruction; unlimited sampling has a different limit. It motivates the position choice of Contribution 2.

Theorem 2 (When minimal necessary backtracking is optimal)**.** *For a currently incorrect selected output and a fixed suffix policy and cap, suppose there is a blocking index e and at least one feasible checkpoint $j < e$. Each permitted checkpoint $j < e$ yields selected success probability p; each $j \geq e$ yields at most ϵ. Among safe positions, costs c_j strictly decrease with j. For $\lambda > 0$, let $j^\dagger = \max\{j : j < e\}$. If $p - \lambda c_{j^\dagger} > \epsilon$, reopening at $j^\dagger$ strictly dominates all other positions and STOP in the one-step objective.*

Equal success p on safe prefixes makes $j^\dagger$ the cheapest safe choice, and the margin makes it dominate blocked prefixes. The assumptions make the mechanism testable: useful retained work must reduce cost, and a retained commitment must substantially restrict success. Natural traces need not satisfy equal p, monotone costs, a unique blocker, or a unimodal surface, so the theorem idealizes minimal necessary backtracking and RAVEC learns the position–budget grid without imposing these structures.

Theorem 3 (Value error to control error)**.** *For a finite-horizon full-history process with feasible actions, suppose $|\widehat{Q}_\lambda - Q^*_\lambda| \leq \varepsilon$ uniformly over reachable state–budget–horizon–action tuples. Give STOP exact value zero. The greedy policy $\hat{\pi}$ satisfies*

$$V^*_\lambda(S, b, H) - V^{\hat{\pi}}_\lambda(S, b, H) \leq 2H\varepsilon. \tag{25}$$

If its expected cost differs from the optimal policy's by at most δ_C, its expected accuracy deficit is at most $2H\varepsilon + \lambda\delta_C$.

The proof bounds the optimal-value loss of each greedy decision by 2ε and inducts on the remaining horizon; adding back $\lambda\delta_C$ gives the accuracy statement. The condition concerns Q^*, not returns under a fixed rollout policy. Representation loss, limited state coverage, and function approximation all enter ε, and the theorem does not assert that offline training attains a small uniform error; it supports Contribution 2 only as a conditional guarantee. Appendix B also gives a tighter fixed-parent concentration bound and a counterexample showing that current-answer confidence alone is not a sufficient control statistic, motivating path conditioning.

B.5 Identification under randomized prefix interventions

This subsection shows that the target of Contribution 1 is identified by the randomized same-parent collection. Let $D(a) = Y(S'_a) - Y(S)$ denote the potential change in selected correctness under a fully specified action. Under A1–A2 and common eligibility,

$$g(I, a) = \mathbb{E}[D(a) \mid I] = \mathbb{E}\left[\frac{\mathbf{1}\{A = a\}D}{\mu(a \mid S)}\middle| I\right]. \tag{26}$$

Proof. By consistency, $D = D(a)$ whenever $A = a$. Conditional randomization then implies

$$\mathbb{E}\left[\frac{\mathbf{1}\{A = a\}D}{\mu(a \mid S)}\middle| S\right] = \frac{\mu(a \mid S)}{\mu(a \mid S)}\mathbb{E}[D(a) \mid S].$$

Taking the conditional expectation over $S \mid I$ proves the result. Under all-action collection, each parent contributes directly to every eligible action mean, so inverse weighting is unnecessary. If eligibility differs across actions, conditioning on each action's eligible subset changes the estimand; without additional restrictions, it no longer identifies the same population surface. □

Applying the same argument to $\mathbf{1}\{Y_0 = 0\}Y_1$ and $\mathbf{1}\{Y_0 = 1\}(1 - Y_1)$, each divided by the corresponding stratum probability, identifies α and β wherever those denominators are positive. A single labeled parent has only one value of Y_0 and therefore cannot identify both conditional branches on its own. The unconditional action effect g thus remains the primary quantity to validate.

The identified quantity is the effect of token-prefix replacement, including the altered conditioning distribution and the retained computational work. The estimand therefore captures both semantic and length-mediated mechanisms and directly measures operational output changes. Equal suffix caps, matched total cost, and within-question feature controls answer distinct scientific questions.

B.6 Proof of Proposition 3

For binary Y_0, Y_1,

$$Y_1 - Y_0 = \mathbf{1}\{Y_0 = 0, Y_1 = 1\} - \mathbf{1}\{Y_0 = 1, Y_1 = 0\}.$$

An intervention leaves the parent correctness distribution unchanged, so the first term has expectation $(1 - r)\alpha$ and the second has expectation $r\beta$, which establishes Eq. (2). For any candidate pool, $Y = U - L_\rho$ by definition; subtracting the pre-action value from the post-action value and taking expectations yields the coverage-loss decomposition. A growing pool guarantees $U_1 \geq U_0$ but imposes no monotonicity on Y or L_ρ, so coverage gains need not become selected-answer gains (Challenge 1).

As a deductive example, take $r = 0.8$, and let action a have $(\alpha, \beta) = (0.30, 0.02)$ while a' has $(0.50, 0.12)$. Their net gains are 0.044 and 0.004, respectively. Although a' repairs more currently incorrect outputs, its additional damage outweighs its additional repair, so it has less deployable value. The construction isolates the selection damage that a repair-only target would ignore.

B.7 Why confidence alone is insufficient

Proposition 5 (An indistinguishability counterexample)**.** *There exist equally likely information states with identical current reliability and identical action costs for which every confidence-only one-step policy has expected correctness gain at most zero, while a state-aware policy gains* 0.29.

Proof. Let both types have $r = 1/2$. In type A, the two actions have $(\alpha, \beta) = (0.60, 0.02)$ and $(0.02, 0.60)$, giving gains of 0.29 and -0.29; type B swaps the two actions. Because a confidence-only policy cannot distinguish the types, it must use the same action distribution in both, and its averaged gain is zero; STOP also yields zero. A state-aware policy instead selects the beneficial action in each type. With a common positive action cost c and $0 < \lambda c < 0.29$, confidence-only control optimally stops, whereas state-aware control attains utility $0.29 - \lambda c$. □

The counterexample pinpoints the action information that a confidence-only statistic discards: optimal actions conflict within a confidence stratum, the simplest source of a positive path-information dividend. The same-parent position analysis tests whether this distinction matters in natural traces, whereas cross-question correctness prediction measures a different capability.

B.8 Proof and interpretation of Proposition 4

Let E_k mean that new completion k is correct. On blocked type $Z = 0$, $\mathbb{P}(E_k \mid Z = 0) \leq \epsilon$. Hence

$$\mathbb{P}\left(\bigcup_{k=1}^{m} E_k \,\middle|\, Z = 0\right) \leq \min\left\{1, \sum_{k=1}^{m} \mathbb{P}(E_k \mid Z = 0)\right\} \leq \min\{1, m\epsilon\}.$$

The complementary type contributes at most one, and taking the mixture over Z proves the stated bound. If the initial pool contains no correct member, selected correctness requires at least one E_k, which gives the output bound. This implication fails when the initial selected answer is wrong but an unselected correct answer already exists, which is why the proposition states the stronger initial-pool condition explicitly.

Under the additional assumption of conditionally independent completions with success probabilities p on $Z = 1$ and ϵ on $Z = 0$, new-candidate coverage equals

$$q[1 - (1 - p)^m] + (1 - q)[1 - (1 - \epsilon)^m].$$

For $\epsilon = 0$ the curve is bounded by q; if both $p > 0$ and $\epsilon > 0$, it approaches one as $m \to \infty$; and if $p = 0$ and $\epsilon > 0$, its limit is instead $1 - q$. A blocked mixture therefore reflects low conditional success within a finite budget rather than an intrinsic impossibility, and earlier reopening can improve the completion distribution by removing the retained commitment.

B.9 Proof and robust extension of Theorem 2

For an initially incorrect selected answer, one-step expected correctness gain equals selected success probability, even if a correct candidate was previously unselected. Every safe checkpoint therefore has utility $p - \lambda c_j$, and strictly decreasing safe costs together with $\lambda > 0$ make $j^\dagger$ the unique best safe checkpoint. Because costs are nonnegative, every blocked checkpoint has utility at most ϵ. The margin $p - \lambda c_{j^\dagger} > \epsilon \geq 0$ then makes $j^\dagger$ strictly better than every blocked position and than STOP.

Equal success across safe prefixes isolates the value of retained work. The following extension relaxes this condition to bounded variation in success. Suppose the safe success probabilities lie in $[p - \eta, p + \eta]$. If

$$(p - \eta) - \lambda c_{j^\dagger} > \epsilon, \qquad \lambda(c_j - c_{j^\dagger}) > 2\eta \quad \text{for every earlier safe } j,$$

then $j^\dagger$ still strictly dominates: in the worst case its success probability is 2η below that of an earlier safe position, and its cost saving exceeds that difference. When these inequalities do not hold, the learned intervention surface selects the appropriate position rather than imposing a fixed backtracking rule (Contribution 2).

B.10 Telescoping rewards and Bellman optimality

In the sum of $Y_{t+1} - Y_t$, every intermediate correctness appears once with a positive and once with a negative sign, so only $Y_T - Y_0$ remains, establishing Eq. (4). Because the initialization cost and Y_0 do not depend on subsequent decisions, maximizing this remaining utility is equivalent to maximizing final correctness minus total cost. The Bellman recursion then follows by conditioning on the first action and maximizing over the continuation decision. At an already scored state, stopping yields neither a correctness change nor additional cost.

This reward prevents a loop from accumulating credit by alternately reselecting the same correct answer: a repair followed by damage sums to zero gain, whereas a coverage reward would remain nonnegative and optimize a different objective. Likewise, adding an uncharged verifier or teacher invalidates the intended compute objective, even if answer quality improves.

B.11 Proof of Theorem 3

Let a^* maximize the true optimal action value over all feasible actions including STOP, and let $\hat{a}$ maximize its estimate. At any feasible history,

$$Q^*(a^*) \leq \widehat{Q}(a^*) + \varepsilon \leq \widehat{Q}(\hat{a}) + \varepsilon \leq Q^*(\hat{a}) + 2\varepsilon.$$

The chain also holds when either action is STOP, since both its exact and estimated values equal zero. Denote the remaining policy regret by $\Delta_h = V_h^* - V_h^{\hat{\pi}}$. If $\hat{\pi}$ continues, replacing the optimal continuation by the continuation of $\hat{\pi}$ gives

$$\Delta_h \leq 2\varepsilon + \mathbb{E}[\Delta_{h-1} \mid S, \hat{a}].$$

If it stops, the second term vanishes. Since $\Delta_0 = 0$, induction on h yields $\Delta_H \leq 2H\varepsilon$.

Writing the remaining utility as $\mathbb{E}[Y_T - Y_0 - \lambda C_{\mathrm{rem}}]$, the bound implies

$$\mathbb{E}[Y_T^*] - \mathbb{E}[Y_T^{\hat{\pi}}] \leq 2H\varepsilon + \lambda\big(\mathbb{E}[C^*] - \mathbb{E}[C^{\hat{\pi}}]\big) \leq 2H\varepsilon + \lambda\delta_C.$$

The benchmark in this bound is the optimum for the *specified price and action set*. Recovering every point of a constrained frontier would additionally require appropriately randomized policies and the corresponding optimization assumptions. Feasibility is enforced by action masking rather than inferred from the approximation bound.

B.12 Finite-sample effect estimation at a fixed parent

Proposition 6 (Fixed-parent concentration)**.** *For a fixed labeled parent and A feasible actions, take n independent completions per action. Let $\hat{g}_a$ be the mean of $D_i = Y_{1,i} - Y_0$. With probability at least $1 - \delta$,*

$$\max_a |\hat{g}_a - g_a| \leq \sqrt{\frac{\log(2A/\delta)}{2n}}. \tag{27}$$

For known expected costs, greedy one-step net-value regret is at most twice this bound. If expected cost estimates have uniform error ε_c, add $2\lambda\varepsilon_c$.

Proof. Because Y_0 is fixed, D_i lies in either $\{0, 1\}$ or $\{-1, 0\}$, an interval of width one. Hoeffding's inequality bounds the error for a single action by $2e^{-2nt^2}$; a union bound over the A actions with $2Ae^{-2nt^2} = \delta$ gives Eq. (27). Comparing the estimated and true maximizing action values incurs two estimation errors, and an expected-cost error perturbs each net value by at most $\lambda\varepsilon_c$. □

The training design uses two repetitions per action and shares statistical information across parents through the neural head, whereas action-level evaluation relies on larger independent diagnostic banks. The concentration bound concerns fixed-parent sampling; generalization of the neural head is a separate question. If parents are pooled without conditioning, the range of the difference variable can have width two, and the corresponding Hoeffding constant changes.

B.13 Probability-head error and action regret

Suppose all predicted probabilities lie in $[0, 1]$ and their errors at a stratum satisfy $|\hat{r} - r| \le e_r$, $|\hat{\alpha} - \alpha| \le e_\alpha$, $|\hat{\beta} - \beta| \le e_\beta$. By adding and subtracting terms,

$$\hat{g} - g = (1 - \hat{r})(\hat{\alpha} - \alpha) - \hat{r}(\hat{\beta} - \beta) + (r - \hat{r})(\alpha + \beta).$$

Therefore $|\hat{g} - g| \le e_\alpha + e_\beta + 2e_r$. With expected-cost error e_c, the immediate net-value error is bounded by $e_\alpha + e_\beta + 2e_r + \lambda e_c$, and twice this expression bounds one-step greedy regret. The inequality links the interpretable heads to decision quality and shows why calibrating r alone is insufficient: errors in α and β also enter the bound. Average calibration error is not a uniform prediction bound and cannot be substituted for e_r in this statement.

B.14 Equal-cost adaptive allocation

For a one-step action of fixed positive cost, let $M(S)$ denote its known net correctness gain. If exactly a fraction q of problems can receive the action, assigning it to the highest-M fraction maximizes $\mathbb{E}[M\mathbf{1}\{\text{selected}\}]$: exchanging a chosen lower-value state for an unchosen higher-value state preserves the budget and improves the objective, and repeated exchanges, with randomized tie-breaking where needed, yield the claim. This allocation dominates question-independent random assignment, whose gain is $q\mathbb{E}[M]$. Under an *at-most* budget constraint, negative-gain states should remain unselected. This elementary fact is the standard rationale for adaptive computation; the learning problem here is estimating M for prefix interventions under imperfect selection.

C Algorithm and implementation specification

C.1 Online inference

Algorithm 1 implements the controller of Contribution 2 under a fully charged ledger. The pseudocode charges each controller call when it is executed, whereas the Bellman equations attribute the *same* call to the preceding transition, except that the first call is attributed to initialization. These are two views of one ledger, not double charging. No next-decision call is charged after the final allowed action or when the hard limits already force a stop. Because the decision pass scores a fixed padded grid (12 slots in the default method), its arithmetic cost is known before masking. The first mask therefore subtracts this cost before admitting any action; otherwise a barely affordable action could exceed the hard budget after scoring. The upper bound additionally reserves one subsequent decision pass when applicable. In the Bellman equations, b denotes the budget remaining at an already scored decision. Reserving a call does not charge it: under a forced stop, the reserved call incurs zero cost.

C.2 Checkpoint construction and native completion

A candidate is an original assistant token stream plus a parsed final answer. For models with an explicit thinking channel, checkpoints lie inside the reasoning region, before its terminal answer

Algorithm 1 RAVEC inference with explicit computation accounting

Require: Problem x, frozen θ, ρ, learned ϕ, total budget B, price λ, horizon $H = 6$
Generate initial candidate z_0 with the common 4,096-token cap
Initialize $\mathcal{Z} = \{z_0\}$, score/cache the incumbent, charge all initialization costs
Record failure candidate $\perp$ if initial completion is invalid
for $h = H, H-1, \ldots, 1$ **do**
    Form unique legal checkpoints and cross with $\{512, 1024, 2048\}$
    Reserve the current padded-grid decision cost d_ϕ before action masking
    if $b < d_\phi$ **then**
        break
    end if
    Mask actions exceeding remaining tokens or $b - d_\phi$, including a next-decision reserve
    if no feasible action remains **then**
        break
    end if
    Charge d_ϕ, set $b \leftarrow b - d_\phi$, score the padded grid, and maximize over feasible actions
    if $\max_a Q_\phi(I, b, h, a, \lambda) \leq 0$ **then**
        break
    end if
    Execute $a = \arg\max Q_\phi$ using the retained original token prefix
    Charge actual generation and prefix processing, including invalid attempts
    if new candidate is complete and valid **then**
        Score it once with ρ and add it to $\mathcal{Z}$
        Select the candidate with maximum cached score, breaking ties by earlier arrival
    end if
    Update checkpoints, observed history, and remaining budget
end for
return selected final answer, or invalid-response marker if all candidates are invalid

delimiter; for models without such a channel, boundaries lie in the solution prose or program text before the final response is closed. The root is always available. For each target fraction of reasoning length, the nearest legal boundary is selected, with ties broken toward the earlier boundary, and identical token offsets are merged. A reasoning segment without an internal legal boundary admits root reopening only.

No natural-language instruction revealing the error, reference answer, or a diagnostic label is inserted. Generation resumes directly after the original problem, the model-native chat prefix, and the retained assistant prefix. In code, a newline checkpoint may retain an incomplete syntactic block, which the continuation must complete; existing indentation and bytes are not reformatted. A generated program with invalid syntax is scored as unsuccessful and is still charged, because syntactic validity is a format check and not a substitute for hidden functional correctness.

The last 128 tokens of each cap are reserved for final-answer completion. In explicit-thinking formats, the native reasoning terminator is inserted at that boundary if it is absent, after which the model generates the final answer. In ordinary mathematical or choice-answer instruction formats, a fixed final-answer cue is used only within the common completion rule, and its tokens are charged. For code output, neither an answer cue nor a thinking terminator is inserted into program text: generation continues naturally to the cap, including the final 128-token allowance, and an incomplete program remains invalid. All baselines share this rule. A response that still lacks a valid task answer is invalid, and the former incumbent is retained. The cap is an upper limit, not a commitment to produce all ℓ tokens. Answer cues and forced endings can affect accuracy and are part of the evaluation protocol.

When no valid initial candidate exists, the pool contains the failure candidate $\perp$, which carries a sentinel selection score of -1 (valid ORM probabilities lie in $[0, 1]$) and no nonroot checkpoint. In the controller inputs, missing incumbent and pooled vectors and empty-pool score summaries are set to zero, the valid pool size is zero, and current reliability is fixed to zero. The sentinel is excluded from valid-pool averages, so no infinite score reaches an MLP. Root actions can replace the sentinel,

and stopping returns the recorded invalid outcome, which scores zero. Hard or truncated questions thus remain in the denominator.

C.3 Exact controller inputs

Four learned projections enter the action head: the problem-end representation, the action-checkpoint representation, the endpoint of the currently selected candidate, and the mean endpoint representation over completed candidates. The projection is shared within a generator, and all endpoint features are recorded before any reference-answer or hidden-test evaluation.

The 20 scalar inputs are retained-token fraction, suffix cap, remaining operation budget, remaining horizon, compute price, selected ORM score, largest-minus-second-largest ORM margin, mean ORM score, ORM score standard deviation, valid pool size, selected response length, initial response length, decoded tokens so far, paid operation count, number of executed reopenings, number of root reopenings, number of invalid suffixes, count of previous actions at the proposed location, last retained fraction, and last suffix length. All continuous resource quantities are scaled by fixed training-set constants, and no normalization uses test labels. With a single candidate in the pool, the score margin is defined as zero; this is a feature convention, not additional model evidence.

At the population level, the action-independent $r(I)$ conditions on the same pre-action information as the repair and damage definitions. The implemented reliability head deliberately restricts this information to the projected question, incumbent, and pooled vectors plus 12 state scalars: selected score, score margin, score mean and standard deviation, pool size, selected and initial lengths, past decoded tokens, paid compute, reopening count, root count, and invalid count. Because it omits the proposed location, suffix cap, future price, and decision horizon, its prediction refers to the existing answer rather than the evaluated action. This restricted summary need not be sufficient for true reliability given all path features. The exact factorization is a statement about the true conditional probabilities; restricted neural heads introduce approximation error, and calibrating their marginal outputs does not remove it.

For generator hidden width d, the controller comprises a shared $d \to 256$ affine projection, an action MLP of widths $1044 \to 256 \to 256 \to 3$, and a reliability MLP $780 \to 256 \to 1$, which together contain

$$N_\phi = 256d + 534{,}532$$

trainable parameters; this architectural count includes all trainable weights and biases. For an illustrative $d = 4096$, it equals 1,583,108 parameters. The action head uses sigmoid outputs for α, β and a linear output for Q, the reliability head uses a sigmoid output, and hidden layers use GELU activations. Hard-cost bounds follow deterministically from the architecture, full context length, maximum suffix length, and verifier/controller charges. Expected cost is predicted by a separate lookup table that averages incurred operations over offline training transitions, with cells defined by generator, cache mode, source-prefix length in 256-token bins, suffix cap, pool size, and terminal versus nonterminal horizon. It uses no test outcomes; empty cells use the deterministic cap-cost bound. This coarse predictor is an approximation used by the greedy controls and the one-step consistency term; Bellman training uses each transition's incurred cost rather than its cap. Cost error enters the stated regret bound rather than being assumed zero.

C.4 Offline learning

The three aggregation rounds extend training to states reached by the current controller. Within each finite-horizon fit, targets remain fixed: horizon-specific training may use a shared horizon-conditioned network initialized from the previously fitted copy, but gradients must not propagate through the target's action maximization. With shared parameters, all supported h values remain in the loss to prevent overwriting earlier horizons. Stored transitions keep separate generation, selector, and controller cost components. Budget and horizon relabeling recomputes whether a next decision exists and what it costs, so a next-controller cost is never copied into a relabeled terminal transition; the generator and selector transition itself remains unchanged.

The design uses 32,000 controller-training questions: 16,000 mathematical questions (6,000 MATH training, 6,000 GSM8K training, and 4,000 automatically verifiable algebra problems) and 16,000 code questions (4,500 APPS training and 11,500 TACO training; Hendrycks et al., 2021a; Li et al.,

Algorithm 2 Intervention collection and fitted finite-horizon learning

Require: Question-disjoint training, calibration, validation sets; frozen generator and selector
for each training question **do**
  Sample two independent complete initial attempts
  for each parent and each available (j, ℓ) **do**
    Clone the parent; sample two independent suffixes; record selected-output transitions
  end for
end for
for aggregation round $k = 0, 1, 2, 3$ **do**
  Fit reliability and masked repair/damage heads with parent-normalized losses
  for $h = 1, \ldots, 6$ **do**
    Build compatible budget/price labels for stored transitions
    Freeze the $h - 1$ target predictor; construct Bellman targets with feasible successor actions
    Fit Q_h with Huber loss and retained probability losses; add consistency if $h = 1$
  end for
  if $k < 3$ **then**
    Run current policy on training questions; choose reached parents without outcome filtering
    Fork selected parents and expand all available actions once; append transitions
  end if
end for
Freeze the shared projection, hidden layers, and probability readouts; calibrate the three probabilities
Refit only the linear Q readout with fixed calibrated targets and all supported horizons
Select compute prices on validation questions; freeze all decision parameters for testing

2023). A further 800 calibration and 1,600 validation questions are drawn from unused training-source items and balanced between mathematics and code. No benchmark test question enters any of these sets, and the generated algebra distribution used for training is disjoint from the controlled mechanism templates and parameter ranges used for analysis.

Each expansion independently resets the cloned parent and cache. The 8,000 parents per aggregation round are sampled uniformly from reached training states, stratified only by source domain and never by correctness; repeated visits to the same state are recorded as visits rather than as new questions. With two parents, 12 distinct actions, and two suffixes per action, the initial design contains at most $32{,}000 \times 2 \times 12 \times 2 = 1{,}536{,}000$ transitions per generator. Each of the three aggregation rounds adds at most $8{,}000 \times 12 = 96{,}000$, for a design maximum of 1,824,000 transitions per generator and 25,536,000 across the 14 generators before duplicate checkpoints are merged. At a 900-token mean suffix length, this maximal intervention collection corresponds to approximately 23.0 billion generated tokens across the 14 models, excluding initial attempts, selector training, and evaluation. Generation therefore dominates the offline cost despite the controller's small parameter count.

Training uses AdamW (Loshchilov & Hutter, 2017) with learning rate 3×10^{-4}, weight decay 0.01, batches of 512 transitions, and at most ten passes per fitted stage, with the checkpoint selected by validation value loss. The learning rate is compared over $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$ and the consistency weight over $\{0, 0.1, 0.3\}$; the Huber transition is 1. Prices are sampled from $\{0, 0.005, 0.01, 0.02, 0.05\}$ per thousand generator-equivalent operation units and then selected on validation data for each budget. Interpolation between two fixed policies may define an intermediate expected-cost point but is never tuned to test correctness. Appendix I defines the computation units and their conversion.

Temperature scaling (Guo et al., 2017) fits separate positive scalar temperatures for r, α, β on supported calibration examples. The final fit updates only the linear Q readout, keeps all horizons in its loss, and uses frozen lower-horizon targets, so it cannot alter calibrated probabilities through shared hidden weights. Calibration therefore precedes this fit whenever it changes the one-step consistency target; calibration applied after freezing Q would change only diagnostics and is not credited with changing decisions. The default BCE applies no class balancing, which preserves probability base

rates; any alternative oversampling would require explicit importance weights. The temperatures are selected by calibration-split loss, not by test accuracy.

C.5 Shared selector and process verifiers

Because the selector gates every generated improvement (Proposition 1), it is frozen and shared by all methods. Selector training uses a fixed 8,000-question subset of the controller-training sources, balanced across math and code, together with 400 selector-validation questions taken from the validation sources. No controller-calibration or diagnostic question enters this training set; learned components may share training questions, but held-out evaluation questions remain disjoint from all of them. The outcome selector is Qwen3-0.6B (Yang et al., 2025) with a scalar endpoint head and rank-16 LoRA (Hu et al., 2021) on the attention projections. It is trained on eight independent responses per selector-training question, drawn evenly from Qwen3-8B, Qwen3-14B, R1-Distill-Qwen-7B, and Gemma-3-12B-IT, with binary correctness given by exact mathematical checking or the training problem's code tests. The scalar is fitted with BCE, and candidate scores are temperature-scaled on the selector calibration split. Selector-training candidates are generated independently of controller parents. The selector reads the question and full candidate, never sees hidden evaluation labels, and is identical across methods within a comparison. LoRA training uses learning rate 2×10^{-5}, batch size 64, and at most three epochs, with the checkpoint selected by validation BCE.

The PAV and process-verifier controls receive a separately fitted step-scoring head built on the same 0.6B backbone and the same question sources. For PAV, $V_p(s)$ denotes the success probability of the frozen Qwen3-4B prover (Yang et al., 2025) when completing prefix s under its fixed native completion policy. A step from s to s^+ receives the target $V_p(s^+) - V_p(s)$, with each value estimated from eight independent prover completions capped at 2,048 tokens. The process head is fitted by squared error to this signed difference, and a separate sigmoid readout fits $V_p(s)$ for search controls that require a nonnegative success score. Sample allocation and failed completions count toward the same offline labeling-operation allowance, consistent with the distinct-prover role in Setlur et al. (2024). This control thus uses PAV continuation-progress supervision in place of RAVEC's terminal selected-output targets. In supervision-matched comparisons, its total labeling-generation operation budget equals RAVEC's collection budget, and both labeling and process-model training enter the offline cost. The comparison thus harmonizes supervision budget and evaluation while retaining the PAV-style target, varying the target rather than the amount of supervision.

D Benchmark and model specification

D.1 Task definitions

Table 3: Specification of the 12 benchmarks spanning mathematics, code, and general knowledge reasoning: evaluation size, metric, and slice. *The 400-item LiveCodeBench subset is a custom deterministic selection defined by release date, source site, source problem identifier, language, and evaluability criteria.

Task	Size	Metric	Evaluation slice
GSM8K	1,319	Exact answer	Official test split; numeric normalization only.
MATH-500	500	Math equivalence	MATH-500 subset (Lightman et al., 2023) of MATH (Hendrycks et al., 2021b); task-wise answer parser.
AMC23	40	Exact answer	AMC 12 A and B, 2023, the conventional 40-problem slice.
AIME24	30	Integer answer	AIME I and II, 2024; both 15-problem sets.
AIME25	30	Integer answer	AIME I and II, 2025; both 15-problem sets.
Minerva Math	272	Math equivalence	Conventional undergraduate STEM/MATH evaluation set.
OlympiadBench	675	Math equivalence	English, text-only, open-ended mathematics subset.
HumanEval+	164	Strict pass@1	HumanEval (Chen et al., 2021) with EvalPlus tests (Liu et al., 2023); one selected program.
MBPP+	378	Strict pass@1	Sanitized MBPP (Austin et al., 2021) with EvalPlus tests (Liu et al., 2023); one selected program.
LiveCodeBench	400*	Strict pass@1	Text-only 2024-08-01–2025-01-31 date slice.
GPQA Diamond	198	Choice accuracy	Fixed question order and answer permutation policy.
MMLU-Pro	12,032	Choice accuracy	Official test split, all categories.

The LiveCodeBench evaluation selects the earliest 400 eligible Python problems in the stated interval, ordered by release date, source site, and source problem identifier; eligibility requires a public problem statement and an evaluable hidden-test specification. The slice is therefore identified by this deterministic rule rather than by a named release version. The coding diagnostic measures functional correctness on this fixed interval; pretraining contamination for later model releases remains a separate consideration.

For mathematics, the selected response must supply one final task answer, and a wrong format, missing answer, or timeout counts as incorrect. Algebraic equivalence uses the benchmark's conventional normalization, with no LLM judge for ambiguous cases. For code, only the selected complete program is run against the hidden tests, so the score is selected pass@1 after inference computation rather than oracle pass@k; this is the selected-answer quantity targeted by Contribution 1. Public examples are part of the problem statement but are not executed online in the main setting. A separate execution-feedback setting may use them, but it must charge execution and is never pooled with no-feedback results.

GPQA and MMLU-Pro use their released option order, a fixed task-wide prompt, and the first unambiguous final option label; malformed outputs or multiple conflicting labels count as incorrect. The answer-order and parsing policy is thus identical across methods.

The core macro average weights MATH-500, AIME24, AIME25, OlympiadBench, HumanEval+, MBPP+, LiveCodeBench, and GPQA equally. This task-weighted average is a reporting convention, and the small AIME sets carry substantial empirical uncertainty, so per-task scores and alternative aggregates are reported alongside it. GSM8K, AMC23, Minerva Math, and MMLU-Pro serve as complementary tasks, and the full matrix permits separate inspection of math, code, and general reasoning.

D.2 Model identifiers and release provenance

Table 4: Checkpoint identifiers for the 14 evaluated generators from four families. Abbreviated entries inherit the organization of the first entry in their row.

Family	Checkpoint identifiers (organization / model)	Evidence
Qwen3	`Qwen/Qwen3-4B`, `Qwen3-8B`, `Qwen3-14B`, `Qwen3-32B`	(Yang et al., 2025)
Qwen3.5	`Qwen/Qwen3.5-9B`, `Qwen3.5-27B`	(Qwen Team, 2026a)
Qwen3.6	`Qwen/Qwen3.6-27B`	(Qwen Team, 2026b)
Qwen3.8	`Qwen/Qwen3.8-27B`	(Qwen Team, 2026c)
R1 distillation	`deepseek-ai/DeepSeek-R1-Distill-Qwen-7B`, `-14B`, `-32B`	(DeepSeek-AI et al., 2025)
Gemma 3	`google/gemma-3-12b-it`, `gemma-3-27b-it`	(Gemma Team et al., 2025)
Llama 3.1	`meta-llama/Llama-3.1-8B-Instruct`	(Grattafiori et al., 2024)

Multimodal-capable models receive text inputs only, and their inactive vision components are excluded from computation accounting. The Qwen3.5–3.8 models use the dense checkpoint sizes listed above. Cross-model comparisons follow the common capped-generation protocol.

Public-score anchors and comparison scope. The Qwen3 report gives thinking-mode MATH-500 scores of 97.0/97.4/96.8/97.2 for its 4B/8B/14B/32B models under a 32,768-token maximum (38,912 on AIME), showing that model size does not enforce task-wise monotonicity (Yang et al., 2025). DeepSeek's official R1-distillation results use a 32,768-token maximum and 64 sampled answers per question and report MATH-500 scores of 92.8/93.9/94.3 for 7B/14B/32B (DeepSeek-AI et al., 2025). These are external single-response averages, whereas our Single row follows the shorter 4,096-token completion rule and RAVEC additionally selects after intervention. For coding, public Qwen3 results use LiveCodeBench v5 and the 2026 Qwen model cards use v6, while our 400-item date slice constitutes a third population; likewise, HumanEval and MBPP base-test scores are not equivalent to the stricter EvalPlus metrics. Public results anchor model capability; they neither determine the accuracy lost to our shorter cap nor validate a RAVEC gain.

D.3 Splits, contamination, and statistical unit

All derived versions of a problem are assigned to the same split. Preprocessing identifies such versions through source problem identifiers, exact normalized-text comparison, and source-aware review of translated or reformatted duplicates. This discipline controls overlap within the controller data; contamination of a generator's pretraining corpus requires a separate, release-specific assessment.

The statistical unit is the question. The primary protocol specifies 64 paired decoding replicates per question–method cell for each of five controller fits. Within a replicate, reopening methods share the initial decode seed and the initial attempt, and downstream sampling is independent after that parent. Long-CoT and Wait controls change the initial completion rule, so they share a seed rather than an identical completed parent. Each fit averages replicates within a question and then weights questions equally within a task; the reported point is the mean of the five fit-level task means. Single and other unfitted controls use one shared evaluation bank rather than five artificial retrainings, and the core macro weights its eight task means equally. Thus a per-fit AIME score has 30 questions times 64 replicates in its denominator, and the five-fit mean adds a further equal-weight average; neither operation creates more than 30 independent question clusters.

The primary accuracy analysis draws 10,000 paired cluster-bootstrap resamples of whole questions within each task, carrying every method, replicate, and operating point of each sampled question. Cluster-only intervals hold the five fitted controllers fixed and average their paired gains, describing question-sampling uncertainty in replicate-averaged outcomes rather than pure decoding noise; a decode-only analysis would instead fix questions and resample paired replicate indices. Between-fit variation uses five independent optimization seeds on the same training transitions, frozen generator and selector, and fitting protocol, and thus isolates controller-fitting variation rather than fresh data collection. The nested analysis first resamples paired fit indices jointly across tasks and then resamples paired question clusters within each task. The fit-level standard deviations of the gain over PB-SMC are 0.16, 0.18, and 0.22 points for core, MATH-500, and LiveCodeBench.

When an expected-cost point mixes two validation-selected policies, the mixing weight is fixed before test evaluation and applied through a Bernoulli policy choice in each replicate. Questions and replicates remain equally weighted, and mixture weights never reweight test outcomes. Matched-accuracy resampling rebuilds the paired accuracy–cost frontiers within each draw and interpolates linearly on their increasing, nondominated segments. A draw in which either method misses the target is recorded as unattained rather than extrapolated or dropped, and this fraction must be stated before interpreting an interval conditional on attainment. Figure 2 presents the interval analysis. Despite repeated decoding, the small AIME sets retain substantial question-level uncertainty.

D.4 Evaluation design

The primary evaluation crosses 14 models, 12 benchmarks, and four common methods. Eight additional Qwen3-8B controls isolate specific components. The natural-parent diagnostic uses 512 math and 512 code questions with two parents each; selector exchange uses the first parent from each question in this cohort. The action-ranking diagnostic uses a separate question pool with independent discovery and evaluation banks. The factorial and matched-feature root comparison isolate reward/search and position-access effects, respectively.

The generator, selector, checkpoint tokenizer, action grid, data splits, and controller-selection rule are fixed before evaluation. Test outcomes do not enter controller fitting or threshold selection.

E Baselines, adaptations, and matched comparisons

The comparators examine *search access*, *information*, and *objectives* under a shared generator and operation ledger, with a common terminal selector where applicable. Single-factor attribution uses the factorial and matched-feature controller comparisons; adapted baseline families can differ along several dimensions.

Single and longer chains. Single returns the shared initial attempt and pays for one selector evaluation. Long-CoT spends the available generation allowance on one extended chain with the same

native termination convention and no intermediate controller; its larger cap remains subject to the common operation and decoded-token hard limits. This control isolates the value of placing computation into new attempts rather than a longer chain. An s1-style "Wait" continuation is reported as a prompt-level adaptation inspired by Muennighoff et al. (2025); its original training recipe is outside this harmonized comparison.

BoN and self-consistency. These are the candidate-coverage baselines. BoN (Lightman et al., 2023) resamples from the root, scores all valid candidates with ρ, and returns the highest-scoring one, with $N \leq 7$ including the initial attempt; adjacent fixed-N policies can be randomized at the question level to match expected budgets. Self-consistency (Wang et al., 2022) instead votes over normalized mathematical final answers, breaking ties with the common ORM, and is therefore meaningful on mathematical or choice tasks; it is not treated as equivalent to semantic voting over arbitrary code. Self-consistency scores the shared first attempt for common initialization and then votes without additional ORM calls unless a tie requires them. All executed scoring is charged; saved scoring cost can buy generation under a matched total budget.

Root-adaptive. This control receives the same projected full-trace, discarded-suffix, incumbent, pool, confidence, resource, and price inputs as RAVEC, together with the same data-collection budget, STOP action, network width, and finite-horizon fit, but it may take root actions only. It therefore tests the value of choosing positions (Contribution 2) beyond learning when to stop and how many tokens to allocate. Its targets are fitted for its smaller action domain rather than set by a constant threshold.

Question-only. The controller observes the question representation, budget, horizon, candidate count, and already-observed length and cost summaries, while trajectory hidden vectors and ORM scores are removed. Its input projections preserve approximately the same trainable parameter count and training schedule. It can thus learn question difficulty and remaining budget, as question-level allocation does, but cannot infer a position from the current trace's content. This makes it a stronger confound control than a scalar difficulty bin.

Confidence-only. This control additionally receives a state correctness estimate, selector score and margin, lengths, and resource summaries, but not checkpoint or trajectory representations. It chooses among the same positions and suffix budgets, and its reliability calibration uses the same training split. It tests whether the net benefit follows merely from knowing that the current answer is wrong, without identifying which retained prefix is useful.

PAV-guided reopening. A process-advantage verifier trained with a distinct frozen prover estimates changes in continuation success. At each iteration, the controller chooses the checkpoint preceding the most negative predicted process advantage among the available boundaries, with its suffix budget and stopping price selected on validation data, and it retains the common terminal ORM. This explicitly specified controller uses a PAV signal for a controlled comparison of step-progress and selected-output targets. A broader PAV tree-search control appears in the supplemental sensitivity scenario.

VGB adaptation. The adaptation preserves stochastic moves guided by a process verifier and permits backing up to earlier checkpoints rather than only expanding the current frontier. It uses the same legal boundary grid, suffix budgets, terminal ORM, and operation ledger as the other controls. The transition bias toward forward or backward movement is tuned over $\{0.25, 0.5, 0.75\}$ and the verifier temperature over $\{0.5, 1, 2\}$, both on validation data. Because the tree and verifier setup of the original method differ (Rohatgi et al., 2025), the tables label this control VGB-adapted throughout, and no convergence claim from the original work is transferred to this harmonized policy without re-derivation.

Persistent-pool Subpool and PB-SMC adaptations. These are the historical-prefix search controls. A persistent pool retains every legal checkpoint exposed by completed candidates, including checkpoints of nonselected candidates. In these harmonized controls, each iteration generates one new candidate, and particle slots are checkpoint proposals. Subpool samples a uniform random subset of at most eight checkpoints and expands the highest process score within it. PB-SMC uses four particle slots, powered process-score weights, and a mixture proposal with 0.2 uniform mass over the persistent pool. For pool scores v_j, define $w_j = \max\{10^{-4}, \min(1, v_j)\}^{\gamma}$ and $p_j = w_j / \sum_k w_k$. Draw four checkpoint proposals with replacement from $q_j = 0.8p_j + 0.2/|\mathcal{P}|$, then select one proposal in proportion to p_j/q_j and generate a single suffix. Duplicated proposals remain duplicated

slots. This finite-proposal importance-resampling rule defines the PB-inspired adaptation within our common action and cost space. The powered weights raise a clipped success score to an exponent selected from $\{1, 2, 4\}$. Unlike RAVEC's current-incumbent grid, these controls can revisit nonincumbent history, which gives them broader search access; generation still counts against the common total budget and candidate limit. The harmonization retains the persistent-pool principle of Tran et al. (2026) while using a bounded search space and the common terminal selector.

Same-target controls. These controls hold the intervention target of Contribution 1 fixed and vary how it is used. Greedy net value ranks actions by $\hat{g} - \lambda\hat{c}$ with exactly the same intervention targets as RAVEC. Gain/cost ranks by $\hat{g}/\hat{c}$ with a validation-selected stopping threshold. Random-position/learned-budget samples an eligible checkpoint uniformly but learns the suffix budget and stopping decision. Direct-ΔY control drops the interpretable probability losses and the one-step consistency term while keeping the same Q architecture and Bellman reward. These controls separate the information target, output factorization, action domain, and multistep planning.

Interpretation of adapted baselines. Harmonization compares control rules under a common generator, selector, and operation budget; supervision matching concerns labeling-generation operations. Capacity is matched within paired controller ablations, whereas the 0.6B pretrained process verifier and the small hidden-state controller retain their respective pretrained capacities and adaptation parameter counts, and their architectures and extra scoring costs are reported separately. The expanded-search sensitivity analysis enlarges historical pools and candidate counts under the same total-operation constraint.

F Additional controls and sensitivity analyses

The supplemental analyses inform search configuration, selector adaptation, domain specialization, and collection budgets. Wider historical search strengthens the comparator, providing a stringent AIME24 comparison. A stronger outcome selector changes both absolute accuracy and the controller target, motivating refitting as predicted by Proposition 1. Domain-matched code training improves transfer over math-only fitting, and increasing collection data strengthens the controller's advantage over search across the tested scales.

The additional selector condition uses a separately trained Qwen3-4B scalar outcome model (4B ORM) with the same question sources and labels. Every baseline in that condition pays the larger scoring cost, and its C_0 includes the changed selector; normalized costs under the primary 0.6B selector and under the 4B ORM therefore do not correspond to equal absolute operations. A controller transferred unchanged to the new selector experiences target shift in both α and β. Retraining therefore uses transitions rescored under the new selector and adds on-policy state collection, because the selected histories also change.

An "oracle candidate" diagnostic records whether a correct candidate exists in the pool a policy has generated; it is an evaluation-only ceiling, not an implementable selector. RAVEC reaches pool coverage of 99.1% on MATH-500 and 64.3% on LiveCodeBench, against selected accuracies of 98.8% and 56.4%, so the coverage–selection gaps of 0.3 and 7.9 points show that selector error remains substantial on code. PB-SMC reaches corresponding coverage of 98.7% and 63.2%, with larger gaps of 1.4 and 8.8 points. RAVEC thus improves selected output over PB-SMC with only a small coverage increase and a smaller coverage–selection gap, consistent with Contribution 1's post-selection target. These coverage values compare policy-generated pools; the fixed-pool selector diagnostic is reported separately.

G Ablations and measurement definitions

The ablations are defined as follows. "No damage term" trains a repair-only objective on positive correctness changes $\max(\Delta Y, 0)$, drops β supervision, and uses $(1 - r)\alpha$ as its one-step target; it directly tests the damage component of Contribution 1. Removing only the β head while retaining ΔY would still penalize damage and would constitute a different ablation. "Coverage target"

Table 5: Additional search controls on Qwen3-8B at the primary expected operation point. Dense and native-budget search variants relax the four-position restriction or candidate-count limit while preserving total operation cost. Each baseline is harmonized to the common selector and accounting protocol. SC is not applied to code.

Control	MATH-500	AIME24	AIME25	LCB
Self-consistency	97.6	72.4	65.9	—
Wait budget forcing	96.1	68.2	62.5	51.2
PAV dense-prefix search	97.9	72.1	65.8	55.0
VGB dense-prefix adaptation	97.5	71.2	65.1	54.6
PB-SMC full-pool budget variant	98.1	74.4	67.0	55.2
RAVEC	98.8	73.0	67.3	56.4

Table 6: Selector dependence. Each selector defines a separate cost normalization; values across selector rows are not absolute-compute matched. Unchanged RAVEC transfers the controller without updating its learned target; refitted RAVEC includes target and reached-state updates.

Selector / controller	MATH-500	LCB	GPQA
0.6B ORM / PB-SMC	97.3	54.4	59.2
0.6B ORM / RAVEC	98.8	56.4	58.7
4B ORM / BoN	97.1	55.2	61.0
4B ORM / PB-SMC	98.0	57.0	61.6
4B ORM / unchanged RAVEC	97.7	56.5	60.8
4B ORM / refitted RAVEC	98.4	58.0	62.0

Table 7: Domain transfer with Qwen3-8B. The shared selector is unchanged. Training-source restrictions apply only to the controller. Math-only to code transfer is weaker than in-domain or joint training.

Controller training	MATH-500	Olympiad	LCB	MBPP+
Math only	98.9	62.8	53.7	78.6
Code only	96.9	59.7	56.4	80.5
Joint math and code	98.8	62.5	56.4	80.0
Joint; unseen math source held out	97.8	61.3	55.6	79.7

Table 8: Collection-size sensitivity. Parent counts refer to available intervention parents, not independent questions or realized datasets. The full setting has at most 88,000 parents per generator (64,000 initial plus 24,000 aggregated).

Parent allocation	MATH-500	LCB	GPQA
4,000 parents	97.0	52.7	58.0
12,000 parents	97.9	54.3	57.6
32,000 parents	98.4	54.1	58.5
64,000 parents	98.3	55.8	58.9
88,000 parents (full)	98.8	56.4	58.7

Table 9: Decoding-temperature sensitivity, with top-p 0.95 and method-specific controller refitting under each generator distribution. The default 0.6 row agrees with the primary table.

Temperature	PB MATH	RAVEC MATH	PB LCB	RAVEC LCB
0.3	97.0	98.0	53.7	55.0
0.6	97.3	98.8	54.4	56.4
0.9	97.0	98.1	54.0	55.5
1.2	96.2	97.0	52.5	54.3

Table 10: Diagnostic errors on held-out diagnostic questions, disjoint from temperature-fitting questions. ECE uses ten equal-mass bins; alpha and beta are scored only on their supported branches. Regret uses a discovery-bank-selected reference action and an independent evaluation bank, as in the paired-learning comparison. Values are fractions, not percentage points.

Setting	r ECE	alpha ECE	beta ECE	Action regret
Math, uncalibrated	0.038	0.064	0.019	0.027
Math, calibrated	0.017	0.041	0.009	0.021
Code, uncalibrated	0.057	0.071	0.034	0.046
Code, calibrated	0.029	0.052	0.018	0.028
GPQA transfer, calibrated	0.074	0.089	0.041	0.051

Table 11: Aggregate controller behavior on Qwen3-8B. Mean reopens includes stopped examples. Root/deep shares condition on a reopening action; deep means the 75 percent checkpoint. Invalid rate conditions on a generated suffix.

Task	Stop after initial (pct)	Mean reopens	Root share (pct)	Deep share (pct)	Invalid (pct)
GSM8K	72.4	0.46	18.7	41.3	0.5
MATH-500	46.8	1.31	20.8	22.6	1.8
AIME25	28.3	2.18	37.1	11.4	4.8
HumanEval+	44.2	1.12	26.9	27.4	3.1
LCB	29.5	1.82	41.2	14.7	6.6
GPQA	53.7	0.88	43.8	15.6	1.0

Table 12: Component and replay ablations of RAVEC on Qwen3-8B. Rows remove or replace one component of the selection-conditioned target (Contribution 1) or of the path-informed controller (Contribution 2), or vary replay accounting. All rows match the $2.4C_0$ expected operation point except "Replay all prefixes; same policy," which keeps the answer distribution identical and isolates the change in cost.

Variant	MATH	LCB	GPQA	Cost/C_0
RAVEC	98.8	56.4	58.7	2.40
No damage term	97.8	55.0	58.0	2.40
Coverage target	97.2	53.9	57.8	2.40
Direct delta-Y Q only	98.1	55.6	58.5	2.40
Greedy net value	97.8	55.3	59.0	2.40
Gain / cost ratio	97.4	54.8	58.4	2.40
Random position; learned budget	96.8	52.7	58.2	2.40
Root only	96.8	53.4	58.6	2.40
No history features	98.1	55.2	58.4	2.40
No randomized collection	97.1	53.9	57.7	2.40
No state aggregation	97.5	54.9	58.1	2.40
One suffix budget (1024)	97.7	55.0	58.3	2.40
Eight prefix positions	98.7	56.1	58.8	2.40
No probability calibration	98.2	55.4	58.0	2.40
Replay all prefixes; same policy	98.8	56.4	58.7	2.62
Replay all prefixes; matched cost	98.1	55.4	58.3	2.40

replaces the supervised reward by ΔU and uses coverage labels consistently in the probability and one-step consistency losses, as specified in Appendix J.1, while evaluation still uses actual terminal selection. This variant deliberately optimizes coverage rather than selected correctness and thereby exposes the selector mismatch of Challenge 1. "No randomized collection" restricts collection to low-PRM checkpoints; its reduced action support is the intended intervention.

"No state aggregation" retains the same initial-parent data but omits all three rounds of reached-state collection. "No history features" removes action counts and previous-action summaries while retaining current pool features and resource budgets. "One suffix budget" leaves only 1,024-token actions. "Eight positions" uses equally spaced legal fractions with duplicate offsets merged; it doubles the number of possible positions, not the number of online generation calls. "No probability

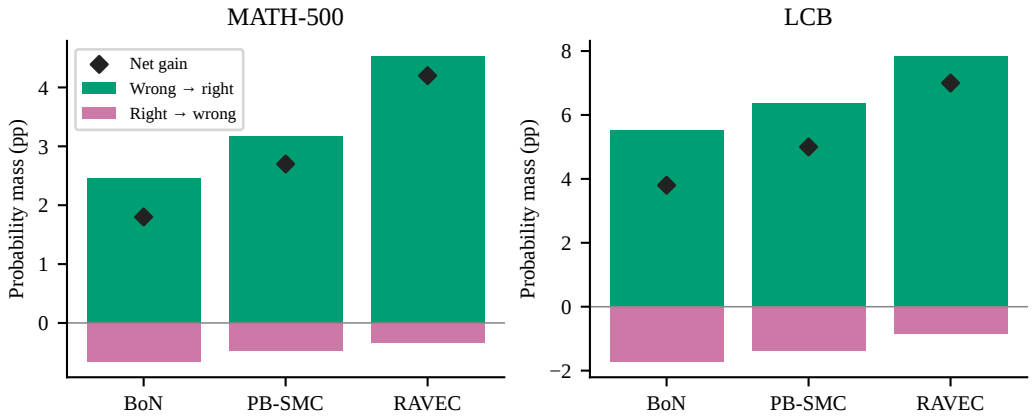


Figure 5: End-to-end repair (wrong-to-right) and damage (right-to-wrong) probability masses from the shared initial attempt to the final selected output. Black diamonds give their difference, which exactly reproduces the accuracy gain over Single in Table 1. These aggregate transitions are distinct from the one-action training labels.

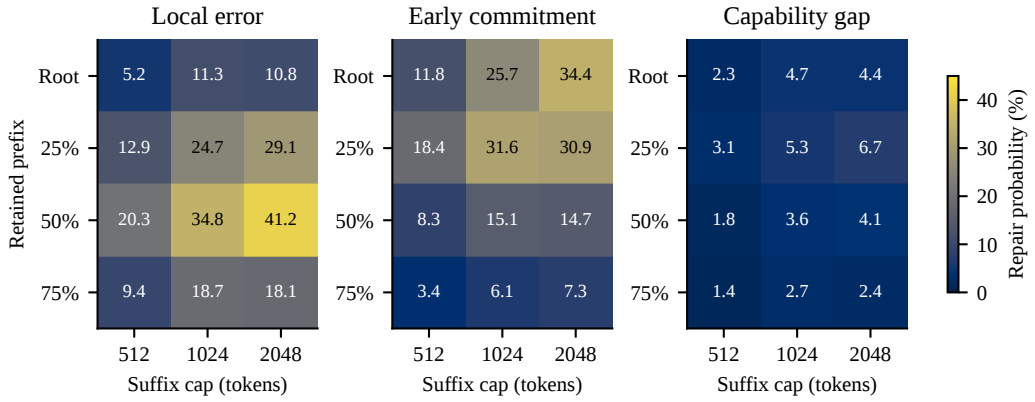


Figure 6: Repair surfaces over retained position and suffix budget for initially wrong, controlled parent classes. Local errors favor retaining progress, an early commitment favors earlier reopening, and capability gaps remain difficult at every position, so the preferred intervention depends on the path. Entries are conditional repair probabilities for these controlled classes rather than benchmark-average accuracies.

calibration" omits both temperature fitting and the corresponding recalibrated final value fit, so it can alter decisions rather than only post hoc displays.

The replay pair shares the generator, selector, and prefix text. The same-policy row replays the original action schedule and stopping decisions with the same decoding draws, assuming numerically equivalent restored and replayed states, and re-accounts the actual replay operations afterward. Its accuracies therefore equal those of the original row exactly, while its cost rises to $2.62C_0$. Feeding the larger cost back into the controller would change its resource features and possibly its actions, so that intervention belongs to the separately retuned, matched-cost row. Hybrid attention/recurrent models require model-specific state handling, so state restoration for them extends beyond attention KV caching.

H Repair, Damage, and Within-Parent Diagnostics

This decomposition separates each method's repair of incorrect incumbents from its damage to correct ones (Contribution 1). For example, the RAVEC row for MATH-500 uses $r = 0.946$,

Table 13: Derived end-to-end repair and damage decomposition of the Qwen3-8B endpoint scores for BoN, PB-SMC, and RAVEC: conditional transition rates (%) and unconditional probability masses (percentage points). On both MATH-500 and LiveCodeBench, RAVEC has the highest repair rate α and the lowest damage rate β of the three methods. α is computed from final accuracy and β so that $r + (1 - r)\alpha - r\beta$ equals the main-table result before rounding; it is an arithmetic consistency check, not an independent mechanism measurement.

Task	Method	r	α	β	W→R	R→W	Final
MATH-500	BoN	94.60	45.60	0.70	2.46	0.66	96.40
MATH-500	PB-SMC	94.60	58.76	0.50	3.17	0.47	97.30
MATH-500	RAVEC	94.60	83.91	0.35	4.53	0.33	98.80
LCB	BoN	49.40	10.93	3.50	5.53	1.73	53.20
LCB	PB-SMC	49.40	12.62	2.80	6.38	1.38	54.40
LCB	RAVEC	49.40	15.49	1.70	7.84	0.84	56.40

$\beta = 0.0035$, and final accuracy 0.988, giving

$$\alpha = \frac{0.988 - 0.946 + 0.946(0.0035)}{1 - 0.946} = 0.839093\ldots.$$

The resulting repair mass is 0.045311 and the damage mass 0.003311, so the net gain is exactly 0.042. On LiveCodeBench, lower current reliability leaves a much larger pool of initially wrong answers, so a lower conditional repair rate can coexist with a larger unconditional repair mass. Reporting only α, or only the number of repaired examples, therefore obscures the deployment effect.

Table 14: Normalized transition summary corresponding to Table 13. Frequencies are expressed as rounded count equivalents per $10{,}000$ initial answers, not raw event counts or an observed sample size. RAVEC has higher wrong-to-right (W→R) and lower right-to-wrong (R→W) frequencies than BoN and PB-SMC on both tasks. Final accuracy is a percentage; uncertainty is assessed with the question-clustered protocol of Appendix D.3.

Task	Method	Scale	Initial wrong	Initial right	W→R	R→W	Final (%)
MATH-500	BoN	10,000	540	9460	246	66	96.40
MATH-500	PB-SMC	10,000	540	9460	317	47	97.30
MATH-500	RAVEC	10,000	540	9460	453	33	98.80
LCB	BoN	10,000	5060	4940	553	173	53.20
LCB	PB-SMC	10,000	5060	4940	638	138	54.40
LCB	RAVEC	10,000	5060	4940	784	84	56.40

The normalized summary makes the initial-correctness branches and their net contributions explicit. Table 13 reports the underlying conditional rates; reconstructing them from the rounded count equivalents can introduce rounding differences. The display denominator is not used for statistical inference.

Table 15: One-step position-choice gain over uniform position selection at a fixed 1,024-token suffix cap, testing whether path information improves the choice of reopening position (Contribution 2). RAVEC attains the largest gain on both math and code, and PAV-reopen, PB-SMC, and RAVEC each lose most of their gain when path features are shuffled. Math and code refer to separate diagnostic parent sets rather than the main benchmark macro. "Shuffled" swaps path features within the same question and narrow observable bins.

Controller	Math gain	Math shuffled	Code gain	Code shuffled
Question-only	0.7	0.7	1.0	1.0
Confidence-only	1.8	1.8	2.1	2.1
PAV-reopen	4.6	1.2	4.1	0.9
PB-SMC	5.7	1.4	5.2	1.3
RAVEC	8.3	1.7	7.1	1.5

The position-choice diagnostic contains 512 mathematical and 512 code questions, two initial parents each, four positions, and 32 independent suffix samples per parent–position pair, split into 16

action-discovery and 16 independent evaluation samples. This amounts to at most $1{,}024 \times 2 \times 4 \times 32 = 262{,}144$ diagnostic continuations per generator. A chosen action is evaluated on a suffix bank separate from the one that produced its diagnostic label, preventing winner's-curse bias toward the largest noisy estimate. Means average actions within each parent first and then parents within each question.

Question-only and confidence-only controls do not read the shuffled trajectory features, so their original and shuffled values coincide by construction. Path-aware controllers retain nonzero gains after shuffling because the shuffle preserves length, confidence, feasibility, and global position information. The drop from original to shuffled gain therefore measures the content-sensitive path information that these observable covariates do not capture, on which Contribution 2 relies.

The controlled surfaces in Figure 6 use three explicitly specified classes. Local-error examples preserve a correct derivation but corrupt a terminal check or edge case; early-commitment examples alter a premise or indexing convention before subsequent reasoning; and capability-gap examples require a rule absent from the narrow task generator's training distribution. Their construction points are known but need not be a language model's optimal reopening boundaries. Natural-error results and controlled-error surfaces are therefore reported separately.

I Computation accounting and resources

For a model-specific operation function F_θ, selection function F_ρ, and controller function F_ϕ, the ledger is

$$C_{\text{total}} = C_{\text{initial}} + \sum_{t=0}^{T-1} \big[F_\theta^{\text{prefix}}(x, z_{\le j_t}) + F_\theta^{\text{decode}}(n_t) + F_\rho(x, z_t) + F_\phi(I_{t+1})\big]. \tag{28}$$

The ledger charges every executed operation, so preserved text is never free computation. An already scored candidate is not rescored by an independent per-candidate ORM. Invalid candidates pay for generation and for any validation or format checks already executed, but not for an ORM call that never occurs. The first controller call, if executed, is included in the method's initialization cost, whereas C_0 remains the shared initial-generation-plus-selector baseline without a controller. The final next-state term is omitted when no next decision is made. Selection among cached scalar scores still incurs its small aggregation cost. Restoring a cached prefix state is charged for the operations performed, and no claim of zero memory traffic follows from a FLOPs ledger.

A generator-equivalent token operation unit is defined as the arithmetic cost of one decode token at a fixed 2,048-token context for the given generator. This gives a readable cost–price scale without treating every token as equally expensive. Initial and suffix operations are computed at their actual context lengths, and the process verifiers, the shared ORM, and the controller are converted through their own operation counts. For dense transformers, both linear-layer and attention terms are included; for Qwen3.5 and later hybrid models, linear-attention/recurrent updates and full-attention blocks are counted according to their actual architectures. Arithmetic cost is determined by the active modules, not by parameter names alone.

Before the first feasibility mask, the current padded-grid controller call is reserved. The action mask then uses the remaining balance and reserves the maximum possible cost of the selected suffix cap, of selector scoring at its maximum resulting length, and of any following decision pass. These arithmetic charges also include feature projection and any endpoint-forward operation needed to record the last token's hidden state, since unavailable hidden states cannot be obtained for free. The remaining decode budget is checked independently. If the initial attempt cannot fit an instance's configured operation budget, its maximum length must be reduced by the common rule before any method is compared, so it can never silently exceed the hard cap; in the reported configuration, the common caps admit the initial attempt. Expected-cost normalization uses the corresponding initial policy rather than a separately cheaper first-pass model.

The cost breakdown is an operation ledger, not a device profiler trace. At the C_0 curve endpoint, every method uses the shared Single bypass policy without invoking a controller, which preserves an exact common anchor. The controller column groups all controller calls, including the first,

Table 16: Itemized operation ledger for Qwen3-8B on MATH-500, in units of C_0 and rounded to 0.01. Initial covers the first candidate and its selector call; every later cost is listed separately. BoN, PB-SMC and RAVEC all total the $2.4C_0$ expected-cost design; replaying the same RAVEC policy without reuse raises only the prefix cost and places the total above that design point.

Method	Initial	Extra decode	Prefix	Selector	Controller / search	Total
BoN	1.00	1.12	0.14	0.14	0.00	2.40
PB-SMC-adapted	1.00	1.03	0.16	0.17	0.04	2.40
RAVEC	1.00	1.10	0.12	0.16	0.02	2.40
RAVEC replay, same policy	1.00	1.10	0.34	0.16	0.02	2.62

which the Bellman timing convention charges at initialization. This regrouping preserves total cost and shows that the headline comparison includes controller overhead: saved decoding is reported alongside, and not in place of, the additional controller work. End-to-end latency further depends on batching, cache traffic, and model size, and Appendix K.6 reports the corresponding service-rate thresholds.

Offline costs must also be accounted for. Let ΔC_{off} be the incremental offline arithmetic cost relative to a comparator, and let a deployed policy save $\Delta C > 0$ operations per question at matched accuracy. The amortization point is then $\lceil \max(0, \Delta C_{\text{off}})/\Delta C \rceil$ questions. A small control head does not make collection inexpensive, because generation dominates collection cost. The upper-bound collection calculation in Appendix C excludes baseline labeling, so a comparative deployment ledger must include the relevant offline costs of both methods. Appendix K.6 provides explicit service-rate, memory, and incremental-cost sensitivities without asserting hardware measurements.

J Mechanism-identifying comparisons and training refinement

The comparisons in this section test reward targets and selector effects for Contribution 1, and computation placement and fitting quality for Contribution 2. Each design specifies its controlled factors and resource normalization. Persistent-prefix search and centered fitting are evaluated as separate extensions of the primary 768-score matrix, and the selector example gives an exact analytical counterpart to the fixed-bank selector experiment.

J.1 Crossing search domain with the learning target

Question and intervention. The factorial asks whether the selected-correctness target of Contribution 1 improves accuracy independently of historical-prefix access. Figure 4 crosses incumbent versus persistent search with coverage versus selected-output rewards. All cells share the six-step fitted backups, width-256 action network, generator, frozen outcome selector, price-selection procedure, and $2.4C_0$ inference operating point. Neither a one-step coverage baseline nor a smaller network can therefore create the target contrast.

For coverage training, define $U_t = \mathbf{1}\{\mathcal{Z}_t \text{ contains a correct candidate}\}$ and use reward $U_{t+1} - U_t - \lambda c_t$. For selected-output training, use $Y_{t+1} - Y_t - \lambda c_t$. Each reward telescopes to its own terminal quantity. The auxiliary targets follow the objective: we write $Z_t = U_t$ in coverage cells and $Z_t = Y_t$ in selected cells, fit reliability, repair, and damage on Z_t, Z_{t+1} throughout, and use the corresponding gain in the one-step consistency term. In a growing pool, coverage damage is identically zero. Replacing only the Bellman reward while keeping selected-output auxiliary consistency would produce incompatible targets, so this mixed objective is excluded. Training-only correctness annotations determine U and Y, whereas the evaluated output is always the frozen selector's choice, so no oracle selector is used at inference. Once the pool contains a correct candidate, the coverage objective can prefer stopping even when the selector still returns an error. This is exactly the distinction being tested.

The incumbent domain proposes four positions from the current selected trace. The persistent domain additionally admits positions from the four most recently inserted valid historical candidates, including candidates that are no longer incumbent; duplicates are merged and the incumbent is always retained. A persistent action records the source candidate, retained position, and suffix cap. The source endpoint fills the same endpoint-feature slot as in the incumbent-only representation, so

network dimension and parameter count are unchanged, and pool summaries continue to describe the whole retained pool. Candidate storage, replay, process scoring, and the larger batched evaluation are charged. This bounded historical-prefix controller is evaluated as a separate adaptation of PB-SMC.

All four fits share one collection-parent mixture, with equal proportions of parents reached by the incumbent and persistent collection policies and no correctness filtering, and every eligible action has positive collection probability. Within each search domain, the two reward fits use identical sampled transitions relabeled by their respective rewards. Across domains, generator-operation allowances are equal: the larger persistent action set receives a recorded allocation of the same budget rather than extra offline generation. For parent S with A_S eligible actions and action inclusion probabilities $\pi_a > 0$, the loss is $A_S^{-1} \sum_{a \in \text{sample}} \ell_a / \pi_a$, followed by an equal-parent average. The π_a are inclusion probabilities of the recorded sampling scheme, and weights are not normalized by their random realized sum, so the subsampled loss is unbiased for the full eligible-action average. The joint collection specification applies only to this factorial; other baselines retain their documented protocols. The Incumbent/Selected cell is refitted under this collection distribution at the same expected-cost point. Its small score differences from the primary row reflect the changed collection distribution and optimization seed, so it is a separate entry, not a reused controller.

Results. Replacing the coverage reward with the selected-output reward raises Qwen3-8B's math/code accuracy by $+1.7/+2.8$ points in the incumbent domain and by $+1.2/+2.2$ in the persistent domain. Under the selected-output objective, expanding the search domain adds only $0.2/0.5$. Qwen3.8-27B shows the same pattern: target effects of $1.4/2.5$ and $1.1/2.2$ in the two domains, and search increments of $0.2/0.8$. For both generators, the target effect is positive in both search domains and larger than the search increment, supporting Contribution 1: the selected-output target adds accuracy that historical-prefix access does not supply. The interaction is nonadditive, because a broader search domain partly substitutes for one source of improvement while still benefiting from the target.

J.2 Natural parents, compute allowances, and cache removal

These controls test the placement component of Contribution 2 under matched information and resources. **Population.** The natural-parent diagnostic contains 512 held-out math questions and 512 held-out code questions, each with two independently generated initial attempts, for 2,048 parent states in total. Questions are drawn from unused training-source distributions and exclude controller training, calibration, validation, all primary benchmarks, and the original diagnostic set. Initial attempts follow the 4,096-token generation rule, and no parent is removed for being correct, wrong, malformed, or hard to repair. Initial math/code accuracies are 75.3% and 52.1%; this diagnostic population is distinct from MATH-500 and LiveCodeBench. All policies are fixed on separate validation parents before evaluation.

Two resource interventions. The fixed-suffix condition allows exactly one reopening with cap 1,024 and the same final-answer reserve at every position. Root discards the entire initial solution, whereas learned position choice retains its chosen prefix. This condition therefore fixes the additional generation length, not total operations or total useful text.

The common-envelope condition instead assigns each parent an incremental operation ceiling $B_{\text{add}}(S) = 0.75 C_{\text{init}}(S)$, where $C_{\text{init}}(S)$ is its already incurred initial-generation and initial-selection cost. The rule uses no correctness labels and is identical across methods. An action is feasible only if its worst-case incremental replay, decoding, selection, and control charge fits within this allowance. Positions range over the same legal set and suffix caps over $\{256, 512, 1024, 2048, 4096\}$, and STOP remains available when no action fits. Root receives the longest feasible cap, whereas learned policies may choose any feasible position–cap pair or stop. Because changing a completion cap changes the generation and termination distributions, the one-step RAVEC fit collects separate training-parent interventions for the additional 256- and 4,096-token caps. These additional training rollouts are shared with the root value control and charged as diagnostic supervision, and the price is selected on validation parents. The root value control receives the same projected full trace, discarded suffix, incumbent and pool state, confidence, resource features, training questions, supervision volume, network width, and STOP rule as RAVEC one-step. It differs only in its feasible action set: root crossed with the suffix-cap grid plus STOP, instead of all

legal root and non-root positions crossed with that grid plus STOP. The comparison thus removes only non-root position choice. PAV uses its documented process-advantage position rule and the longest feasible suffix, with its process-scoring cost charged.

The initial attempt is a common sunk cost, so every method has $C_{\text{init}}(S) + B_{\text{add}}(S)$ total available operations on that parent. A shared allowance does not force identical realized operations: early completion, STOP, and discrete feasible caps leave slack. These rows therefore compare accuracy under the same per-parent resource constraint, not along an equal-spend frontier; the equal expected-spend comparison is provided separately by the main $2.4C_0$ curves.

The cache-removal condition replays all required generator context from token zero, including the question/chat prompt, keeps the same numerical $B_{\text{add}}(S)$, recomputes the feasibility mask, and charges the actual replay. Root can change slightly because question-prefix cache reuse is removed, though it never retains a solution prefix; all methods obey the same semantics. The final dose-response row doubles only root's $B_{\text{add}}(S)$ while keeping the initial attempt fixed, giving a total ceiling of $2.5C_{\text{init}}(S)$ rather than twice the original total cost.

Table 17: One-step intervention controls on naturally sampled parents (Contribution 2). Initial accuracies are 75.3 and 52.1 percent. Root value control shares the RAVEC full-trace, incumbent, pool, confidence, and resource features, supervision, network width, and STOP rule, but is restricted to root positions. Within every protocol, one-step RAVEC exceeds all root-restricted rules and PAV positions on both domains, including under full replay without cache. The fixed-suffix comparison spends different total operations; common-envelope comparisons charge full initial generation, context processing, selection and control. The last row, a dose-response control, doubles the additional allowance and brings root reopening roughly to the common-envelope RAVEC level.

Protocol	Position rule	Math	Code
Fixed suffix cap	Root	82.1	58.5
Fixed suffix cap	RAVEC one-step	85.0	61.6
Same operation envelope	Root	82.7	58.9
Same operation envelope	Root value control	83.2	59.5
Same operation envelope	PAV position	83.6	59.8
Same operation envelope	RAVEC one-step	84.9	61.4
No cache; same operation envelope	Root	82.4	58.7
No cache; same operation envelope	Root value control	82.5	59.0
No cache; same operation envelope	RAVEC one-step	84.3	61.0
Root only; twice additional envelope	Root	84.8	61.5

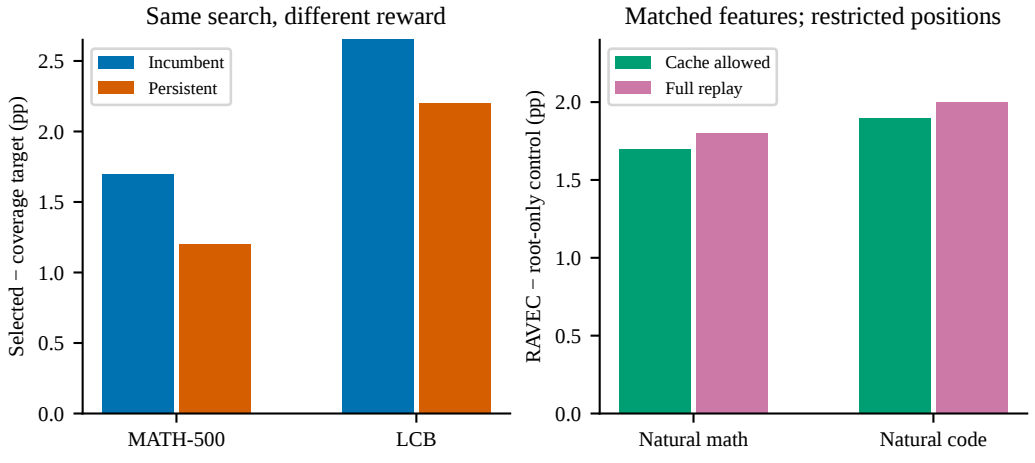


Figure 7: Left: accuracy change from replacing the coverage reward with the selected-output reward in the incumbent and persistent search domains (Contribution 1). Right: one-step RAVEC position control versus a matched-feature root-only control that differs only in its action space, under the same per-parent total-operation envelope, with and without caches (Contribution 2). Diagnostic parents differ from the main benchmark population.

Results and interpretation. With equal suffix caps, RAVEC exceeds root by $2.9/3.1$ math/code points. Under equal per-parent operation allowances, it exceeds the longest-feasible root rule by $2.2/2.5$ points and the matched-feature root-only control by $1.7/1.9$ points. Under full replay without caches, the corresponding gains are $1.9/2.3$ and $1.8/2.0$. Doubling root's incremental allowance nearly closes the gap. The matched-feature contrast isolates a position-choice advantage with trajectory information, learned length selection, and STOP equalized, supporting Contribution 2; the larger gap to the fixed root rule also includes allocation quality. The advantage concerns efficient placement under a bounded budget; retained text and its semantic content jointly define the intervention.

J.3 Exchanging the selector on a fixed generated bank

This diagnostic tests Proposition 1 and hence Contribution 1: changing only the selector on a fixed generated bank should change the preferred position. **Construction and split discipline.** We use the first initial parent of each of the 1,024 natural-diagnostic questions, giving 512 math and 512 code parents. Each parent has its original single candidate and up to four distinct legal 1,024-token reopening positions. Invalid or very short initial attempts remain in the denominator and may admit only root; when no alternative position exists, they contribute zero to position-reversal counts. The 64-suffix bank of each parent–position pair is split into 32 action-discovery samples and 32 independent evaluation samples, for at most $1{,}024 \times 4 \times 64 = 262{,}144$ generated suffixes, the design maximum before duplicate checkpoint merging. Initial correctness, generated text, valid-candidate flags, and reference labels are shared by all selector conditions.

Each bank sample defines a two-candidate choice between the initial attempt and its suffix completion; the 64 suffixes are not pooled into a best-of-64 answer. Averaging these choices estimates the one-intervention selected gain. The replacement 4B ORM uses the same selector-training questions, candidate responses, correctness targets, and validation protocol as the original 0.6B ORM, and its scoring and training operations are counted separately. Rescoring a bank requires no new generator sample, but the replacement selector is not free.

For each parent, let a be the best position on the action-discovery bank under the original selector and b the best under the replacement. A reversal is counted only if three conditions hold: $a \neq b$; on the independent evaluation bank, the original selector gives a a gain advantage exceeding 0.02 over b; and, on the same bank, the replacement selector gives b an advantage exceeding 0.02 over a. The denominator includes all parents, including ties and near-ties. The threshold is an operational separation rule for this diagnostic, and the finite banks determine its sampling variability.

To estimate the recoverable decision loss, we fit one position controller to original-selector targets and a second to replacement-selector targets on an independent training bank, freeze both before the diagnostic, and evaluate both under the replacement selector on the same held-out evaluation bank. "Regain" is their selected-accuracy difference under the replacement selector. The preferred position reverses for 23.1% of parents, and retargeting to the replacement selector regains $1.7/2.4$ math/code points. These results link the selector gate to a change in the preferred intervention, beyond the accuracy effect of the larger selector itself, as Proposition 1 predicts.

Table 18: Selector exchange on a fixed generated bank. Reversal is the percentage of parent states whose preferred position changes; regain is the selected-accuracy gain, in points, from refitting position choice to the replacement selector. Replacing the 0.6B ORM with the 4B ORM reverses preferences and yields positive regain in both domains, whereas a strictly increasing temperature transform changes neither candidate ranks nor selected outputs. Generator and pool are held fixed throughout.

Source	Replacement	Reversal (%)	Math regain	Code regain
0.6B ORM	0.6B ORM	0.0	0.0	0.0
0.6B ORM	4B ORM	23.1	1.7	2.4
0.6B ORM	Monotone temperature transform	0.0	0.0	0.0

The unchanged-selector and strictly increasing score-temperature conditions are analytical negative controls: with the bank, fitted choices, and tie handling held fixed, every terminal selection is identical, so reversal and regain are exactly zero. These zeros are identities rather than three independently measured outcomes, consistent with the invariance statement of Proposition 1. The temperature con-

trol is not permitted to retrain a stochastic online policy, nor to feed rescaled scores into an otherwise unchanged policy while invoking terminal-selection invariance, because rescaling can change such a policy's input even when candidate rankings are unchanged.

Table 19: Exact probability construction of the selector-gating reversal (Proposition 1) with 75 percent initial reliability. Generated candidates and their correctness distribution are identical across selectors, so coverage is selector invariant; nevertheless, the better action is Late under the weak selector and Early under the strong one. Values follow the displayed probability equations exactly.

Reopen	Selector	Coverage (%)	α (%)	β (%)	Net gain (pp)
Early	Weak	90.00	30.0	5.5	3.375
Early	Strong	90.00	55.0	1.2	12.850
Late	Weak	86.25	36.0	1.5	7.875
Late	Strong	86.25	40.0	1.0	9.250

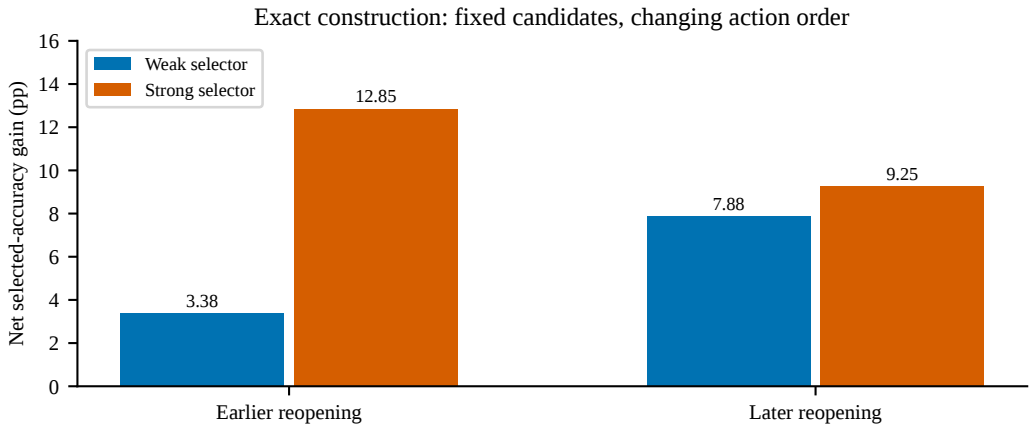


Figure 8: Exact analytical construction from Appendix B.1 (displayed labels rounded). The two selectors face identical generated-candidate distributions yet prefer different reopening actions, the reversal predicted by Proposition 1 and observed empirically in Table 18.

J.4 Learning relative values without additional generation

This refinement of Contribution 2 sharpens within-parent action ordering by reusing existing outcomes. **Fit and selection.** The centered extension adds $\eta\mathcal{L}_{\text{ctr}}$ to the existing probability-fitting objective and retains all value losses and calibration. Minibatches group transitions by parent and eligible action set. Initial parents use two stored outcomes per action; aggregation parents use their single available outcomes. Each d_a averages only its own parent's action repetitions, and the $1/A$ normalization prevents parents with more legal positions from dominating; parents with a single available action contribute zero centered loss. The primary centered ablation uses the original full-action collections, not the factorial's subsampled action protocol, which is a separate extension.

The validation grid is $\eta \in \{0, 0.05, 0.2, 0.5\}$; the value 0.2 is chosen once from Qwen3-8B validation action regret and then reused for all four generators and their test tasks without per-model retuning. Generator calls, the number of training transitions, controller dimensions, inference passes, and the inference operation allowance are all unchanged; the extra vector arithmetic in fitting is disclosed as training overhead. Gradients act on the gain $\hat{g} = (1-\hat{r})\hat{\alpha} - \hat{r}\hat{\beta}$ at each action; they neither center the absolute Q value nor mix different remaining horizons. After probability calibration, only the linear Q readout is refit with the resulting one-step consistency targets, while the projection and shared hidden layers remain frozen, as in the original method.

Action-level diagnostic. Figure 11 uses the original calibration-diagnostic population and its independent banks, not the natural-parent population above. "Calibration diagnostic" names an evaluation of calibration quality; these held-out questions are disjoint from the 800 questions used to fit temperature parameters. All estimators score the same eligible position grid at a fixed 1,024-token

cap. The diagnostic sets the computation price to zero and ranks selected-correctness gains; multistep cost-adjusted regret is evaluated separately. The question/resource and confidence/length fits include action position and cap but omit path-state representations, so their predicted ordering can still vary across positions although question features stay fixed.

Within each parent, Kendall τ_b is computed between predicted gains and action-discovery bank means, accounting for ties; a completely tied prediction vector receives score zero rather than being dropped. Regret selects the reference action on the discovery bank and estimates, on the independent evaluation bank, the selected-gain gap between that fixed reference and the learned action. It is thus regret relative to a finite-bank-selected reference, and signed gaps are retained rather than clipped. Parent metrics are averaged first within question and then across questions, so repeated branches do not inflate the statistical unit.

The reference action and uniform policy are shared with Table 15, so differences in regret must exactly equal reversed differences in gain over uniform: $R_m - R_{\text{RAVEC}} = G_{\text{RAVEC}} - G_m$. For example, PAV's math regret is $2.2+(8.3-4.6) = 5.9$ points and its code regret is $2.7+(7.1-4.1) = 5.7$, and the question and confidence controls obey the same identity. The shared reference fixes this relation across all compared estimators.

Results. Relative to the original fit, the centered refinement reduces Qwen3-8B's math/code action regret from $2.2/2.7$ to $1.4/2.2$ points and raises τ_b from $0.40/0.41$ to $0.48/0.46$, consistent with the within-parent ordering improvement that Proposition 2 anticipates. At the original operating point, its four-task macro increments are 0.70, 0.33, 0.58, and 0.80 points for Qwen3-8B, Qwen3.8-27B, R1-Distill-Qwen-14B, and Gemma-3-12B, respectively. Fourteen of the sixteen task-level changes are positive, one is unchanged, and one is -0.1 point; the changes range from -0.1 to 1.4 points, and the largest gains occur on different mathematical and coding tasks. Macro gains are thus positive for all four generators, although task-level gains are small and uneven. The centered variant is reported separately from the original $2.4C_0$ matrix, and its gain is not added to the persistent-search result.



Figure 9: Task-level accuracy of parent-centered fitting versus the original fit at the same $2.4C_0$ operating point, for four generators. Each cell reports both accuracies and their signed difference, and color encodes the difference on a common scale. The loss adds no parameters, inference calls, or generated supervision.

J.5 Longer initial reasoning and aggregate sensitivity

A 4,096-token initialization may leave incomplete or rushed answers that are easy to repair, so Table 20 repeats the PB-SMC and RAVEC comparison at 8,192 and 16,384 initial tokens using model-native reasoning formats. The total generated-token caps increase to 20,480 and 28,672, respectively, preserving the same 12,288-token allowance beyond the maximal initial length, and at most six reopenings still occur. The hard operation cap is $16C_0$ in each row. Parent collection and controller fitting use the corresponding initial policy, not an unchanged out-of-distribution short-

trace controller, and comparator validation and offline operation allowances are matched within each row.

C_0 is recomputed for each model, task, and initial policy, so every row compares methods at $2.4C_0$ expected cost while absolute operations differ across rows. At the largest initial cap, Qwen3-8B's RAVEC–PB gap contracts from $1.5/2.0$ to $1.1/1.2$ math/code points, and Qwen3.8's from $1.5/2.5$ to $1.0/1.1$. The advantage over PB-SMC thus remains positive for both generators at every tested initial length, although it shrinks as repairable headroom contracts and varies with capability and reasoning length.

Table 20: Initial-answer budget sensitivity, RAVEC versus PB-SMC. Each row uses $2.4C_0$ with C_0 recomputed for that initial policy. For 8,192 and 16,384 initial tokens, total decoded-token caps increase to 20,480 and 28,672; the six-reopening limit is retained. RAVEC exceeds PB-SMC in every row on both tasks, with smaller margins at the two longer caps. Compare methods within a row, not absolute costs across rows.

Model	Initial cap	PB MATH	RAVEC MATH	PB LCB	RAVEC LCB
Qwen3-8B	4096	97.3	98.8	54.4	56.4
Qwen3-8B	8192	98.0	99.1	57.2	59.0
Qwen3-8B	16384	97.8	98.9	59.4	60.6
Qwen3.8-27B	4096	97.8	99.3	82.5	85.0
Qwen3.8-27B	8192	98.4	99.1	84.8	85.6
Qwen3.8-27B	16384	98.2	99.2	84.6	85.7

Table 21 derives four summaries directly from the same 14-model matrix. Removing both AIME sets avoids assigning two of eight core weights to 30-item competitions. Domain balancing then assigns one third each to mathematics (MATH-500 and OlympiadBench), coding (HumanEval+, MBPP+, and LiveCodeBench), and science (GPQA), and the all-12 mean is a separate reporting convention. None of these summaries was selected using a significance test, and no alternative average replaces the original macro. For Qwen3-8B, the no-AIME gain is 1.05 points, so its positive aggregate does not depend on the two AIME entries.

Table 21: Aggregation sensitivity of the RAVEC-minus-PB-SMC gain (points), derived from the primary matrix. No AIME averages the six remaining core tasks; Domain balanced weights the math (MATH/Olympiad), code (HE+/MB+/LCB), and science (GPQA) means equally; All-12 weights all tasks equally. The gain is positive for every model under all four aggregations, although its magnitude depends on the weighting.

Model	Core-8	No AIME	Domain balanced	All-12
Qwen3-4B	1.89	1.30	1.02	1.33
Qwen3-8B	1.28	1.05	0.77	0.87
Qwen3-14B	0.96	0.60	0.76	0.84
Qwen3-32B	0.91	0.70	0.80	0.75
Qwen3.5-9B	0.86	1.25	0.92	0.70
Qwen3.5-27B	1.76	1.12	0.87	1.40
Qwen3.6-27B	1.18	0.98	0.66	0.75
Qwen3.8-27B	1.39	1.08	0.99	0.99
R1-Distill-Qwen-7B	1.35	0.93	0.65	1.09
R1-Distill-Qwen-14B	1.79	1.72	1.42	1.35
R1-Distill-Qwen-32B	1.03	0.62	0.53	0.80
Gemma-3-12B-IT	1.46	1.68	1.83	1.29
Gemma-3-27B-IT	1.38	1.30	0.96	1.08
Llama-3.1-8B-Instruct	1.59	1.05	1.37	1.35

Table 22 applies the paired nested protocol to every generator. The core-gain intervals over PB-SMC exclude zero for twelve of the fourteen models, and all fourteen point estimates are positive; the unweighted cross-model mean and median are also positive. This supports a broad positive effect across the evaluated models (Contribution 3) while preserving model-level heterogeneity.

Table 22: Cross-model uncertainty for the eight-task core gain of RAVEC over PB-SMC. All fourteen point estimates are positive, and twelve nested intervals exclude zero. Points are recomputed from the primary matrix. Nested intervals resample paired question clusters and controller fits under the same protocol as Figure 2; fit SD is the standard deviation across the five controller fits.

Model	Core gain	Nested 95% interval	Fit SD
Qwen3-4B	1.89	[1.18, 2.42]	0.20
Qwen3-8B	1.28	[0.79, 1.76]	0.16
Qwen3-14B	0.96	[0.31, 1.44]	0.19
Qwen3-32B	0.91	[0.58, 1.82]	0.17
Qwen3.5-9B	0.86	[-0.03, 1.46]	0.21
Qwen3.5-27B	1.76	[1.20, 2.63]	0.23
Qwen3.6-27B	1.17	[0.39, 1.57]	0.20
Qwen3.8-27B	1.39	[0.68, 1.69]	0.15
R1-Distill-Qwen-7B	1.35	[-0.08, 1.91]	0.24
R1-Distill-Qwen-14B	1.79	[1.05, 2.46]	0.22
R1-Distill-Qwen-32B	1.03	[0.35, 1.55]	0.18
Gemma-3-12B-IT	1.46	[0.67, 2.08]	0.25
Gemma-3-27B-IT	1.37	[0.72, 2.20]	0.23
Llama-3.1-8B-Instruct	1.59	[1.02, 2.57]	0.27

J.6 Offline supervision volume as a separate resource

This section separates RAVEC's target quality from offline supervision volume. The primary PAV comparison matches labeling-generation operations, accounting for PAV's distinct 4B prover and RAVEC's 8B intervention generator. Table 23 additionally matches *generated labeling tokens*. The normalizations answer different questions, since a smaller-prover token need not cost as much as a target-generator token; neither substitutes for the other.

At each token allowance, both methods draw from the same training-question sources, preserve both correct and incorrect parent strata, and spend the allowance on their own target construction. Nested question subsets, followed by action-repetition allocation, follow a training-source ordering independent of labels, and token expenditure includes labeling rollouts that terminate unsuccessfully. The shared outcome selector uses the same separately accounted training set. Models and policies are refitted at each allowance, while the inference point remains $2.4C_0$. The endpoint of 1,641.6 million tokens equals the upper-design count of 1.824 million transitions times a 900-token mean suffix length. Initial-parent and selector collection costs remain separate from this suffix-token axis.

Table 23: RAVEC versus PAV at matched labeling-token budgets, in millions. The common ORM is accounted for separately. RAVEC is higher than PAV at every budget on both tasks, although its LCB accuracy is not monotone in the budget. Because the provers differ in size, equal token counts differ from equal operation counts; the original matched-operation comparison remains the primary protocol.

Label tokens (M)	PAV MATH	RAVEC MATH	PAV LCB	RAVEC LCB
150	95.2	95.8	50.4	51.3
450	96.4	97.2	52.3	54.1
900	96.9	98.1	53.1	53.9
1641.6	96.8	98.8	53.8	56.4

With matched labeling tokens, the RAVEC–PAV math/code gap widens from $0.6/0.9$ points at 150 million tokens to $2.0/2.6$ at the largest allowance. At that endpoint, PAV reaches 96.8/53.8, slightly below its primary equal-operation result of 97.0/53.8, because matching 4B-prover tokens to 8B-generator tokens gives the prover less labeling compute than the equal-operation allowance, so the two normalizations yield distinct endpoints. The centered refinement is excluded so that the curve isolates the original intervention objective. Together with the equal-operation comparison and the shared-data centered ablation, this analysis separates target quality from additional supervision expenditure and indicates that the advantage stems from the former, supporting Contribution 1.

K FROM DESIGN CHOICES TO INTERPRETABLE EVIDENCE

This appendix maps each technical contribution to a measurable quantity. For Contribution 1, selector acceptance determines how much generated repair and damage reaches the returned answer; for Contribution 2, within-parent action regret measures how well the controller chooses where and how much to compute. Service-rate and sampling calculations characterize deployment cost and uncertainty of the Contribution 3 evaluation under explicit analytical inputs; exact identities are kept separate from aggregated or sensitivity-analysis quantities.

K.1 A RUNNING EXAMPLE AND THE ALTERNATIVES IT SEPARATES

For $\sqrt{x+2} = x$, squaring gives $x^2 - x - 2 = 0$, which yields the algebraic candidates $2, -1$. Because the original equation requires $x \geq 0$, only 2 is valid. Consider an illustrative initial trace in three parts: correct squaring and factorization, enumeration of both roots, and an incorrect final answer that contains both. A checkpoint placed before the final answer preserves the valid computation and regenerates only the missing domain check, whereas root resampling discards that valid work. A second trace first substitutes $u = x + 2$ but then writes $u = x^2 + 2$, introducing an earlier inconsistency; the same late checkpoint now preserves faulty algebra. The two constructions differ only in where the defect lies, which is what a position-aware controller must resolve. Because a model can still self-correct a retained inconsistency, the construction does not assume an irreversible semantic blocker.

Suppose the original two-root answer has selector score 0.81, a corrected completion scores 0.78, and another confident two-root answer scores 0.87. These illustrative scores specify one possible selector failure: the correct completion is rejected, and the higher-scoring wrong completion cannot repair the output. Conversely, if the incumbent already returns only 2, an accepted two-root completion damages correctness. Identical candidate correctness thus carries different control value depending on the incumbent and the selection event, the gap that motivates Contribution 1. The score example is explanatory only; the exact probability example in Table 19 separately supplies a mathematically complete action-order reversal.

Table 24: Design logic of RAVEC. Each row pairs a design choice with the problem it addresses, the most direct alternative, and the controlled comparison in this paper that distinguishes the two mechanisms. The first two rows serve Contribution 1 (selection-conditioned target); the remaining rows serve Contribution 2 (path-informed control).

Choice	Problem addressed	Direct alternative	Discriminating comparison
Selected repair and damage	Correct generations can be rejected; errors can displace correct answers	Coverage or continuation success	Search–target crossing and fixed-bank selector exchange
Same-parent interventions	Question difficulty obscures action effects	Pooled outcome prediction	Within-parent ranking; natural-parent length/confidence control
Position and suffix cap	Good retained work and remaining repair effort vary separately	Root with its longest affordable cap	Common operation ceiling, with and without replay
Finite-horizon value	An action may improve a later decision without immediate gain	Greedy net value; gain/cost	Shared-target controls with charged control costs
Centered plus absolute losses	Action order and STOP use different information	Direct gain; centered contrasts alone	Same-data regret comparison and offset invariance test

Role of the probability factorization. A sufficiently expressive direct gain predictor can learn the same conditional mean as Eq. (2); the two parameterizations therefore access the same target information. The factorization is instead justified by what it makes explicit: it exposes the two correctness branches, renders the damaging-selection mechanism testable, and supplies proper probability losses on the examples that support each branch. The direct-gain comparison measures its finite-data estimation benefit. Dense-prefix controls separately examine the resolution of the finite action grid; the default grid is a bounded control design, not a claim of optimal discretization.

K.2 MEASURING WHAT THE SELECTOR GATE PERMITS

This subsection tests the selector-gating mechanism behind Contribution 1. Table 25 decomposes the existing fixed-1,024-token natural-parent comparison, in which one-step RAVEC reaches 85.0/61.6% math/code accuracy against 82.1/58.5% for Root. Population, initial answers, and policies follow Appendix J.2, with no additional benchmark population. The diagnostic performs exactly one intervention and has no learned STOP decision. RAVEC chooses a legal position, and malformed parents remain eligible under the root fallback. No shorter suffix cap is silently substituted to match operations; the equal-operation comparison is reported in a separate row of Table 17.

The quantities $q_0, q_1, \kappa_+, \kappa_-$ use the definitions of Eq. (14). Here they are pooled conditional probabilities under the given policy, not unweighted averages of per-parent ratios. Before the ratios are formed, weights are assigned hierarchically: equal per question, equal across its two parents, and equal across continuations within a parent. Invalid generations have $Y_a = 0, K_a = 0$. Because all factors are computed from the same weighted joint distribution, the law of total probability guarantees

$$\mathbb{P}(Y_1 = 1) = r + (1 - r)q_0\kappa_+ - r(1 - q_1)\kappa_-. \tag{29}$$

For each continuation, the diagnostic records initial correctness, new-candidate correctness, score acceptance, and actual selected correctness. The last label is observed directly and serves as a closure check, rather than being predicted by multiplying independently fitted heads. The accompanying table expresses transition and gate frequencies as normalized count equivalents per 10,000 parents; this display denominator is not an observed sample size. The factor columns summarize the corresponding conditional probabilities. All four factors are offline diagnostics; correctness never becomes an online input.

Table 25: Gate-factor decomposition of Root versus one-step RAVEC reopening in the fixed-cap natural-parent diagnostic. All entries are percentages; Repair and Damage are unconditional probability masses in percentage points. Initial accuracy is 75.3% (math) and 52.1% (code). q_0 ($1-q_1$) is new-candidate correctness (error) given an incorrect (correct) incumbent; gates are conditional probabilities of beating the incumbent score. Output = initial + Repair − Damage. In both domains RAVEC raises q_0 and κ_+ and hence Repair; Damage is unchanged on math and slightly higher on code. Gate columns are derived and rounded. Suffix caps are fixed, so total operations are not equalized.

Domain	Policy	q_0	κ_+	$1 - q_1$	κ_-	Repair	Damage	Output
Math	Root	37.0	86.4	16.0	9.1	7.9	1.1	82.1
Math	RAVEC	47.0	93.0	14.0	10.4	10.8	1.1	85.0
Code	Root	30.0	56.4	39.0	8.4	8.1	1.7	58.5
Code	RAVEC	37.0	65.5	36.0	11.2	11.6	2.1	61.6

Table 26: Normalized gate-event frequencies for the fixed-cap diagnostic, expressed as rounded count equivalents per 10,000 parents, not an observed sample size. Candidate correct counts correct continuations of initially wrong parents; Candidate error counts incorrect continuations of initially correct parents, before selector gating. W→R and R→W count the corresponding selected-answer transitions. RAVEC raises repair in both domains; damage is unchanged on math and higher on code. Final accuracy is a percentage. The display denominator is not used for statistical inference.

Domain	Policy	Scale	Candidate correct	Candidate error	W→R	R→W	Final (%)
Math	Root	10,000	914	1205	790	110	82.1
Math	RAVEC	10,000	1161	1054	1080	110	85.0
Code	Root	10,000	1437	2032	810	170	58.5
Code	RAVEC	10,000	1772	1876	1160	210	61.6

Relative to Root, RAVEC's advantage is carried mainly by accepted repair. On math, repair mass rises from 7.9 to 10.8 points while damage stays at 1.1, which accounts for the full 2.9-point output increment. On code, repair rises from 8.1 to 11.6 points while damage rises from 1.7 to 2.1, yielding a 3.1-point output increment. A stronger policy can therefore accept slightly more damage when the additional repair is larger, a trade-off visible only because repair and damage are measured separately. Derived gate values are stored at full precision, so rounded columns need not multiply exactly at displayed precision.

An active gate intervention with an exact prediction. For a fixed parent and candidate bank, independently retain an otherwise accepted candidate with probability $t \in [0, 1]$; otherwise keep the incumbent. Write $B_t \sim \text{Bernoulli}(t)$, independent of the existing labels and scores, and $K_a^{(t)} = K_a B_t$. Then

$$\alpha_t = t\alpha, \quad \beta_t = t\beta, \quad g_t = tg, \quad \mathbb{P}(U_1^{(t)} = 1) = \mathbb{P}(U_1 = 1). \tag{30}$$

The proof conditions on the original bank and uses $\mathbb{E}[B_t] = t$; for this diagnostic output, candidates enter the coverage pool regardless of acceptance. The one-step randomized-selector diagnostic thus complements the deterministic maximum-score selector and isolates the gate effect without any regeneration. With $t = 1/2$, the fixed-cap RAVEC factors predict math output $75.3 + 9.7/2 = 80.15\%$ and code output $52.1 + 9.5/2 = 56.85\%$, with unchanged candidate coverage, so halving acceptance halves the selected-output gain.

Thinning deliberately violates the deterministic-selector assumption of Proposition 1, while the general identity $\Delta Y = (Y_a - Y_0)K_a^{(t)}$ remains exact. It isolates how the joint acceptance event mediates intervention value and gives the exact stopping threshold: if generation cost $\bar{c}$ is unchanged and $g > 0$, the one-step intervention beats STOP precisely when $t > \lambda\bar{c}/g$. A selector that discards more correct candidates can therefore shift the optimal stopping boundary even when generator quality is unchanged.

K.3 Connecting action diagnostics to the information theorem

This subsection links Contribution 2 to its empirical counterpart. Theorem 1 concerns the best measurable fine- and coarse-information policies, whereas the diagnostic in Figure 11 evaluates trained policies against a finite-bank reference. Its coarse confidence/length network uses the same actions, resources, and labels as RAVEC and differs only in omitting the path-state vectors. Denote that fitted policy by π_D, the learned fine policy by π_I, and the discovery-bank-selected reference by a_{ref}. On the common independent evaluation bank define

$$\widehat{R}_D = \overline{Y(a_{\text{ref}}) - Y(\pi_D)}, \qquad \widehat{R}_I = \overline{Y(a_{\text{ref}}) - Y(\pi_I)}. \tag{31}$$

The sample-level accounting identity

$$\overline{Y(\pi_I) - Y(\pi_D)} = \widehat{R}_D - \widehat{R}_I \tag{32}$$

holds whenever both regrets use identical weights and reference choices, including signed reference gaps. The identity motivates Figure 10 without identifying V_D with a trained comparator or V_I with a noisy oracle. Since the reference uses offline continuation labels beyond online information, its gap diagnoses only the specified population and action set. This fixed-cap, zero-price one-step diagnostic is also distinct from the regret of the deployed six-step controller.

The left decomposition in Figure 10 is an accounting identity, $Y = U - L_\rho$, not an additional mechanism experiment. The independent mechanism evidence comes from the fixed-bank selector exchange and the natural-parent position comparison; the decomposition makes the endpoint gap between RAVEC and PB-SMC auditable in terms of coverage and selection loss.

Table 27: Decision-gap decomposition (percentage points) in the independent-bank action diagnostic (path-informed control, Contribution 2). Coarse is the trained confidence/length policy; Ref. is the discovery-bank-selected action evaluated on a separate bank; the first column minus the second equals the third. RAVEC recovers most of the coarse-to-reference gap in both domains, and centered fitting narrows the remaining gap further. Gaps are relative to this finite-bank reference, not a true optimum.

Domain	Policy	Ref. − coarse	Ref. − policy	Policy − coarse
Math	RAVEC	8.7	2.2	6.5
Math	Centered	8.7	1.4	7.3
Code	RAVEC	7.7	2.7	5.0
Code	Centered	7.7	2.2	5.5

Against the coarse confidence/length policy, the original path-informed fit recovers 6.5/5.0 math/code points of reference gap, leaving 2.2/2.7 points to the bank reference; the centered fit recovers 7.3/5.5 points, leaving 1.4/2.2. Path information thus closes most of the coarse policy's gap

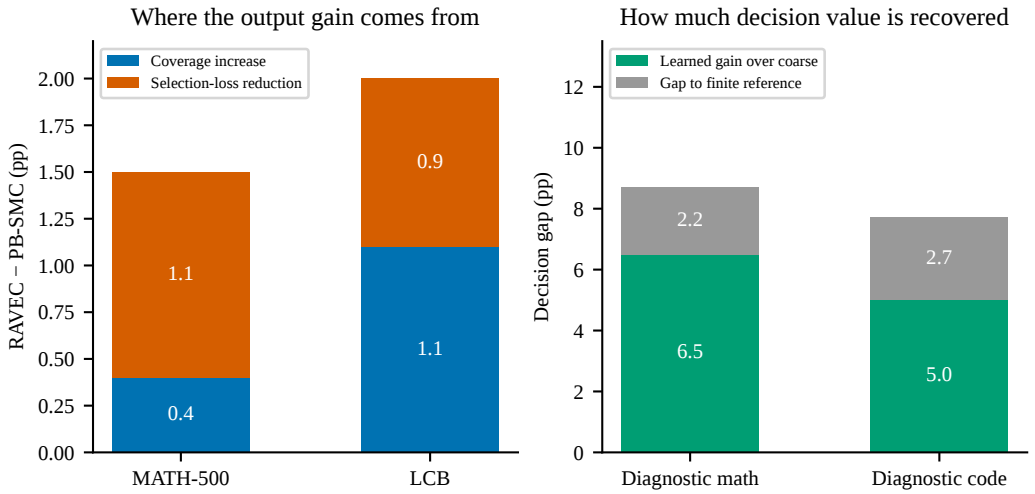


Figure 10: Evidence bridges for the two technical contributions. Left (Contribution 1): for Qwen3-8B at $2.4C_0$, the primary RAVEC–PB-SMC selected-accuracy gap is decomposed into coverage gain and reduction of the coverage-minus-selected loss. Right (Contribution 2): held-out one-step diagnostic with a fixed 1,024-token suffix cap and zero compute price, using the trained confidence/length policy as the coarse comparator. Each coarse-policy reference gap splits into realized RAVEC gain and residual gap to the finite-bank reference.

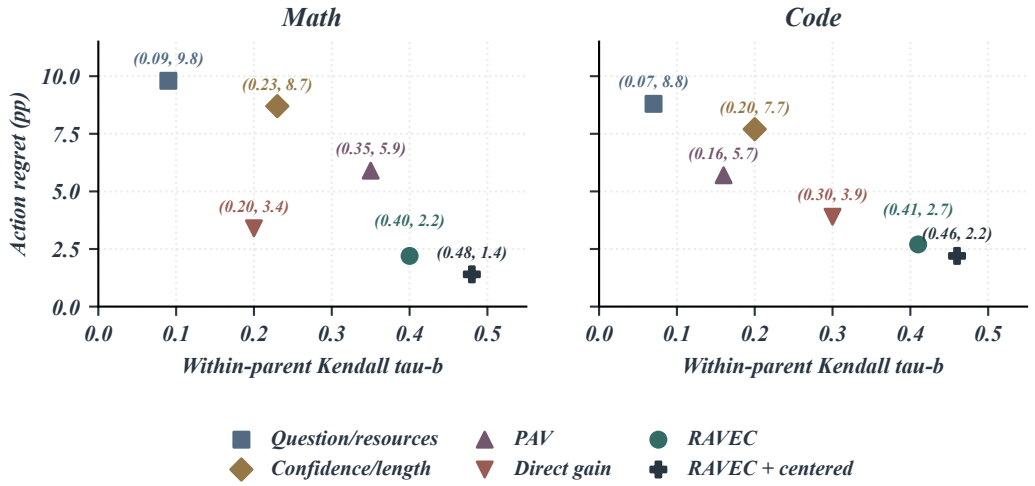


Figure 11: Within-parent action ranking (Kendall tau-b) and regret on the original held-out diagnostic population, testing whether path information improves the choice of computation (Contribution 2). Coordinates preserve the exact tau-b and regret values for math and code. All methods use the same parents and eligible actions; regret is computed on an independent evaluation bank and measured in percentage points. Points further to the lower right indicate better decisions.

in both domains, and centering narrows the residual further. Because all four gaps share the same reference, the 0.8/0.5-point reductions in diagnostic regret are arithmetic consequences of the corresponding action-gain improvements, not independent confirmations; the independent downstream benchmark changes are not forced to equal them.

Assumptions and measurable counterparts. Several theoretical conditions have direct empirical counterparts. The feasible mask belongs to D, so removing features cannot change the set of legal actions. Same-parent position differences probe whether coarse summaries conceal distinct best actions. Independent discovery and evaluation banks prevent a noise-selected maximum from being evaluated on its own samples. Temperature calibration evaluates the supported probability branches, while the theorem's uniform-error condition remains an explicit analytical criterion. A margin analysis would fix discovery-bank action gaps before evaluation and report evaluated action

regret within those strata. The numerical bridge establishes the operational identity and the expected ranking improvements, with the uniform-error and optimal-policy assumptions stated explicitly.

A mechanism-specific contrast for STOP. For any parent-dependent offset $b(I)$ shared by reopening actions,

$$C_A(\hat{\boldsymbol{g}} + b(I)\mathbf{1}) = C_A\hat{\boldsymbol{g}}. \tag{33}$$

The centered loss and the ranking among reopenings are therefore unchanged, whereas the one-step decision against STOP changes from $\max_a(\hat{g}_a - \lambda\hat{c}_a) > 0$ to $\max_a(\hat{g}_a - \lambda\hat{c}_a) > -b(I)$. Consider a purely analytical state with two net action values $-0.01, -0.02$, for which STOP is optimal. An offset $+0.015$ preserves their 0.01 gap but incorrectly replaces STOP with the first action. This offset invariance establishes why centered contrasts must be paired with absolute-value supervision for STOP. In deployment the corresponding absolute quantity is Q_λ, calibrated and refit under the stated finite-horizon procedure.

K.4 Generalization results as conditions for method choice

Each extension tests a different alternative explanation and thereby delimits when RAVEC should be preferred. Longer first attempts test whether truncation alone creates the repair opportunities: at the largest initial cap, Qwen3-8B's math/code advantage contracts from 1.5/2.0 to 1.1/1.2 points but remains positive. With a stronger Qwen3.8 generator, the advantage contracts from 1.5/2.5 to 1.0/1.1. The controlled capability-gap surface remains near zero, as predicted when moving a checkpoint cannot supply a missing rule. Together these trends support an advantage conditional on recoverable errors, not universal returns to repeated reasoning.

Domain transfer tests whether the decision representation travels with the surface. Joint training reaches 56.4% on LiveCodeBench (LCB), improving 2.7 points over math-only fitting and exceeding the math-only transfer result of 53.7% (Table 7); its MATH-500 score changes by only 0.1 point. Selector exchange is a separate shift: the generator's candidates remain unchanged while action preferences move, so new selector scores require refitting the target. These results identify joint domain training and selector-specific fitting as material components of the method.

K.5 Paired clustered uncertainty and analytical sensitivity

The primary evaluation follows Appendix D.3: 64 paired decoding replicates per question–method cell, 10,000 question-cluster bootstrap draws, and five independently initialized controller fits. For the RAVEC–PB-SMC accuracy gain, this protocol yields nested 95% intervals of $[0.79, 1.76]$ points on the core macro, $[0.40, 2.58]$ on MATH-500, and $[0.35, 3.62]$ on LiveCodeBench. Matched-accuracy operation savings are $20.8\,[14.2, 27.0]\%$ and $25.4\,[17.1, 33.0]\%$; their intervals redraw the paired accuracy–cost curves and repeat the interpolation in every bootstrap sample. All lower endpoints remain positive under this nested protocol.

The following calculation is a secondary stress test that assumes one paired Bernoulli trial per question; it is not the source of the primary intervals. Let $D_i = Y_{R,i} - Y_{B,i} \in \{-1, 0, 1\}$ for two fixed policies on the same independently sampled question and paired decoding trial. If $\delta = \mathbb{E}D_i$ and $p_{\neq} = \mathbb{P}(D_i \neq 0)$, then

$$\mathrm{Var}(D_i) = p_{\neq} - \delta^2, \qquad \mathrm{SE}(\overline{D}) = \sqrt{(p_{\neq} - \delta^2)/n}. \tag{34}$$

This follows from $D_i^2 = \mathbf{1}\{D_i \neq 0\}$. Table 28 applies the formula to three disagreement rates, taking the primary gaps as mean differences. Every rate satisfies the joint-distribution constraint $|p_R - p_B| \leq p_{\neq} \leq \min(p_R + p_B, 2 - p_R - p_B)$ for the primary marginal accuracies: with MATH-500 accuracies of 98.8% and 97.3%, feasible disagreement ranges from 1.5% to 3.9%, so the 2–3.8% inputs are feasible, and the same inputs are feasible for the 56.4% and 54.4% LiveCodeBench marginals. At 3% disagreement, the paired standard errors are about 0.77 math and 0.86 code points, roughly half of the corresponding differences of 1.5 and 2.0 points.

Three sources of variation remain distinct: repeated decoding characterizes stochastic outcomes conditional on a question, retraining changes the fitted policy, and sampling questions governs generalization to a task distribution. Because multiple parents and suffixes from one question form a cluster, the primary interval pairs policies, resamples whole questions, and separately incorporates

Table 28: Analytical paired standard error of the RAVEC-minus-PB-SMC difference under binary outcomes, with $n = 500$ math and $n = 400$ code questions. The mean differences are 1.5 and 2.0 points, and every listed standard error is smaller than the corresponding difference. This diagnostic is separate from the reported nested intervals.

Assumed disagreement (%)	Math paired SE (pp)	Code paired SE (pp)
2	0.63	0.70
3	0.77	0.86
3.8	0.87	0.97

independently fitted controllers; it never counts suffix repetitions as independent questions. A fixed public benchmark also supports a different estimand from a random population of tasks. Table 28 isolates the disagreement-rate assumption and does not replace the nested protocol.

K.6 From arithmetic savings to service time and deployment cost

Let C_B, C_R be complete expected operations at the same accuracy, R_B, R_R their effective arithmetic service rates under a specified execution regime, and H_B, H_R non-arithmetic delays, including uncovered scheduling, synchronization, and cache traffic. Define $q = C_R/C_B$, $v = R_R/R_B$, and $h_m = H_m/(C_B/R_B)$. Then

$$\frac{T_R}{T_B} = \frac{q/v + h_R}{1 + h_B}, \qquad T_R < T_B \iff v > \frac{q}{1 + h_B - h_R}, \tag{35}$$

when $1 + h_B - h_R > 0$. Effective rates cover the complete operation mix, so selector and controller arithmetic is counted once and only remaining non-arithmetic delays enter H. Queueing latency is excluded. In a single-request serial regime, reciprocal service time is the service capacity; in a concurrent batch regime, throughput additionally depends on batch size and scheduling and is not inferred from this ratio.

Table 29: Service-rate sensitivity for RAVEC relative to PB-SMC at matched 97% MATH-500 and 55% LCB accuracy. At equal effective rates, operation savings translate to 18.1% and 22.5% service-time savings; the lower-rate rows quantify the thresholds required to retain these savings. Non-arithmetic overheads are 0.04 (baseline) and 0.06 (RAVEC) times baseline arithmetic service time. Rates cover the complete operation mix, including verifier work, rather than token throughput.

$v = R_R/R_B$	Math time ratio	Saving (%)	Code time ratio	Saving (%)
1.00	0.819	+18.1	0.775	+22.5
0.80	1.009	-0.9	0.954	+4.6
0.65	1.229	-22.9	1.161	-16.1

The matched-accuracy operation frontiers give $q = 0.79167$ on math and $q = 0.74603$ on code. Under the explicitly assumed $h_B = 0.04, h_R = 0.06$, the corresponding rate thresholds are 0.80782 and 0.76126. At $v = 0.8$, math is about 0.9% slower while code retains a 4.6% service-time saving; at $v = 0.65$, both become slower despite using fewer operations. Operation savings thus become service-time savings only when v exceeds these thresholds; the cases isolate this dependency on sequential execution efficiency and are not GPU-second predictions for an unspecified implementation.

Memory is a separate constraint. For a conventional attention model, retained KV storage is

$$M_{\mathrm{KV}} = 2Ln_{\mathrm{kv}}d_h b\, N_{\mathrm{live}}, \tag{36}$$

where L is the number of attention layers, n_{kv} the number of KV heads, d_h the head dimension, b bytes per entry, and N_{live} the number of physically stored token positions across active cache branches. As an analytical example, $L = 32, n_{\mathrm{kv}} = 8, d_h = 128, b = 2, N_{\mathrm{live}} = 4096$ gives exactly 512 MiB. Four deep copies then use 2 GiB, whereas shared-prefix storage can reference a single copy until branches diverge. The example does not describe the architecture of any evaluated checkpoint. Model weights, selector memory, activations, allocator overhead, and active decode workspace are additional, and hybrid models require their recurrent-state term and actual attention-layer count. Total serving memory is thus determined jointly by model states, cache branches,

activations, allocator overhead, and controller state; persistent-prefix search can store more live states than incumbent-only reopening.

For deployment accounting, use incremental offline cost relative to the comparator,

$$\Delta C_{\text{off}} = C_{\text{off,R}} - C_{\text{off,B}}, \qquad N_{\text{break}} = \left\lceil \frac{\max(0, \Delta C_{\text{off}})}{C_B - C_R} \right\rceil, \tag{37}$$

provided $C_B > C_R$ and per-query operation savings remain stable. Offline cost comprises initial-parent and intervention generation, outcome labeling or code execution where relevant, selector fitting, controller fitting, hyperparameter trials, and reached-state recollection. Work shared by both methods cancels when identical, whereas transfer or selector refitting adds its own cost. Storage and CPU work are measured in separate physical units or through an explicit common conversion; the C_0 analysis below covers only comparable arithmetic charges.

Table 30: Analytical break-even question counts at which the per-question operation saving of RAVEC over PB-SMC at matched accuracy repays the incremental offline cost, rounded upward; counts scale with that cost. Normalization differs by task, so compare counts within a task. Shared selector training cancels when it is identical across methods.

Incremental offline cost (C_0)	Math questions	Code questions
100,000	213,334	140,625
500,000	1,066,667	703,125
1,000,000	2,133,334	1,406,250

At the matched-accuracy points, an assumed incremental cost of $500{,}000C_0$ is amortized after roughly 1.067 million math or 703 thousand code queries; these are separate task-specific normalizations. The break-even scale makes explicit how operation savings translate into total deployment cost once context lengths, architecture, baseline labeling, and fitting costs are supplied. Operation savings therefore provide the strongest total-cost benefit in repeated deployment, with the break-even query count available for task-specific planning.

K.7 Positioning against the closest specified alternatives

The comparison suite separates three sources of improvement: search access, supervised generation, and terminal selection. Existing comparators mainly target one of these sources: BoN and self-consistency enlarge coverage, PAV and process verifiers score continuation progress, Subpool and PB-SMC widen historical-prefix access, and difficulty or resource controllers allocate budgets from coarse summaries. RAVEC instead targets the post-selection effect of an intervention from a fixed parent. The factorial fixes search within each reward contrast, equal-operation and equal-label-token conditions examine distinct supervision budgets, and fixed-bank selector exchange holds generated candidates constant. Together with classical value-of-information and greedy-error analysis applied to this utility, these controls establish the paper's specific methodological distinction: RAVEC learns a selection-conditioned intervention surface and within-parent action contrasts for affordable prefix control, rather than a better candidate generator, step scorer, or search space.

L Limitations

RAVEC allocates computation through suffix replacement after a complete initial answer; arbitrary edits and long-horizon environment interaction lie outside this action space. The four-position grid trades search resolution for predictable cost, with a denser alternative evaluated in Table 12. The controller depends on the generator's hidden states and the frozen selector, so changing either component calls for refitting, and domain transfer can shift calibration. Reported operation savings describe arithmetic cost; deployment latency additionally depends on hardware utilization, batching, and cache traffic.

Table 31: Comparison map for the algorithmic specifications in this manuscript. Each row states a comparator's training signal and the comparison dimension it addresses. Citations identify the motivating methods; adaptation details are not attributed to the native algorithms. All online comparators share a frozen terminal selector where applicable. Equal downstream supervision does not equalize pretrained model capacity or historical pretraining cost.

Comparator	Training signal and control object	Search / information	Comparison dimension
Difficulty/resource control (Snell et al., 2024; Bulut, 2026)	Outcome-based budget allocation	Question and resource summaries, no path vectors	Path information beyond a strong coarse predictor
PAV adaptation (Setlur et al., 2024)	Change in prover continuation success	Prefix process values; charged prover/process work	Selected-output effect versus continuation progress
VGB adaptation (Rohatgi et al., 2025)	Process-score-guided backtracking	Imperfect verifier; charged scoring	Learned intervention values versus verifier-guided location
Subpool / PB adaptation (Tran et al., 2026)	Process-success scores for historical-prefix proposals	Persistent prefixes; charged replay and search	Search access and proposal policy; native SMC guarantees are not transferred
Coverage-target refit	Pool-coverage increments	Same network, search domain and intervention data	Reward target with architecture and data held fixed
Direct-gain refit	Selected ΔY and matching Bellman targets	Same hidden-state inputs and action set	Probability factorization, without claiming extra information
RAVEC	Selected repair, damage, absolute value and action contrasts	Incumbent grid; persistent and centered variants separate	Which affordable computation changes the returned answer

M Complete model–benchmark results

M.1 Complete Qwen3-8B control comparison

Table 32: Complete eight-task Qwen3-8B comparison of all methods underlying Table 1, including the controls of Appendix E. Non-Single methods are evaluated at $2.4C_0$; Single is evaluated at C_0.

Method	M500	A24	A25	Olym.	HE+	MB+	LCB	GPQA	Mean
Single	94.6	65.8	59.1	56.7	86.2	75.8	49.4	56.7	68.04
Long-CoT	96.0	68.2	62.0	59.1	87.2	77.2	51.8	57.6	69.89
BoN	96.4	70.0	63.8	60.0	87.8	78.3	53.2	58.3	70.98
Root-adaptive	96.8	69.6	64.2	60.4	87.9	78.5	53.4	58.6	71.18
Question-only	96.5	69.1	63.0	59.4	87.6	78.0	52.6	58.1	70.54
Confidence-only	96.7	69.7	63.7	60.2	87.9	78.5	53.3	58.5	71.06
PAV-reopen	97.0	70.3	64.5	60.7	88.0	78.7	53.8	58.6	71.45
VGB†	96.9	70.5	64.8	60.5	87.8	78.6	54.0	58.4	71.44
Subpool†	97.1	70.6	65.0	60.6	88.2	78.8	54.2	58.9	71.68
PB-SMC†	97.3	70.9	65.5	60.8	88.3	78.8	54.4	59.2	71.90
Greedy-net-value	97.8	72.2	66.3	61.8	88.5	79.4	55.3	59.0	72.54
RAVEC	98.8	73.0	67.3	62.5	88.7	80.0	56.4	58.7	73.18

M.2 Cross-model aggregates and task differences

The full matrix contains 14 models $\times$ 12 tasks $\times$ 4 methods = 672 scores, and the eight additional Qwen3-8B controls contribute 96 more, for 768 in total. RAVEC improves 74 of the 112 core model–task comparisons against PB-SMC, with positive aggregate estimates for every evaluated model. The remaining task-level variation identifies where the recoverability signal is most valuable, while the cross-model intervals establish the breadth of the aggregate effect. Statistical uncertainty remains question-clustered, as specified in Appendix D.3.

Table 33: Eight-task core macro accuracy for each of the 14 generators, supporting the cross-model comparison of Contribution 3. BoN, PB-SMC-adapted, and RAVEC are evaluated at $2.4C_0$ and Single at C_0. Δ is RAVEC minus PB-SMC, recomputed from the complete task matrix. Cost normalization is model-specific.

Model	Single	BoN	PB-SMC	RAVEC	Δ
Qwen3-4B	63.13	67.10	68.24	70.13	+1.89
Qwen3-8B	68.04	70.98	71.90	73.18	+1.28
Qwen3-14B	72.26	74.84	76.74	77.70	+0.96
Qwen3-32B	75.01	77.55	78.43	79.34	+0.91
Qwen3.5-9B	77.89	80.00	81.09	81.95	+0.86
Qwen3.5-27B	82.08	83.79	84.58	86.34	+1.76
Qwen3.6-27B	84.08	85.73	86.68	87.85	+1.18
Qwen3.8-27B	86.50	87.83	88.39	89.78	+1.39
R1-Distill-Qwen-7B	55.40	59.44	61.14	62.49	+1.35
R1-Distill-Qwen-14B	63.75	67.18	68.55	70.34	+1.79
R1-Distill-Qwen-32B	68.31	71.30	72.59	73.61	+1.03
Gemma-3-12B-IT	45.50	49.63	51.76	53.23	+1.46
Gemma-3-27B-IT	52.16	55.69	57.16	58.54	+1.38
Llama-3.1-8B-Instruct	31.41	35.63	37.43	39.01	+1.59

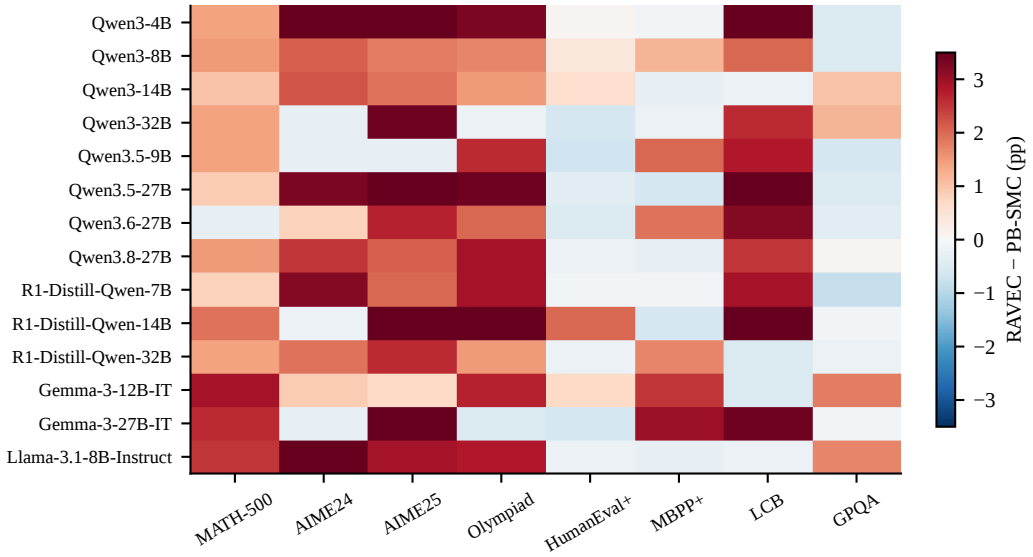


Figure 12: Task-level accuracy change of RAVEC over PB-SMC-adapted across models and tasks, in percentage points. Aggregate gains remain positive across models, while task-level variation identifies where recoverability-aware allocation contributes most.

M.3 Complete task scores by model

All tables in this subsection use the same protocol: Single is evaluated at C_0, and BoN, PB-SMC, and RAVEC at $2.4C_0$ expected operations. PB-SMC denotes the harmonized adaptation described in Appendix E.

The matrix makes the task-level pattern behind the aggregate comparison explicit. RAVEC's largest improvements over PB-SMC appear on tasks where the initial trace contains a recoverable commitment or an omitted verification step, while gains are smaller on tasks whose errors require information absent from the retained trajectory. This distinction is visible across both mathematical and coding benchmarks: the method raises selected accuracy on the principal MATH, AIME, and LiveCodeBench columns while preserving the common operation point. The same selector and scoring convention are used for every row, so differences reflect the controller's choice of prefix and suffix budget rather than a change in answer ranking.

Model scale changes the baseline difficulty and the available repair headroom, but it does not determine the direction of the intervention effect. The model rows therefore complement the cross-model interval table: they show absolute performance, while the final column isolates the gain over the matched persistent-pool comparator. Coding and general-knowledge columns also expose cases in which candidate coverage and selected accuracy diverge, which is the setting targeted by the repair–damage objective. The complete rows preserve these task-specific details instead of collapsing them into a single macro, allowing the aggregate claims to be checked directly from the displayed values.

The tables use the same fixed task slices and parsing rules as the primary evaluation. Single remains the C_0 reference, and all adaptive methods share the $2.4C_0$ expected-operation design; consequently, the comparison is within each model–task pair and does not interpret absolute scores across different checkpoints as a scaling law. Together, Tables 34–40 provide the complete numerical basis for the model aggregates and for the task-level variation shown in Figure 12.

Table 34: Complete 12-task selected accuracy (%) for Qwen3-4B and Qwen3-8B.

Method	GSM	M500	AMC	A24	A25	Min.	Oly.	HE+	MB+	LCB	GPQA	M-Pro
Qwen3-4B												
Single	94.8	93.1	70.8	61.4	54.8	42.7	49.5	80.6	68.9	45.6	51.1	60.3
BoN	95.6	95.8	76.2	67.1	60.9	46.4	54.1	83.3	72.3	50.0	53.3	61.3
PB-SMC	95.7	97.2	77.0	68.1	62.5	47.0	55.0	83.1	75.0	51.2	53.8	62.0
RAVEC	95.8	98.6	77.4	71.9	66.0	47.8	58.3	83.2	74.9	54.8	53.3	61.5
Qwen3-8B												
Single	96.1	94.6	76.4	65.8	59.1	48.9	56.7	86.2	75.8	49.4	56.7	66.4
BoN	96.5	96.4	80.1	70.0	63.8	52.0	60.0	87.8	78.3	53.2	58.3	67.6
PB-SMC	96.4	97.3	81.2	70.9	65.5	52.5	60.8	88.3	78.8	54.4	59.2	67.8
RAVEC	96.6	98.8	80.9	73.0	67.3	53.2	62.5	88.7	80.0	56.4	58.7	67.4

Table 35: Complete 12-task selected accuracy (%) for Qwen3-14B and Qwen3-32B.

Method	GSM	M500	AMC	A24	A25	Min.	Oly.	HE+	MB+	LCB	GPQA	M-Pro
Qwen3-14B												
Single	96.4	94.4	80.7	70.6	63.2	53.1	62.4	89.8	80.7	55.8	61.2	70.8
BoN	96.8	96.3	83.9	74.4	67.6	56.0	65.6	91.8	83.0	59.0	61.0	71.3
PB-SMC	96.6	98.5	84.8	75.9	70.4	56.4	67.7	91.6	85.4	63.0	61.4	72.3
RAVEC	96.9	99.5	85.3	78.1	72.3	57.5	69.2	92.2	85.1	62.8	62.4	72.8
Qwen3-32B												
Single	97.0	97.0	84.3	73.1	65.6	56.9	67.1	91.5	83.8	58.6	63.4	74.2
BoN	96.9	96.8	87.7	77.2	69.9	59.3	70.0	93.3	85.8	62.1	65.3	75.2
PB-SMC	96.9	97.8	87.9	77.6	71.3	59.8	72.7	93.4	86.5	62.9	65.2	75.4
RAVEC	97.1	99.2	88.3	77.3	74.7	60.3	72.5	92.8	86.3	65.5	66.4	76.0

Table 36: Complete 12-task selected accuracy (%) for Qwen3.5-9B and Qwen3.5-27B.

Method	GSM	M500	AMC	A24	A25	Min.	Oly.	HE+	MB+	LCB	GPQA	M-Pro
Qwen3.5-9B												
Single	96.8	95.8	85.6	74.9	68.7	59.8	69.4	94.2	85.0	60.4	74.7	80.4
BoN	97.0	97.1	88.2	78.1	72.4	62.4	71.9	94.3	86.8	63.4	76.0	81.1
PB-SMC	97.2	98.0	88.8	81.4	74.3	63.1	72.4	94.1	87.6	64.7	76.2	81.6
RAVEC	97.3	99.4	89.1	81.1	74.0	64.0	75.0	93.4	89.6	67.5	75.6	81.8
Qwen3.5-27B												
Single	97.4	97.3	88.3	78.2	72.6	64.1	73.8	95.6	88.1	70.8	80.2	84.0
BoN	97.3	97.2	91.0	81.3	76.4	66.2	76.4	95.4	90.0	73.5	80.1	84.4
PB-SMC	97.2	98.6	91.1	82.2	76.6	66.9	76.2	96.9	90.6	74.6	80.9	85.0
RAVEC	97.5	99.5	92.0	85.5	80.7	67.9	79.6	96.5	90.0	78.5	80.4	85.5

Table 37: Complete 12-task selected accuracy (%) for Qwen3.6-27B and Qwen3.8-27B.

Method	GSM	M500	AMC	A24	A25	Min.	Oly.	HE+	MB+	LCB	GPQA	M-Pro
Qwen3.6-27B												
Single	97.2	97.5	90.1	81.0	75.4	66.7	76.2	96.1	89.7	74.9	81.8	84.5
BoN	97.2	97.4	92.2	83.6	78.6	68.1	78.2	96.0	91.1	77.7	83.2	85.1
PB-SMC	97.2	99.5	92.5	86.4	79.1	69.3	78.6	96.8	91.7	78.4	82.9	85.4
RAVEC	97.1	99.2	92.0	87.2	81.8	69.2	80.6	96.3	93.6	81.6	82.5	85.7
Qwen3.8-27B												
Single	97.7	97.7	91.8	83.6	78.9	69.8	79.3	96.8	92.2	79.7	83.8	85.3
BoN	97.6	98.0	93.9	86.2	81.7	71.8	81.0	96.7	92.1	82.1	84.8	85.9
PB-SMC	97.8	97.8	94.2	86.8	82.1	72.1	81.3	97.2	94.0	82.5	85.4	86.2
RAVEC	97.7	99.3	94.8	89.3	84.2	73.2	84.2	97.0	93.7	85.0	85.5	85.4

Table 38: Complete 12-task selected accuracy (%) for R1-Distill-Qwen-7B and R1-Distill-Qwen-14B.

Method	GSM	M500	AMC	A24	A25	Min.	Oly.	HE+	MB+	LCB	GPQA	M-Pro
R1-Distill-Qwen-7B												
Single	94.1	89.6	65.4	49.3	31.7	38.8	44.2	80.9	67.3	34.2	46.0	57.2
BoN	94.8	92.4	70.6	55.2	38.1	42.5	48.6	83.3	70.8	38.7	48.4	58.4
PB-SMC	94.4	96.1	71.3	56.6	39.2	43.1	49.8	84.8	73.4	40.3	48.9	59.2
RAVEC	94.9	96.9	72.1	59.8	41.2	43.8	52.7	84.7	73.3	43.2	48.1	59.5
R1-Distill-Qwen-14B												
Single	95.1	91.9	73.6	61.4	40.8	46.3	52.1	86.5	75.0	47.7	54.6	63.4
BoN	95.6	94.2	78.0	66.3	46.4	49.5	56.0	88.4	77.9	51.6	56.6	64.4
PB-SMC	95.7	95.5	78.6	68.7	47.9	49.9	57.1	88.6	79.3	53.0	58.3	65.1
RAVEC	95.4	97.4	79.8	68.5	52.1	51.3	60.6	90.6	78.7	56.6	58.2	64.7

Table 39: Complete 12-task selected accuracy (%) for R1-Distill-Qwen-32B and Gemma-3-12B-IT.

Method	GSM	M500	AMC	A24	A25	Min.	Oly.	HE+	MB+	LCB	GPQA	M-Pro
R1-Distill-Qwen-32B												
Single	96.0	92.7	79.2	65.2	47.2	51.5	59.4	89.7	81.1	52.5	58.7	69.1
BoN	96.3	94.6	82.7	69.3	52.0	54.2	62.6	91.8	83.4	55.8	60.9	69.9
PB-SMC	96.1	95.6	83.4	70.6	52.8	54.7	65.2	92.0	85.5	58.3	60.7	70.5
RAVEC	96.5	97.0	83.5	72.5	55.4	55.2	66.7	91.8	87.2	57.8	60.5	70.9
Gemma-3-12B-IT												
Single	93.7	82.3	55.9	18.6	15.4	31.7	34.6	79.5	66.1	26.9	40.6	60.9
BoN	94.5	85.6	61.5	24.0	21.3	35.7	39.3	82.6	69.8	31.6	42.8	62.2
PB-SMC	94.7	87.2	62.3	27.2	25.4	36.4	40.5	82.4	70.8	36.2	44.4	63.1
RAVEC	95.0	90.1	63.7	28.1	26.1	38.2	43.2	83.1	73.3	35.7	46.2	63.4

Table 40: Complete 12-task selected accuracy (%) for Gemma-3-27B-IT and Llama-3.1-8B-Instruct.

Method	GSM	M500	AMC	A24	A25	Min.	Oly.	HE+	MB+	LCB	GPQA	M-Pro
Gemma-3-27B-IT												
Single	95.4	87.1	67.8	27.4	21.5	38.9	43.7	85.1	74.1	35.2	43.2	67.1
BoN	95.9	89.6	72.5	32.2	26.7	42.3	47.7	87.1	78.1	39.1	45.0	68.1
PB-SMC	96.0	90.8	73.2	36.1	27.6	42.8	50.6	88.2	77.9	40.7	45.4	68.7
RAVEC	96.0	93.4	74.0	35.8	31.1	43.8	50.1	87.6	80.9	44.1	45.3	68.9
Llama-3.1-8B-Instruct												
Single	84.0	50.2	32.8	6.4	3.1	15.2	16.8	69.8	56.8	16.7	31.5	47.6
BoN	86.4	55.2	39.0	11.0	7.1	19.1	21.0	74.6	61.5	21.1	33.5	49.7
PB-SMC	86.2	56.9	39.9	12.5	8.1	19.7	22.3	76.8	62.7	26.1	34.0	50.5
RAVEC	87.0	59.4	41.1	16.0	11.0	21.7	25.1	76.6	62.4	25.9	35.7	50.0