# KEEP A BRANCH: COMPLETION-AWARE WIDTH CONTROL FOR VERIFIER-GUIDED REASONING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

A branch kept alive by test-time search can survive a layer without advancing, advance without finishing, and finish without winning the final vote. We study how verifier-guided reasoning should choose its width, the number of branches retained per layer, under a fixed token budget, and find that each link of this chain fails separately. On 195 MATH problems, widening a frozen 7B beam from width 1 to 5 under a fixed ceiling halves accuracy, from 82% to 42%. The wide beam leaves 112 runs with no completed answer: at width 5, only 7,116 of the 65,536 budget tokens are newly generated text. Exact-distance labels on 1.22 million planning candidates isolate the earlier links, separating retention of an improving branch from sustained progress toward the goal. The diagnosis yields two requirements for useful exploration: an extra branch must justify its cost, and enough budget must remain to finish one retained path. SCA+ turns them into a per-layer width rule built from an estimated marginal gain and an estimated completion reserve. Under total-token accounting, with verifier calls charged like generation, SCA+ exceeds its strongest budget-aware baseline on MATH by 4.7 points with 16% fewer tokens while both discover correct answers equally often. The margin comes from completion quality: narrower width concentrates tokens per candidate, raises verifier scores at completion and shifts a fixed final vote.

## 1 INTRODUCTION

Test-time search keeps several partial solutions alive and uses a learned verifier to decide which survive. Extra width can preserve a useful alternative, but it also spends tokens that the surviving paths need in order to finish. This creates an allocation problem: *when is another retained branch worth its immediate cost and the future work needed to reach an answer?* Verifier-guided search can fall behind repeated sampling as compute grows, in part because of misranking (Yu et al., 2025). Budget-aware search responds by tapering exploration and favoring completion as the budget runs out (Miyamoto et al., 2026). Yet the fraction of budget remaining does not say how much unfinished work remains: the same token balance can be ample for one set of live branches and insufficient for another.

Two contrasts expose the decisions hidden by an endpoint accuracy curve. In planning, where each width runs its own recipe with no shared token ceiling, width can be almost the whole story or none of it. With Qwen2.5-7B-Instruct proposing and scoring steps, raising retained width from 1 to 5 lifts Miconic success from 0.08 to 0.99; Gripper stays near zero even though it retains a locally improving candidate in 90% of layers (Figure 1A). A beam can keep good branches at every layer and still go nowhere. In mathematics, where all methods share one total-token ceiling, the verifier is Qwen2.5-Math-PRM-7B, a process reward model (PRM) that scores each intermediate step. The same proposer solves 159 of 195 MATH problems at width 1 and 82 at width 5. Of the width-5 runs, 112 end with no completed answer at all (Figure 1B). Under a fixed budget the wide beam simply fails to answer, and no ranking intervention can recover an answer that the search never finishes.

These observations motivate two quantities for a width decision. The first is the marginal success gain of retaining another candidate, which depends on the branches currently alive. The second is the cost of completing a retained path, which cannot be read off a verifier score or a budget fraction.



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.

SCA+ estimates both from continuations logged during development and spends on widening only what the completion budget can spare. Because the final selector acts on a completed-answer pool, we also track where a correct answer is discovered but not returned. Width therefore fails in two layers, and both recur throughout the paper: against a fixed wide beam, the dominant loss is answers that never finish; against methods that already protect completion, the remaining contest is over the scores with which finished answers enter the pool.

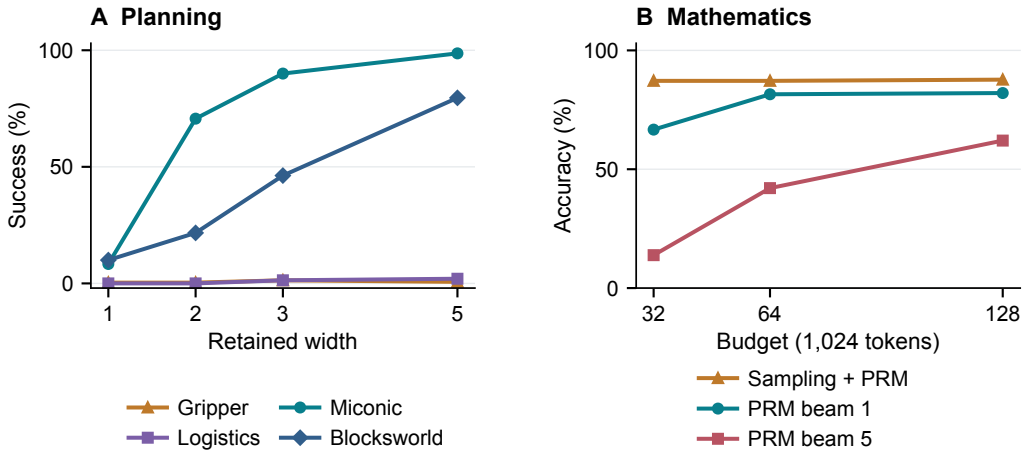


Figure 1: **The budget regime decides what width is worth.** **A**: planning success for the same frozen proposer–verifier recipe at widths 1, 2, 3 and 5, where each width runs to its depth cap with *no shared token ceiling*; width separates domains it rescues from domains it cannot. **B**: accuracy on 195 MATH problems where all methods share one *fixed total-token ceiling*; here the wide beam loses because most of its runs never complete an answer. Panel B is the regime the rest of the paper studies. Retention and progress are examined in Figure 2.

**Contributions.** We connect a diagnosis of search failures to a concrete allocation rule. Exact-distance planning labels separate keeping a good branch from advancing with it, and math outcomes separate finishing, finding and returning a correct answer (Sections 2–3). From this diagnosis, SCA+ combines a state-dependent marginal gain with an estimated completion reserve, using only signals available during search (Section 4). Section 5 tests the two components, transfers the rule to mathematics, and measures cost once accuracy saturates; a trajectory-level attribution follows individual width decisions through the completed pool to the final vote (Appendix H).

## 2 Width, Retention and Frontier Progress

Planning supplies a diagnostic unavailable in most reasoning tasks: the exact distance from any state to the goal. A frozen proposer generates candidates; a frozen verifier ranks them; the policy retains the top $k$ and a terminal checker tests success. We analyze 1.22 million labeled candidates from runs executed at widths $\{1, 2, 3, 5\}$ on four classical planning domains (Valmeekam et al., 2023a): Gripper (robots carrying balls), Logistics (delivering packages), Miconic (an elevator serving passengers) and PlanBench Blocksworld (stacking blocks), with 100 tasks per generated domain, 80 Blocksworld tasks and three seeds (Appendix A). The distance oracle is a measuring instrument: it labels trajectories after the runs, and the search never sees it.

**Width buys layer survival.** A candidate improves locally when its exact distance is smaller than its parent's. For any single parent, the chance that its improving candidate survives selection falls as the beam widens, from 0.44 at width 1 to 0.37 at width 5 on Gripper, because more candidates compete for each slot. For the layer as a whole, the chance that at least one improving candidate survives somewhere rises, from 0.44 to 0.90 on the same domain (Figure 2A). Keeping more branches raises

the chance that the layer still contains an improving branch; it says nothing yet about whether that branch will reach the goal.

**Retention and progress respond differently to width.** Let $D_t$ be the smallest exact distance among the branches alive at layer $t$ (the frontier); $u_t = D_t - D_{t+1}$ is signed progress. At width 5, Miconic advances 0.87 steps per layer and its verifier ranks improving candidates above non-improving ones in 69% of pairs. Gripper advances only 0.20 steps per layer with a 50% pairwise win rate, on tasks requiring 8–18 steps. Blocksworld ranks below chance yet progresses at width 5, where redundancy compensates for some ranking errors (Figure 2B; ranking curves in Appendix A, Figure 7). Pairwise win rates barely change with width. Survival, progress and ranking quality therefore carry different information.

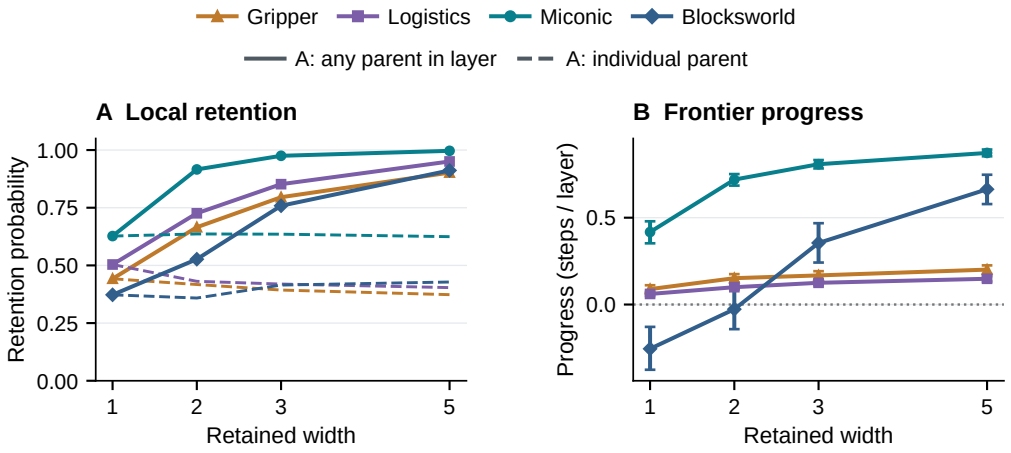


Figure 2: **Keeping a good branch and advancing with it are distinct measurements. A**: probability that an improving candidate survives anywhere in a layer (solid) or from an individual parent (dashed). **B**: signed frontier progress; zero separates progress from retreat, and error bars are task-bootstrap 95% intervals. Ranking quality and replay bias are shown in Figure 7; all estimands are defined in Appendix A.

**Ranking intervention identifies a limit of width control.** A control campaign that swaps in exact-distance ranking (Appendix B) bounds what width can fix: oracle ranking lifts Gripper and Logistics from zero to 0.68–1.00 at every width, whereas its gain on Miconic shrinks from $+0.89$ at width 1 to $+0.02$ at width 5. Width can therefore compensate for some ranking errors, but it cannot repair near-chance ranking, and ranking quality is a property of the verifier–task pair. Our planning evaluation (E4, Section 5) therefore uses task distributions where fixed-width-1 success is 66–75%, where allocation is the live question.

**Implication for width control.** Width can insure against some ranking errors, but neither retention nor average ranking quality measures what one more branch is worth at a particular state. A shortcut fails too: replaying a wider selection on traces recorded at a narrower width overstates what wide search actually keeps, by up to 0.34 (Appendix A, Figure 7B), so the gain must be measured from runs executed at the width in question. The controller in Section 4 does exactly that. A second constraint is still missing: these planning diagnostics measure progress in state space, and they say nothing about the token cost of turning that progress into a completed answer.

## 3 Completion and Selection under a Token Budget

A useful branch must finish before it can affect the returned answer. Mathematics makes this requirement observable without an exact-distance oracle: completed answers can be graded offline, and every token can be assigned to the call that spent it. Let $T$ denote completion of at least one

usable answer, $C$ the presence of a correct answer in that pool, and $S$ return of a correct answer. Since $S \subseteq C \subseteq T$,

$$\Pr(S) = \Pr(T)\ \Pr(C \mid T)\ \Pr(S \mid C). \tag{1}$$

The identity is exact and separates completion, discovery and selection. It yields four disjoint outcomes: correct returned, correct in the pool but lost, only wrong answers, and no completed answer. Figure 3 applies this decomposition to the 195-problem study (protocol in Appendix C).

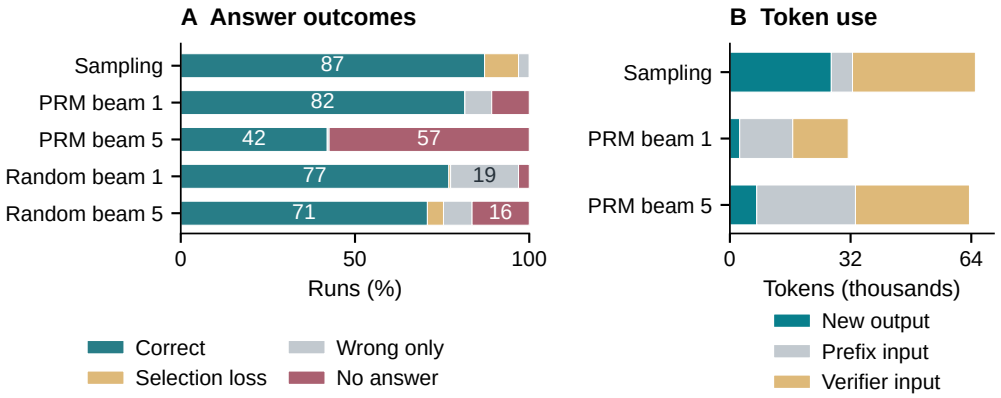


Figure 3: **The wide beam loses before it has an answer to select. A**: outcomes for 195 problems at 65,536 tokens; labels are rounded percentages. Width-5 PRM search finishes only 83 problems, 82 correctly. **B**: mean charged-token composition at the same ceiling: at width 5, only 7,116 of the 65,536 tokens are newly generated content; prefix and verifier input absorb the rest. Pool coverage across budgets is shown separately in Appendix C, Figure 10.

**Wide search spends its budget before finishing.** At width 5, 112 problems have no completed answer; of the 83 that complete, 82 are correct. Width 1 completes 174 under the same ceiling; the wide beam's failure lies in how it spends. Re-feeding growing prefixes and scoring candidates absorb most of the ceiling (Figure 3B), and the wide search advances only 4.8 of its at most 12 layers, each a round of proposing one step per retained branch, against 7.9 at width 1. Even doubling the ceiling raises width-5 accuracy only to 62.1%, still below sampling at 87.7%.

**Sampling finds answers that selection can still lose.** Sampling has a correct answer in 189 pools but returns it in only 170; majority or PRM-weighted voting on the same pools returns 171. The gap is a second kind of loss: the answer was found and then lost at selection (Figure 10).

**How width reaches the final vote.** Width affects the answer through a specific causal pathway: retained candidates determine subsequent expansions and token spending; trajectories that finish and are scored before stopping enter the pool; the fixed selector acts on that pool. Under PRM-weighted voting, answers are grouped by mathematical equivalence, each completed trajectory adds its exponentiated score to its group, and the group with the largest total wins. We call the weight lead of the best correct group over the best incorrect group the *vote margin* $m(\mathcal{P})$ (formal definition in Appendix H). With the ordered, scored pool fixed, the answer is fixed. New completions can strengthen either side of the vote, so the effect of a particular width decision depends on the trajectories it changes.

**Implication for the controller.** A width controller must therefore value a branch by more than its chance of surviving the next layer: it needs an estimate of the final-success gain and an estimate of the work still remaining after expansion. Reading that remaining work from depth, context length and the observed per-step cost separates states that share a token balance but differ in how much is left to do. The two failure layers of the introduction now have addresses in Equation 1: answers that never finish sit in $\Pr(T)$, and the contest among finished answers sits in $\Pr(S \mid C)$, through the vote margin. The next section builds the rule.

## 4 SCA+: GAIN AND COMPLETION-AWARE WIDTH CONTROL

The diagnosis asks for two numbers at every layer: what another branch would add, and what finishing still costs. SCA+ estimates both from continuations logged during development and combines them in a single width rule.

**State and actions.** At a live layer, the controller sees $n \leq 5$ scored unfinished candidates, the remaining tokens $b$, the current depth $d$ and depth cap $D$, the observed per-parent expansion cost $\hat{c}$ with its trend, and the context length. Action $k \in \{1, \ldots, n\}$ retains the top $k$ candidates by verifier score. The controller reserves future work for one path only; the other retained branches are free to stop unfinished. Goal distances, reference answers and future outcomes are unavailable to it. Figure 4 shows the decision loop.

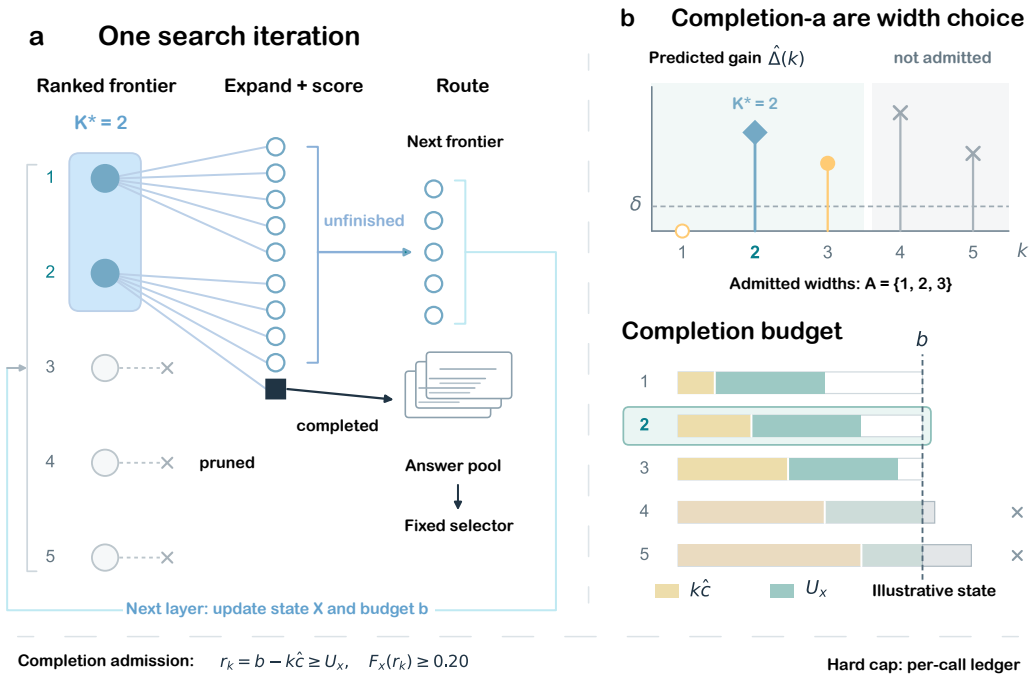


Figure 4: **Two estimates determine one width decision.** The gain model values exploration; the completion reserve restricts which widths are affordable. Only unfinished candidates return to the next layer; completed answers enter the shared final selector. Algorithm 1 gives the full rule, including the fallback for states the models have not seen; a per-call ledger, which checks before every model call that its worst-case cost still fits, independently enforces the hard ceiling.

**Question one: is the extra branch useful?** Let $Q_0(x, b, k)$ be the final-success probability after retaining $k$ candidates now and continuing narrowly, at width 1, ever after. The widening gain is $\Delta_0(k) = Q_0(x, b, k) - Q_0(x, b, 1)$. To label it, development runs continue each source state twice from the same budget and seed: once keeping $k$ branches, once keeping one. The paired difference in success is the label $Y(k) - Y(1)$. A ridge regression (Hoerl & Kennard, 1970) predicts this gain from the score profile, candidate novelty (how much the retained candidates differ from one another), depth, remaining budget, observed cost and context length (Appendix D), and is built so that $\hat{\Delta}(1) = 0$. The estimate values widening the current layer under a narrow future; at deployment the controller re-decides at every layer.

**Question two: can a retained path still finish?** The width-1 continuations from those same pairs also record how many further tokens the first usable answer took; a run that fails is recorded as a

failure, and one the budget cuts off first is censored. From the most similar development states, a standard estimator for censored records (Aalen & Johansen, 1978) turns them into a completion curve $F(t)$: the probability that a narrow continuation has finished within $t$ further tokens. Two numbers then guard the decision. The reserve $U$ is the cost by which 85% of the completions that do occur have finished, and answers how much budget a finishing path needs. The completion probability $F$ itself keeps a state that rarely finishes from looking affordable just because its few completions were cheap. Retrieval details are in Appendix D.

**Choose useful exploration within the completion budget.** Paying $k\hat{c}$ for the next expansion leaves $r_k = b - k\hat{c}$. A width is *admitted* when what remains still covers the reserve, and *useful* when its predicted gain clears a small threshold:

$$\mathcal{A}(x, b) = \{k \in \{1, \ldots, n\} : r_k \geq U_x,\ F_x(r_k) \geq 0.20\}, \quad \mathcal{W} = \{k \in \mathcal{A} : k > 1,\ \hat{\Delta}(k) > 0.02\}.$$

The controller takes the admitted width with the largest predicted gain, and keeps one branch when $\mathcal{W}$ is empty. States too far from the development data fall back to a fixed depth-based reserve and width at most 2. The rule is a forecast; a per-call ledger independently enforces the hard ceiling. Completed answers leave the search and enter the shared selector: in math, the equivalence-grouped PRM-weighted vote of Section 3; in planning, a goal check. Algorithm 1 states the boundary cases; Appendix E bounds the decision loss.

**Development, then freezing.** Sixty-four planning source problems fit both models, 32 validate them, and 24 select between two threshold settings; the winner (gain threshold 0.02, quantile 0.85) was frozen before any evaluation, including mathematics and the 14B model. Development consumed 22.36 million tokens. A hand-set variant with no learned models is kept as the ablation SCA: it widens until the retained candidates cover most of the score mass, capped so the budget still carries one path to the depth limit at the observed cost. SCA+ is the same allocation idea with the two learned estimates in place of the hand-set rules. The development tasks are disjoint from every evaluation panel.

## 5 Evaluation

The evaluation asks three linked questions: do both estimates matter (E4), where exactly do math outcomes differ (E5), and how much work does accuracy cost once it saturates (E6)?

**Protocol.** Three panels share the frozen controller; their labels continue the numbering of the development-phase runs E1–E3, which ship with the released code. **E4**: 360 planning tasks (120 each of Gripper, Logistics and Miconic) $\times$ 3 seeds with Qwen2.5-7B-Instruct as proposer and verifier. **E5**: 480 MATH problems (Hendrycks et al., 2021) $\times$ 3 seeds with Qwen2.5-7B-Instruct and Qwen2.5-Math-PRM-7B (Zhang et al., 2025), disjoint from the 195 problems of Section 3. **E6**: the E4 tasks and seeds with Qwen2.5-14B-Instruct in both roles.

All arms share the generator and budget ledger, and verifier-based arms also share the verifier and final selector; the exceptions are the vote without a PRM, which has no verifier to share, and BG early stopping, which stops by its own rule. The budget caps every token a model call touches: generation input, generation output and verifier input. We call these *charged tokens*. Primary budgets are 32,768 for E4 and E5 and 65,536 for E6. The unit of analysis is the task, with seeds averaged within task; intervals are 95% task-bootstrap and the declared primary contrasts are Holm-adjusted (Appendix G).

**Comparisons.** We compare: fixed widths; REBASE, which spreads an expansion quota over branches in proportion to their scores (Wu et al., 2025), and REBASE+F, the same method capped by our completion reserve; best-of-$n$ sampling (BoN), which generates complete solutions independently and lets the PRM-weighted vote pick; a vote with no PRM at all; and budget-guided Monte Carlo tree search (BG-MCTS), with and without its early stop (Miyamoto et al., 2026). Four ablations each disable one part of SCA+: *no gain model* always takes the widest affordable width, *no completion filter* drops the reserve, *pooled statistics* replaces the live scores with development averages, and *depth schedule* replaces the learned reserve with a fixed depth rule. Appendix F specifies every adaptation; Appendix I reports every arm.

### 5.1 E4: SEPARATING THE GAIN MODEL FROM THE COMPLETION RESERVE

Figure 5 and the full table in Appendix I compare SCA+ with fixed widths, REBASE and the rule without learned models at the primary budget. Giving REBASE the completion reserve already helps, by 2.0 pp and 7.7% fewer tokens. On top of that, SCA+ adds 9.1 pp and saves a further 16.8% of tokens. The margin holds in every domain (Appendix I) and at every budget from 16,384 to 65,536 (Figure 5A).

The ablations separate the components' intended roles (Figure 5B). Removing the completion filter reduces accuracy by 18.9 pp, while the share with no usable answer rises from 2.1% to 9.2%. Taking the largest affordable width without a gain model reduces accuracy by 10.4 pp; pooled scores and a fixed depth schedule lose 8.9 pp and 11.9 pp.

**How the components change width.** The layer decisions make the distinction concrete: SCA+ averages about 1.9 branches in the first three layers and 1.15 in layers 5–9, with widths 4 and 5 never chosen in any panel, whereas removing the completion filter keeps mean width near 3.4. Without the gain model, affordable breadth is used more often (Appendix H). The gain model governs whether to spend; the reserve limits how long breadth can continue. Non-completion explains only part of the accuracy difference, so E5 follows the chain to its second layer: how completed trajectories enter the pool.

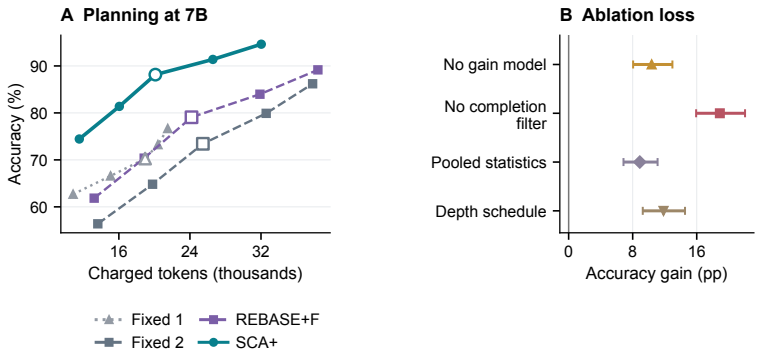


Figure 5: **E4: the two components answer different questions. A**: accuracy versus mean tokens spent across five ceilings; hollow points mark 32,768 tokens. **B**: paired accuracy loss from replacing one component of SCA+ at that ceiling, with 95% task-bootstrap intervals. 360 tasks × 3 seeds.

### 5.2 E5: FROM COMPLETING PATHS TO SELECTING AN ANSWER

E5 moves the rule, unchanged, from planning to mathematics. Table 1 reports the primary 32,768-token comparison; Figure 6 pairs the budget frontier with the outcome decomposition of Equation 1.

Table 1: E5, MATH, 7B proposer with Qwen2.5-Math-PRM-7B, 480 problems × 3 seeds, 32,768-token budget. Differences are SCA+ minus the row with 95% task-bootstrap intervals; the BG comparison, reported by its package as run-level pairs, is tested as its own two-contrast family (Appendix G).

| Method | Accuracy | Tokens | SCA+ minus method |
|---|---|---|---|
| BoN + PRM vote | 83.33% | 27,755 | $+\mathbf{9.03}$ $[\mathbf{7.01}, \mathbf{11.11}]$ |
| REBASE+F | 84.31% | 27,278 | $+8.06$ $[6.32, 10.00]$ |
| BG-MCTS | 87.64% | 30,578 | $+\mathbf{4.72}$ $[\mathbf{2.64}, \mathbf{6.81}]$ |
| **SCA+** | **92.36%** | **25,666** | |

**Against sampling, the gain comes from the pool.** Under the total-token accounting of the protocol, SCA+ exceeds PRM-weighted sampling by 9.0 pp with 7.5% fewer tokens, and it reaches at

32,768 tokens the accuracy that sampling reaches at 65,536 ($+0.14$ pp, $[-1.32, 1.67]$) with 53.6% fewer tokens. The outcome decomposition locates the margin (Figure 6B). Fixed width 5 leaves 373 of 1,440 runs without a usable answer and SCA+ leaves 2, so non-completion is nearly absent for the controller. Against BoN, the runs whose pool holds no correct answer fall from 78 to 36, and the runs that found a correct answer and lost it at selection fall from 162 to 72. SCA+ leads BoN at all five difficulty levels (Appendix I).

**Against budget-guided tree search, the margin lies in completion quality.** SCA+ leads BG-MCTS by 4.72 pp [2.64, 6.81] with 16.1% fewer charged tokens at 32,768 and by 2.15 pp [0.90, 3.47] with 32.5% fewer at 65,536. The two methods contain a correct answer in almost equally many pools: 1,402 for SCA+ and 1,403 for BG-MCTS. Yet BG-MCTS loses 141 at selection where SCA+ loses 72. Thus 69 fewer selection losses, offset by one fewer correct-containing pool, account for 68 more correct returns. The selector is shared, so the difference lies in what it receives: the completed answers and their relative vote weights. The token margins here are in charged tokens; Appendix J restates them in output tokens and modeled time, where they shrink or partly reverse.

**Completion quality tips the vote.** To isolate the mechanism, we compare SCA+ with its no-gain-model ablation, which keeps the completion reserve but always takes the widest affordable width, so completion is protected in both arms, and only the narrowing differs. SCA+ selects correctly in 199 runs where this ablation does not (28 run the other way). In 193 of the 199, both arms completed the same number of correct trajectories. The narrower arm concentrated tokens per candidate, raising the pool's highest correct-class score from 0.755 to 0.926, and its smaller pools put less weight on wrong classes, so the margin moves from both sides at once; the 28 reverse cases mirror the pattern (Appendix H). Paired early-stop examples show the last link directly: accumulating further vote weight can either repair or reverse a selected answer (Figure 11).

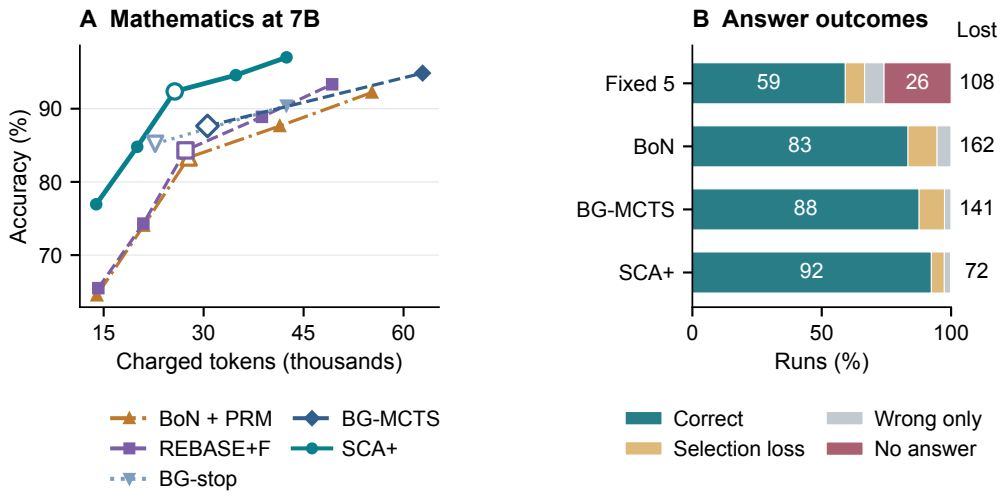


Figure 6: **E5: compare the frontier, then locate the loss.** **A**: accuracy versus mean charged tokens; hollow points mark 32,768 tokens. BG methods are available at two ceilings. **B**: outcome shares at 32,768 tokens, with rounded percentages; the right column gives exact selection-loss counts ("Lost"). The margin over BG-MCTS arises after a correct answer reaches the pool. The selector is shared, so the margin reflects completion quality: the vote weights with which finished answers arrive. 480 problems $\times$ 3 seeds; primary paired intervals are in Table 1; all arms are in Appendix I.

### 5.3 E6: COMPLETION-AWARE ALLOCATION AT SATURATED ACCURACY

With Qwen2.5-14B-Instruct in both roles and the controller unchanged, the strongest adaptive methods approach the accuracy ceiling at 65,536 tokens (Appendix I). SCA+ reaches 98.70% with 22,974

tokens and REBASE+F 98.24% with 31,603: statistically comparable accuracy at 27.3% $[26.6, 27.9]$ lower cost, and the paired difference of $+0.46\,\mathrm{pp}$ $[-0.56, 1.57]$ rules out any loss beyond the pre-declared $2\,\mathrm{pp}$ margin. Removing the completion filter lowers accuracy by $2.4\,\mathrm{pp}$ and more than doubles mean tokens, from 22,974 to 48,087 (Figure 14). This regime complements E5: once final accuracy leaves little headroom, the relevant question is how much work produces it.

## 6 Related work

**Allocation and ranking.** Sampling and voting (Wang et al., 2023; Brown et al., 2024), adaptive allocation (Snell et al., 2025), reward-weighted expansion (Wu et al., 2025), tree search (Yao et al., 2023; Xie et al., 2023; Hao et al., 2023; Feng et al., 2024) and long-trace reasoning (OpenAI, 2024; Muennighoff et al., 2025) allocate inference compute in different ways. Verifiers began at the outcome level (Cobbe et al., 2021; Uesato et al., 2022); PRMs move the evaluation to intermediate steps (Lightman et al., 2024; Wang et al., 2024; Zhang et al., 2025), and progress-sensitive verifiers ask what a step contributes (Setlur et al., 2025). Verifier misranking can make search scale worse than sampling (Yu et al., 2025), and self-verification has documented limits of its own (Huang et al., 2024; Stechly et al., 2025). Our diagnosis separates this ranking limitation from failures to finish and failures to select, using exact planning semantics as an offline measurement instrument, in the spirit of planning benchmarks (Valmeekam et al., 2023a;b; Kokel et al., 2025) and of work that assigns planning its proper role around LLMs (Kambhampati et al., 2024; Katz et al., 2024).

**Budget-aware search.** Budget-guided MCTS anneals exploration and late branching with the fraction of budget left and prefers completion (Miyamoto et al., 2026); VG-Search treats verification frequency as the control variable (Chen et al., 2025); adaptive frameworks use PRM scores to steer pruning within and selection across iterations (Bilal et al., 2026). SCA+ instead reserves the estimated cost of completing one retained path, learned from censored completion records, decides the beam width at every layer, and charges verifier and input tokens to the same budget as generation.

## 7 Limitations

Width control can use only the information in its verifier: the hard planning distributions show that changing width alone does not repair weak ranking. The allocation study covers one model family and one PRM, in settings where ranking is informative or accuracy saturated, and E4 and E6 share tasks. The trajectory-level attribution covers one comparison (the no-gain-model ablation at 32,768 tokens in E5); extending it to other components and budgets remains open. The gain model is trained against width-1 continuations while the deployed controller re-decides at every layer; this mismatch, and the calibration of the deployed forecasts, remain unmeasured.

The cost advantage depends on the currency. Charging verifier input gives a different frontier from output-only accounting, and modeled seconds depend on the service assumptions (Appendix J). Baselines are adaptations to the shared interface, and the depth cap is a proxy for unknown solution length. The 195-problem diagnosis uses a pinned equivalence grader; an author audit found three false negatives without changing the ordering.

## 8 Conclusion

A retained branch earns its cost only at the end of a chain: it must survive, advance, finish, and win the final vote, and each link fails separately. SCA+ turns that requirement into a decision based on marginal success gain and the estimated cost of completing one retained path. In the primary comparison, under the shared total-token budget, the controller and its strongest baseline discover correct answers equally often; the accuracy difference comes from completion quality, specifically the scores with which completed trajectories enter the pool. Together, the rule and diagnostics make the value of an extra branch a question about the answers it can deliver within budget. The finding that allocation quality can be the binding constraint suggests that test-time compute scaling has as much to gain from smarter spending as from larger budgets.

## REPRODUCIBILITY STATEMENT

Appendices A–J specify task and model settings, token accounting, controller features and fitting, completion estimation, the action rule, baseline adaptations, statistics and service assumptions. Quantitative figures are rendered from the accompanying summary tables. The two example traces preserve the plotted geometry of the original vector figures with redrawn axes and labels. Code, configurations, task lists and result tables will be released.

## AI USE STATEMENT

We used a large language model (LLM) to assist with grammar checking and improve the clarity of the manuscript. All technical content, experimental design, and analyses were developed by the authors. All LLM-generated suggestions were manually reviewed and revised as necessary to ensure consistency with the objectives of the paper and to preserve technical accuracy.

## REFERENCES

Odd O. Aalen and Søren Johansen. An empirical transition matrix for non-homogeneous Markov chains based on censored observations. *Scandinavian Journal of Statistics*, 5(3):141–150, 1978.

Ahsan Bilal, Ahmed Mohsin, Muhammad Umer, Ali Subhan, Hassan Rizwan, Ayesha Mohsin, and Dean Hougen. What if we allocate test-time compute adaptively? In *International Conference on Machine Learning*, 2026. arXiv:2602.01070.

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

Hao Mark Chen, Guanxi Lu, Yasuyuki Okoshi, Zhiwen Mo, Masato Motomura, and Hongxiang Fan. Rethinking optimal verification granularity for compute-efficient test-time scaling. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.11730.

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

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1994.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

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

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. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B. Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 22895–22907, 2024.

Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi. Thought of search: Planning with language models through the lens of efficiency. In *Advances in Neural Information Processing Systems*, 2024.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. ACPBench: Reasoning about action, change, and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

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. In *International Conference on Learning Representations*, 2024.

Sora Miyamoto, Daisuke Oba, and Naoaki Okazaki. Aligning tree-search policies with fixed token budgets in test-time scaling of LLMs. In *International Conference on Machine Learning*, 2026. arXiv:2602.09574.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, 2025.

OpenAI. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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. In *International Conference on Learning Representations*, 2025.

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

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. In *International Conference on Learning Representations*, 2025.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023a.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models: A critical investigation. In *Advances in Neural Information Processing Systems*, 2023b.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *International Conference on Learning Representations*, 2025.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Self-evaluation guided beam search for reasoning. In *Advances in Neural Information Processing Systems*, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023.

Fei Yu, Yingru Li, and Benyou Wang. Scaling flaws of verifier-guided search in mathematical reasoning. *arXiv preprint arXiv:2502.00271*, 2025. URL https://arxiv.org/abs/2502.00271.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. arXiv:2501.07301.

## APPENDIX READING GUIDE

The appendices answer three questions. *Where does search lose useful work?* A–C define the diagnostic measurements and component controls. *How is the width rule implemented?* D–F specify the models, the algorithm, the decision-loss bounds and the baselines. *How should its outcomes be interpreted?* G defines uncertainty, H follows layer choices through answer pools, I gives the full comparisons, and J examines the dependence on cost accounting. Each figure is introduced by the question it answers and followed by its implication.

## A PLANNING SYSTEM, LABELS AND ESTIMANDS

**Components.** The learned proposer $P_\theta$ is Qwen2.5-7B-Instruct at revision `a09a3545` (Qwen Team, 2024); it receives the current state and its legal actions and proposes five next steps per active parent. The learned verifier $V_\phi$ is the same model under a separate scoring prompt. The search policy $S_k$ pools the candidates of a layer, applies legality, duplicate and cycle filters, and retains the top $k$ by verifier score with a cumulative-score tiebreak. The terminal checker $G$ tests goal satisfaction. The diagnostic oracle $O$ computes exact goal distance by breadth-first search over the state graph, offline, after the runs; online oracle access exists only in the explicitly labeled ranking-control arm of Appendix B. The 14B runs of Section 5.3 use Qwen2.5-14B-Instruct at revision `cf98f3b3` for both roles.

**Tasks and sample.** We use generated Gripper, Logistics and Miconic instances, and PlanBench Blocksworld instances (Valmeekam et al., 2023a), with depth caps 24, 48, 36 and 24, respectively. The trajectory analysis of Section 2 covers 100 tasks per generated domain and 80 Blocksworld tasks, three seeds, widths $\{1, 2, 3, 5\}$: 3,600 generated-domain episodes, 960 Blocksworld episodes and 1,222,284 labeled candidates (1,193,748 in the generated domains, 28,536 in Blocksworld). Gripper horizons range from 8 to 18 steps. Blocksworld reachable state spaces hold 125 to 866 states.

**Estimands.** The four measurements below address different parts of the search loop. A feasible expansion starts at positive exact distance, where an improving legal action exists. Retention conditions on that availability; it does not certify that a complete solution path survives.

| Measurement | Definition and interpretation |
|---|---|
| Per-parent retention | Probability that an improving candidate from a given parent survives selection. |
| Per-layer retention | Probability that at least one improving candidate survives anywhere in the layer. |
| Signed progress | $u_t = D_t - D_{t+1}$, with $D_t$ the smallest exact distance on the live frontier; positive values mean progress. Average within task, then across tasks. |
| Pairwise win rate | Within-layer probability that an improving candidate outranks a non-improving candidate; ties count as one half. |

**Replay and depth matching.** Replay bias is width-$k$ retention computed from width-2 traces minus retention in true width-$k$ runs. Intervals bootstrap paired tasks. The depth-matched comparison measures improving rates and win rates at depths common to widths 2 and 5: 24, 27 and 48 common depths in Gripper, Miconic and Logistics, respectively. Figure 7B plots the replay difference itself; online width-$k$ performance is measured only from runs executed at width $k$.

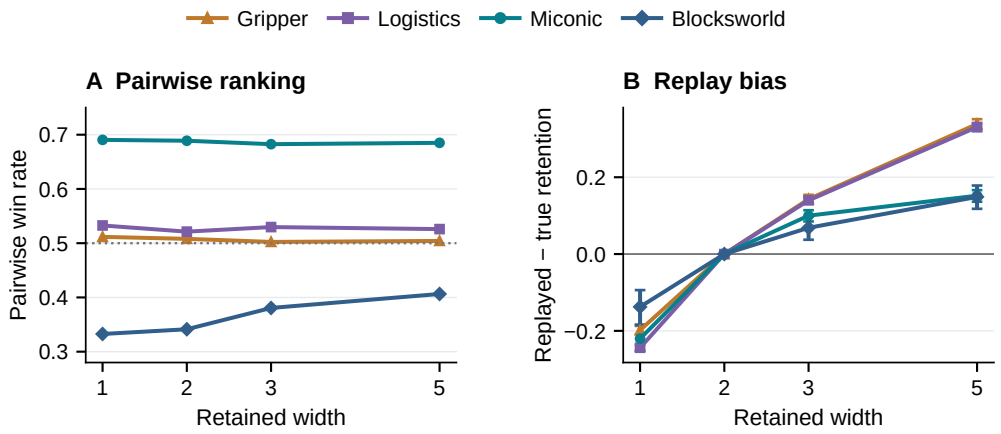


Figure 7: **Ranking quality and replay bias.** **A**: pairwise ranking accuracy by retained width; the dotted line is chance. Win rates barely move with width: ranking quality is a property of the verifier–domain pair. **B**: width-$k$ retention replayed on width-2 traces minus retention in true width-$k$ runs, with task-bootstrap 95% intervals; replay overstates true retention by up to 0.34, which is why the controller's gain labels come from paired continuations at true width. Domain colors and markers as in Figure 2.

## B Component interventions

The intervention campaign uses matched 40-task subsets with three seeds per domain. Arms: the learned system (V1); the same verifier under a strict prompt (V2); a cross-family verifier, Mistral-7B-Instruct-v0.3 (Jiang et al., 2023) (V3); exact-distance ranking in place of the learned ranking (V0, the oracle control); exhaustive legal successors in place of the learned proposer (B); and the original verifier on an easier task distribution (V1easy). Read Figure 8 for the paired oracle-ranking gain and Table 2 for the proposal control; Figure 9 shows the width curves under every verifier condition.

Table 2: Paired gains of oracle ranking (V0 minus V1) and of exhaustive proposal (B minus V1), 40 tasks × 3 seeds, 95% task-bootstrap intervals. Dashes indicate unreported proposal contrasts.

| Domain | $k$ | Oracle ranking gain | Exhaustive proposal change |
|---|---|---|---|
| Gripper | 1 | $+0.875$ $[0.800, 0.942]$ | $0.000$ $[0.000, 0.000]$ |
| Gripper | 2 | $+0.983$ $[0.958, 1.000]$ | $0.000$ $[0.000, 0.000]$ |
| Gripper | 5 | $+0.992$ $[0.975, 1.000]$ | $+0.008$ $[-0.025, 0.050]$ |
| Logistics | 1 | $+0.675$ $[0.558, 0.792]$ | $0.000$ $[0.000, 0.000]$ |
| Logistics | 2 | $+0.958$ $[0.917, 0.992]$ | $+0.008$ $[0.000, 0.025]$ |
| Logistics | 5 | $+0.992$ $[0.975, 1.000]$ | $+0.008$ $[-0.025, 0.050]$ |
| Miconic | 1 | $+0.892$ $[0.800, 0.967]$ | $+0.033$ $[0.000, 0.092]$ |
| Miconic | 2 | $+0.275$ $[0.167, 0.392]$ | $-0.067$ $[-0.175, 0.033]$ |
| Miconic | 5 | $+0.017$ $[0.000, 0.050]$ | $-0.017$ $[-0.042, 0.000]$ |
| Blocksworld | 1 | $+0.825$ $[0.700, 0.925]$ | — |
| Blocksworld | 2 | $+0.750$ $[0.625, 0.867]$ | — |
| Blocksworld | 5 | $+0.158$ $[0.067, 0.267]$ | — |

**Ranking headroom is domain dependent.** The large oracle gains on Gripper and Logistics persist at width 5: the learned ranking is still the bottleneck even after adding redundancy. Miconic differs. Its oracle gain falls from 0.892 at width 1 to 0.017 at width 5, because the wider learned search already approaches the ceiling. Thus an oracle intervention establishes headroom under each state distribution; whether a practicable ranker can attain that headroom is a separate question.

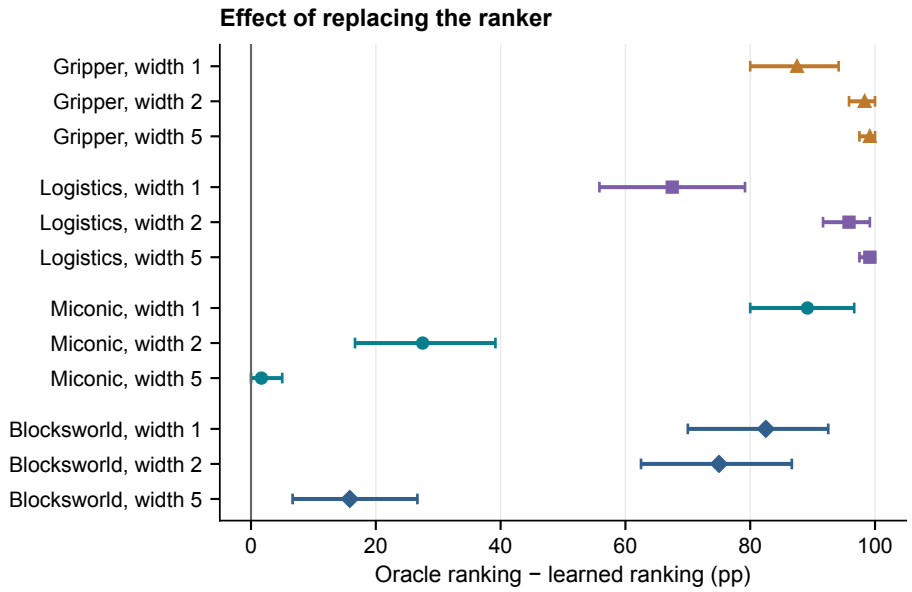


Figure 8: **How much does replacing the ranker buy?** Paired success gains from exact-distance ranking, in percentage points with 95% task-bootstrap intervals (40 tasks × 3 seeds per domain). Read within each domain from width 1 to width 5. Table 2 gives the corresponding proposal intervention; Figure 9 separates verifier and task-distribution changes.

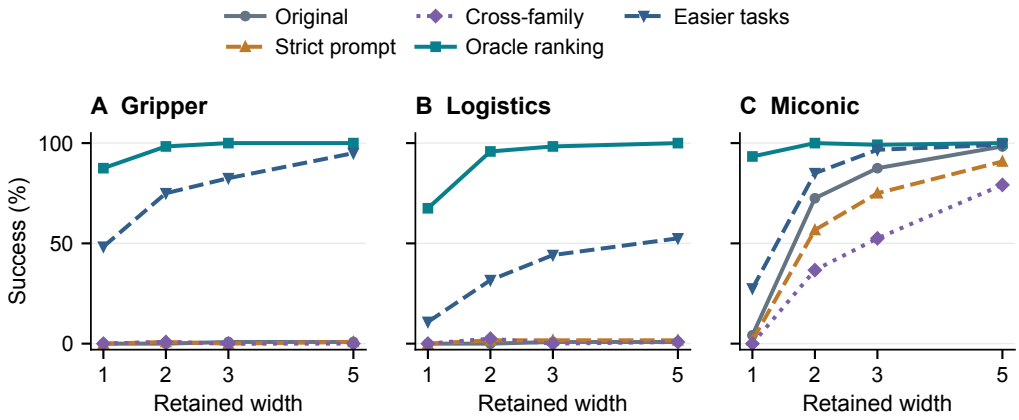


Figure 9: **The same width can behave differently under different ranking conditions.** Each panel is one planning domain; colors and line styles identify the verifier condition. Oracle ranking tests the headroom of ranking, while the easier-task curve changes the state distribution. Each intervention answers its own question; no single calibration curve connects them.

**Prompt changes, model changes and easier tasks are different controls.** The strict-prompt and cross-family curves alter the ranker on the same tasks. The easier-task curve alters the distribution being ranked, while the oracle curve removes ranking error by design. Reading them separately avoids attributing every improvement to verifier calibration. Their common message is that the useful width depends on the verifier together with the states it encounters.

**Why proposal coverage does not explain the oracle gain.** Exhaustive legal proposals leave eight of the nine reported generated-domain cells within 0.033 of the learned proposer. The remaining cell, Miconic at width 2, is negative with an interval spanning zero. These controls support focusing on ranking and allocation in this campaign. They do not rule out proposal failures on other tasks or at other budgets.

## C MATHEMATICS DIAGNOSIS PROTOCOL

This appendix specifies the 195-problem diagnosis in Section 3. The 480-problem E5 panel is disjoint from this sample. Table 3 gives the counts at the primary ceiling; Figure 10 shows how pool coverage and returned accuracy separate across ceilings.

**Sample and models.** MATH problems are balanced across five difficulty levels (39 per level), with one fixed seed. The proposer is Qwen2.5-7B-Instruct; the verifier is Qwen2.5-Math-PRM-7B at revision `06107400`. PRM scores are mean log step probabilities.

**Generation, selection and accounting.** Beam search proposes five steps per active parent, with at most 256 generated tokens per step and 12 layers per episode. Sampling generates each complete solution in one call of at most 3,072 tokens, up to 128 solutions under the same ceiling, and returns the highest-scored answer. Voting variants regroup that same pool by exact rational and symbolic equivalence. Ceilings are 32,768, 65,536 and 131,072 total model tokens, including proposer input and output and verifier input. A candidate contributes to the completed pool only when its generation and required scoring finish within the ceiling.

**Offline grading and audit.** Grading uses exact rational comparison and pinned symbolic equivalence. A blinded author audit re-graded 105 sampled answer pairs: 102 agreed with the machine grader (97.1%), and three were machine false negatives (one percentage written as `10%`, two spellings of `12th grade`). Correcting them moves sampling from 170 to 172 correct, random ranking at widths 1 and 5 from 150 to 151 and from 138 to 139, and leaves both PRM search cells unchanged. Neither the ordering nor the verdicts of the three registered contrasts change. The paper consistently reports the machine-graded numbers, using one grader across runs.

Table 3: Outcomes at 65,536 tokens for 195 MATH problems. Pool: a correct answer was completed and scored. Selection loss: pool held a correct answer and another was returned.

| Policy | Correct returned | Correct in pool | Selection loss | No completion | Wrong only |
|---|---|---|---|---|---|
| Sampling + PRM | 170 | 189 | 19 | 0 | 6 |
| PRM search $k$=1 | 159 | 159 | 0 | 21 | 15 |
| PRM search $k$=5 | 82 | 82 | 0 | 112 | 1 |
| Random ranking $k$=1 | 150 | 151 | 1 | 6 | 38 |
| Random ranking $k$=5 | 138 | 147 | 9 | 32 | 16 |

**Read the two gaps separately.** For sampling, the dashed curve is already high while returned accuracy remains lower: selection loses some discovered answers. For width-5 search, both curves start low and rise together with more budget: failure occurs mainly before the selector receives a usable pool. Table 3 makes the denominator explicit. For each method, correct returns plus selection losses equal correct-containing pools, and the four disjoint outcomes sum to 195. This is why a completion reserve and a better final selector address different losses.

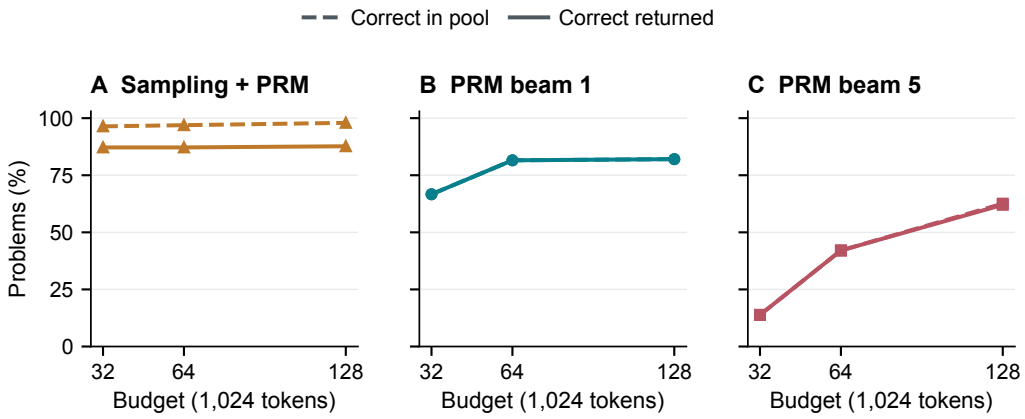


Figure 10: **Discovery and selection across token ceilings.** For each method, dashed curves show the percentage of problems with a correct answer in the completed pool, and solid curves show the percentage returned correctly. Their vertical gap is selection loss; the gap from pool coverage to 100% combines wrong-only pools and non-completion. All 195 problems are in each denominator. The full 0–100% scale includes the low-budget width-5 result.

## D CONTROLLER DETAILS

Read this appendix in implementation order: score representation, gain features, completion records, then the action rule. Appendix H examines the resulting decisions; Appendix E states the conditional guarantees.

**Scores.** The controller consumes verifier scores on one common scale: a candidate's log score $s$ maps to quality $q = \exp(\min\{0, s\}) \in [0, 1]$. For the math PRM, $s$ is the mean log step probability. Quality enters the gain features without per-layer rescaling, so the magnitude of a gap in $q$ means the same thing at every layer. Candidates are ordered by descending score with a deterministic key tiebreak.

**Gain-model features.** Write $a = (k-1)/4$, $\ell_b = \log(1 + b/\hat{c})$ and $\ell_d = \log(1 + D - d)$. The 16 features are grouped below; every feature vanishes at $k = 1$, fixing $\hat{\Delta}(1) = 0$.

| Group | Features |
|---|---|
| Action and resources | $a$, $a^2$, $a\ell_b$, $a\ell_d$, $a$·used-depth fraction |
| Score profile | $a$·top score, $a$·top-two gap, $a$·score standard deviation, $a$·mean of ranks $2{:}k$ |
| Candidate diversity | $a$·novelty of the retained candidates |
| Observed cost | $a \log(\text{cost trend})$, $a$·generation share, $a \log(1 + \text{context tokens}/\hat{c})$ |
| Interactions | $a(\ell_b - \ell_d)$, $a \cdot \text{gap} \cdot \ell_b$, $a \cdot \text{novelty} \cdot \ell_b$ |

Novelty is one minus the largest word-set Jaccard similarity to previously retained candidates, averaged over newly retained candidates. It uses public state text in planning and generated steps in math. Features are scaled by their weighted root mean square on the fit data. Tasks carry equal total weight, with equal weight for states within each task. The fitted artifact records the reference policy as "first action, then fixed width 1".

**Fitting and prediction.** For state $i$ and width $k > 1$, the target is $y_{ik} = Y_i(k) - Y_i(1)$ from paired continuations. A task's weight is divided equally among its states and then among each state's non-narrow actions. After normalizing weights to mean one, each feature is divided by its weighted root mean square, floored at $10^{-3}$. With scaled feature matrix $Z$ and diagonal weight matrix $\Omega$, the fit is

$$\beta = (Z^\top \Omega Z + I)^{-1} Z^\top \Omega y, \qquad \hat{\Delta}(k) = \text{clip}(z_k^\top \beta, -1, 1).$$

The ridge coefficient is 1, with no separate intercept. These details specify both the regularization scale and why the width-1 prediction is exactly zero.

**Completion model.** Completion records come from the width-1 continuations already collected for the paired gains, so no extra data is needed. The recorded cost $L$ runs from the end of the next expansion to the first new usable answer and includes generation and required scoring; the next expansion itself is charged as $k\hat{c}$. Events are completed, failed (configured stop without completion) and censored (budget or time). Only healthy records enter; backend errors are resolved first.

**Local retrieval and reserve.** To estimate completion for the current state, the controller finds the most similar development states and reuses their completion records. Similarity is measured on five features that describe where a search stands: remaining depth $\log(1 + D - d)$, used-depth fraction $d/D$, context size per unit of expansion cost $\log(1 + \ell/\hat{c})$, the share of charged tokens spent on generation $g$, and the cost trend $\log \tau$ (current over preceding expansion cost). Features are centered and standardized on source states, with standard deviations floored at 0.1; Euclidean distance selects one nearest state per source task and at most 32 tasks. Because tasks differ in per-step cost, times are normalized by the observed per-parent cost before retrieval and rescaled by the current $\hat{c}$. With $t_{\max}$ the end of local support, $U = \lceil \inf\{t : F(t) \geq 0.85F(t_{\max})\} \rceil$ in token units, and ties in time process events before censorings. $F$ and $U$ are used separately, so a low completion probability is never masked by a finite conditional quantile.

**Algorithm 1** SCA+ width decision. Input: $n \leq 5$ candidates sorted by verifier score, tokens $b$, cost $\hat{c} > 0$, depth $d$ and cap $D$, frozen gain and completion models. Output: retained width $k$; $k = 0$ stops expansion.

**if** $n = 0$ or $d \geq D$ **then**
  **return** $0$
**end if**
**if** $b < \hat{c}$ **then**
  **return** $1$ ledger still checks each call
**end if**
Retrieve nearby width-1 development records; estimate $(F_x, U_x)$
**if** outside feature support, fewer than 12 source tasks, or fewer than 4 completions **then**
  $h \leftarrow \max(0, D - d - 1)$ depth after the next expansion
  $k_{\text{aff}} \leftarrow \max\{1, \min\{n, \lfloor b/\hat{c} \rfloor - h\}\}$
  **return** $\min(2, k_{\text{aff}})$ bounded depth-reserve fallback
**end if**
$A \leftarrow \{k \in \{1, \ldots, n\} : b - k\hat{c} \geq U_x,\ F_x(b - k\hat{c}) \geq 0.20\}$
$W \leftarrow \{k \in A : k > 1,\ \hat{\Delta}(k) > 0.02\}$
**if** $W = \emptyset$ **then**
  **return** $1$
**end if**
**return** $\arg\max_{k \in W} \hat{\Delta}(k)$, ties to smaller $k$

**Support and execution.** A state is outside completion-model support when any standardized retrieval feature is more than two units beyond the source minimum or maximum. Insufficient support invokes the bounded fallback in Algorithm 1; it can return width 1 when the budget or candidate set cannot support width 2. Returning width 1 with no admitted action expresses a decision to try a narrow continuation even though the reserve may be uncovered.

After the width decision, the engine expands the chosen candidates, moves completed answers into the pool, scores new candidates, and updates the observed cost and trend. Before each model call, the ledger checks that its prompt and maximum output are payable, then settles the actual cost. The ledger can stop execution even when the width rule returns a positive action; it enforces the ceiling independently of forecast accuracy.

**Development.** Sixty-four source planning tasks provide the paired continuations for fitting, 32 held-out tasks report selection gain, reference narrow-search gain, action-oracle gap and fallback

rate, and 24 further tasks select between setting A (gain threshold 0.02, quantile 0.85) and setting B (0.05, 0.90). Setting A was selected. Development spent 22,361,472 tokens in total; against BoN at the primary MATH ceiling SCA+ saves 2,089 tokens per problem, so this one-time cost is repaid after about 10,700 problems. All parameters were then frozen for E4, E5 and E6.

## E Decision loss and budget feasibility

The two statements below separate decision quality from budget enforcement. They concern a fixed reference continuation and explicit premises; neither is a calibration result for the deployed controller.

**Decision loss.** Fix the reference continuation and an affordable set $A$ that contains width 1. Suppose $|\hat{\Delta}(k) - \Delta(k)| \le \varepsilon$ for every $k \in A$. Let $k^\star$ maximize $\Delta$ on $A$ and $\hat{k}$ maximize $\hat{\Delta}$ on $A$. Then

$$\Delta(k^\star) - \Delta(\hat{k}) = \big[\Delta(k^\star) - \hat{\Delta}(k^\star)\big] + \big[\hat{\Delta}(k^\star) - \hat{\Delta}(\hat{k})\big] + \big[\hat{\Delta}(\hat{k}) - \Delta(\hat{k})\big] \le \varepsilon + 0 + \varepsilon = 2\varepsilon.$$

With the widening threshold $\delta$ the controller may keep width 1 when $\hat{\Delta}(\hat{k}) \le \delta$, adding at most $\delta$; if filtering excludes the unrestricted optimum, an additional oracle gap $\gamma = \max_k \Delta(k) - \max_{k \in A} \Delta(k)$ must be added. The bound is conditional on $\varepsilon$ and concerns one decision under the reference continuation.

**Feasibility.** If the current expansion costs at most $k\hat{c}$, the retained reference path completes within $U$, and $k\hat{c} + U \le b$, the completion falls inside the budget. The conditional quantile does not by itself certify the second premise; the per-call ledger, which checks payability before each call, is what enforces the ceiling at run time.

**What the bound explains.** The three terms correspond to three distinct errors: $2\varepsilon$ for estimating the gain, $\delta$ for requiring a minimum gain before widening, and $\gamma$ for excluding an otherwise useful width. The reserve trades this last possibility against preserving a path to completion. Neither the bound nor the accounting condition estimates these errors from data, and neither transfers automatically to a policy that changes width again at the next layer.

## F Baseline adaptations

Verifier-based arms share the generator, verifier, budget ledger and final selector. The no-PRM vote and BG early stopping are explicit exceptions. The following paragraphs specify each adaptation. The two trace examples at the end illustrate why early stopping can either preserve or overturn a final vote.

**Prompt interface for E4–E6.** Planning prompts provide the current state, exact goal specification, domain guidance, the eight most recent actions and numbered legal actions. The proposer requests distinct action indices in JSON at temperature 1.0; the verifier sees the parent, action and successor and returns `sure`, `likely` or `impossible` at temperature 0.1. These ratings map to 10, 1 and 0.1 before taking logs. Goal checking uses the public task specification; the distance oracle stays offline.

For math, the system instruction is: "Please reason step by step, and put your final answer within `\boxed{}`." The problem forms the user message. Step generation appends prior numbered steps and the next `Step i:` prefix to the chat template, and stops at a blank-line boundary or 256 output tokens. The base temperature is 1.0; full-solution calls allow 3,072 output tokens. The PRM receives the same problem and steps separated by `<extra_0>`. The no-PRM vote cycles temperatures 0.7, 1.0 and 1.3. The accompanying implementation snapshot contains the literal templates and formatting functions.

**REBASE and REBASE+F.** REBASE allocates expansion quotas across nodes by a softmax over rewards with sequential rounding up, following the reference implementation (Wu et al., 2025), on the shared mean log score with target quota 16 and temperature 0.1. Its generator, verifier, budget

ledger and final selector match the verifier-based comparison arms. REBASE+F applies SCA+'s completion reserve to cap the expansion quota of each layer; everything else is unchanged.

**Sampling arms.** BoN samples complete solutions in one call each, scores them with the PRM, groups them into equivalence classes by exact rational and symbolic comparison without reference answers, and returns the class with the largest summed PRM weight. The no-PRM vote samples at several temperatures and returns the largest equivalence class without any PRM call.

**Shared weighted selector.** Answers are grouped without reference solutions. Exact rational equality is tried before bidirectional symbolic equivalence; a new answer joins a class only if it agrees with every member. Unknown comparisons leave answers in separate classes. Each trajectory contributes $\exp(s_j - s_{\max})$, where $s_{\max}$ is the largest score in the pool; this common normalization leaves the winning class, and hence the sign of the vote margin (Equation 2), unchanged. Ties in class weight favor the class formed first in completion order. The count-vote arm substitutes class size for weight; the maximum-score variant selects one trajectory with a deterministic key tiebreak.

**BG-MCTS and BG-stop.** Budget-guided MCTS (Miyamoto et al., 2026) is adapted to the shared cost accounting.

**Selection.** Each descent from the root compares existing children with the option of generating a new child. Let $\rho$ be the remaining-budget fraction. An existing child scores its subtree mean of $Q + (1 - \rho) \cdot (\text{unfinished depth}/\text{mean answer depth})$ plus an exploration term $\rho\sqrt{2}\,\text{prior}\sqrt{\log(\text{parent size})/\text{child size}}$; a new child scores the sibling mean of $Q$ plus $\rho$ times the sibling variance of $Q$. Completed nodes have zero depth term and may open a revision segment.

**Expansion and payment.** Leaf expansion generates two children; choosing the new-child option generates one. Before each generation the maximum output and full-prefix scoring cost are reserved and paid before a candidate enters the pool. The original method budgets output tokens and selects by maximum score; here it is charged for generation input and output and verifier input like every other arm, and both weighted-vote and maximum-score selection are reported.

**Early stopping.** BG-stop stops the same run once at least 60% of the budget is spent, at least two complete scored answers exist and the best completed score reaches 0.70. Figure 11 shows two paired runs in which continuing past the stopping point repairs and reverses the answer.

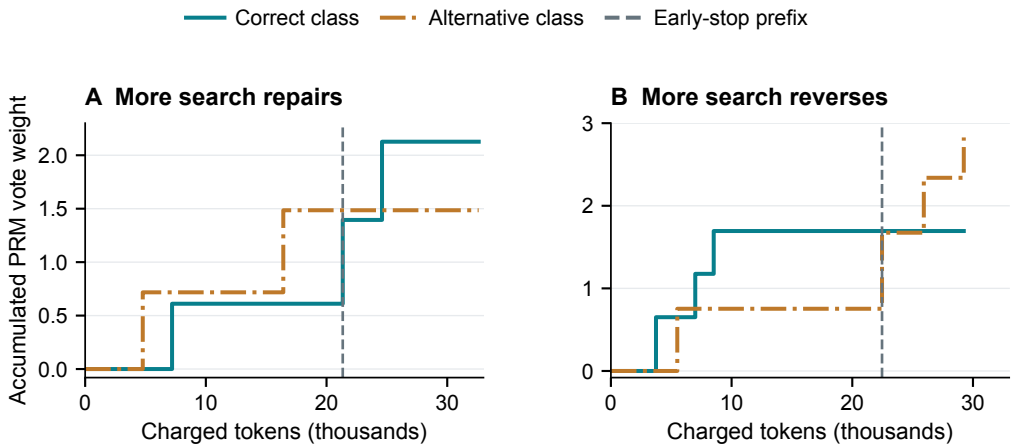


Figure 11: **Continuing a BG-MCTS run past its early stop.** Accumulated PRM vote weight of the correct and the leading alternative answer class along charged tokens for MATH 0066 (left) and MATH 0044 (right), seed 0; the dashed line marks the early-stop prefix. On the confirmation panel the full run wins 46 and loses 11 of the 1,440 paired runs against its early stop.

**A later completion matters through its vote weight.** In the left example, the correct class overtakes the alternative after the early-stop prefix; in the right, the alternative overtakes the correct class. The selector is unchanged in both comparisons. These examples make the final pool-to-vote link concrete; they compare continuation with stopping, and SCA+'s own decisions are examined through the ablation analysis of Appendix H.

## G STATISTICAL PROCEDURES

**Unit and uncertainty.** The unit is the task. Accuracy is averaged over seeds within each task before comparison. Paired differences use 4,000 task-bootstrap resamples (Efron & Tibshirani, 1994) for E4–E6 and 10,000 for the BG comparisons; the reported intervals are 95% intervals. Sign-flip tests over task clusters use 20,000 permutations, resolving $p$-values to $5 \times 10^{-5}$.

**Declared comparisons.** Holm's procedure (Holm, 1979) adjusts $p$-values over the declared primary family of 19 contrasts, and separately over the two BG contrasts. The 19 contrasts are SCA+ against each comparison arm at the primary budget: eight in E4 (fixed width 2, REBASE, REBASE+F, SCA and the four ablations) and eleven in E5 (BoN, the two no-PRM votes, fixed width 2, REBASE, REBASE+F, SCA and the four ablations). All 19 reach the resolution floor of the sign-flip test after adjustment (Holm-adjusted $p = 9.5 \times 10^{-4}$). The E6 non-inferiority margin is $-2\,\mathrm{pp}$. Bootstrap intervals and multiplicity-adjusted tests are reported as distinct summaries.

**Paired counts and panels.** Wins, losses and ties in the E4–E6 tables count tasks. The BG package instead reports run-level pairs (1,440 per panel), so the BG contrasts are not assigned task-level win/loss counts in the comparison tables. Core panels (60 planning tasks or 80 MATH problems, one seed) support inspection of individual records and are not pooled with confirmation panels. E4 and E6 share their 360 tasks.

## H FROM LAYER DECISIONS TO ANSWER-POOL COMPETITION

Section 3 describes the structural pathway from width to completed trajectories to the final vote. This appendix examines its intermediate quantities. Figure 12 and Table 4 locate changes in the controller's actions; Figure 13 and Table 5 describe the completed pools. The last subsection states exactly what can be inferred from a fixed selector and what needs an intervention on a width decision.

### H.1 HOW THE CONTROLLER DISTRIBUTES EXPLORATION ACROSS LAYERS

Figure 12A shows mean E4 width near 1.9 in layers 1–3, 1.53 at layer 4 and 1.15 in layers 5–9. Without the gain model, width is usually 3 early and 2 later; without the completion filter, its mean stays near 3.4. The gain model avoids taking every affordable branch; the completion constraint limits continued breadth.

Panel B regroups the same actions by remaining budget instead of depth, giving a second descriptive view of the same decisions. Panel C shows that the frozen rule has a similar overall action mix across these settings: width 1 accounts for about 55% of expansions, width 2 for 34%, and width 3 for 11%. Similar marginal distributions alone do not establish that the same state features drive each decision.

### H.2 WHY THE MEAN TRAJECTORY IS NOT A FIXED SCHEDULE

Table 4 conditions on layer and budget to distinguish individual actions from the average curve. All three widths occur at layer 1 with full budget, excluding a fixed first-layer width. Scores, context and observed costs can all contribute to this variation.

From layer 5 onward, width 1 accounts for 80–95% of decisions within the reported budget bins. Early state-dependent choices thus lead into predominantly narrow continuation; whether those choices are useful requires linking decisions to subsequent outcomes.

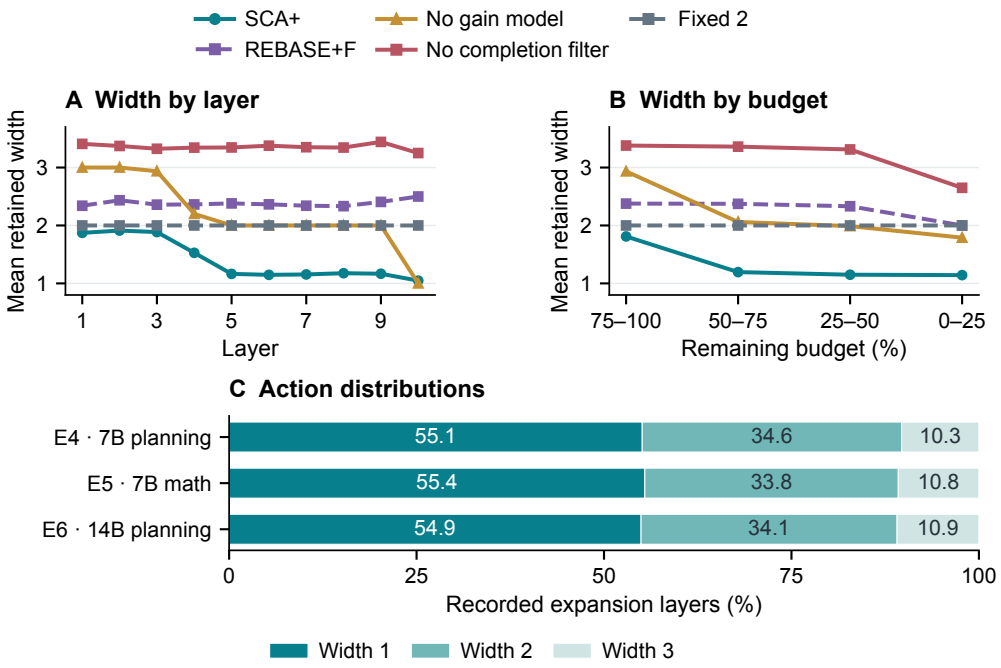


Figure 12: **Layer decisions under the two controller components.** **A**: mean retained width by E4 layer. **B**: the same actions grouped by remaining-budget fraction. **C**: width distributions in E4, E5 and E6. E4/E5 use 32,768-token ceilings; E6 uses 65,536. Means condition on recorded expansion layers, so later layers contain only runs that are still active. Widths 4 and 5 have zero share in these records.

Table 4: Conditional width choices in E4 at 32,768 tokens. Percentages are rounded; the last row sums to 101% because of rounding. Layer 2 has no budget-bin condition in that summary.

| Layer | Remaining budget | Width 1 | Width 2 | Width 3 |
|---|---|---|---|---|
| 1 | 100% | 31% | 50% | 19% |
| 2 | All recorded states | 28% | 53% | 19% |
| 4 | More than 75% | 52% | 37% | 11% |
| 4 | 50–75% | 65% | 30% | 6% |

### H.3 What the completed-answer pool adds to the explanation

The E5 outcome records give the key contrast: SCA+ and BG-MCTS contain a correct answer in 1,402 and 1,403 of 1,440 runs, but select it in 1,330 and 1,262. Figure 13 asks how the templated pools differ despite nearly equal discovery.

**Denominators and grouping.** Two bookkeeping choices matter when reading Table 5. First, the diagnostic grouping uses exact rational or normalized-string equality. Second, its conditional plurality is the frequency with which a correct class has the largest weight, among runs containing a correct class. Multiplying this rate by correct-class presence gives an overall rate, which agrees with the official selection outcomes within about 0.6 percentage points in the accompanying summaries. The official selector also uses symbolic comparison (Appendix F), so these diagnostics and official counts are separate estimands. Table 5's means and percentiles are marginal summaries; they cannot recover the joint weight of the correct class and its strongest competitor.

BG-MCTS has fewer answer classes and a larger mean correct-class weight when correct is present. Yet, conditional on correct-class presence, it wins a correct plurality less often than SCA+. There is no contradiction: the mean weight does not determine how often the correct class beats its strongest

Table 5: Templated-pool summaries for E5 at 32,768 tokens. "Present" is a percentage of all runs; plurality and vote-share summaries condition on correct-class presence. Classes is the mean number of distinct answer classes.

| Method | Present (%) | Classes (mean) | Correct vote share Mean | Correct vote share 10th pct. | Correct plurality (%) |
|---|---|---|---|---|---|
| SCA+ | 97.5 | 2.96 | 0.48 | 0.50 | 94.2 |
| BoN | 94.6 | 2.89 | 0.46 | 0.13 | 87.7 |
| REBASE+F | 95.0 | 2.89 | 0.46 | 0.13 | 89.4 |
| BG-MCTS | 97.4 | 1.84 | 0.73 | 0.50 | 90.0 |
| BG-stop | 96.7 | 1.78 | 0.74 | 0.46 | 88.1 |

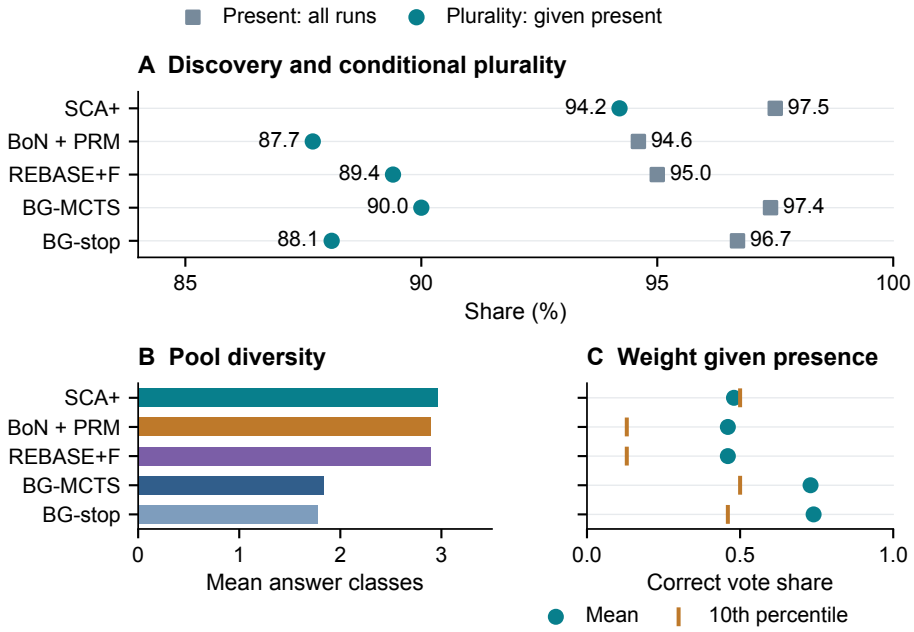


Figure 13: **Completed-pool structure in E5 at 32,768 tokens.** **A**: correct-class presence across all runs and plurality conditional on presence; these have different denominators. **B**: mean answer-class count. **C**: mean and 10th percentile of correct-class vote weight, conditional on presence. These summaries describe the templated pools.

competitor. SCA+ and sampling have similar average class counts and mean correct weight, but different plurality rates. The distribution across runs, and the runner-up within each run, matter more than a single mean.

### H.4 Causal pathway and attribution to a width decision

**The pool mediates the selector's response.** Fix a problem, budget, generator, verifier and selection rule. Let $\mathcal{P}(k)$ be the ordered, scored pool produced after intervening on one width decision and applying the same continuation policy thereafter. The returned answer is $g(\mathcal{P}(k))$, where $g$ is the deterministic selector. If two interventions produce identical pools, including order and tie metadata, they return the same answer. Consequently, an effect of width on the returned answer must pass through changes in the pool. This follows from the search procedure; it does not require a statistical mediation estimate.

**A vote change requires a change in the winning class.** For PRM-weighted voting, let $a_j, s_j$ be completed trajectory $j$'s answer class and log score, and $\mathcal{C}^\star$ the classes graded correct offline. The class weights and the vote margin of Section 3 are

$$W(c) = \sum_{j:a_j=c} \exp(s_j), \qquad m(\mathcal{P}) = \max_{c\in\mathcal{C}^\star} W(c) - \max_{c\notin\mathcal{C}^\star} W(c), \tag{2}$$

with an empty maximum set to zero; for nonempty pools, $m > 0$ returns a correct answer and $m < 0$ a wrong one, apart from ties, which use a fixed rule. For a fixed grouping, adding a trajectory to class $c$ adds $\exp(s_j)$ to $W(c)$. When both correct and incorrect classes are present, the sign of $m(\mathcal{P})$ determines whether the winning class is correct. A width change can alter that sign by admitting useful descendants, diverting work toward a competitor, or preventing later completions through its cost. The BG prefix examples in Figure 11 exhibit both directions of the final link. They also show why class count or average correct-class weight alone cannot predict selection accuracy.

**Empirical attribution: narrowing to vote flip in E5.** Among the 1,520 paired E5 runs at 32,768 tokens (1,440 confirmation and 80 core), SCA+ selects correctly in 199 cases where the no-gain-model ablation does not, and loses 28 the other way. In the outcome taxonomy of Equation 1, the 199 wins split into 100 where the ablation's pool held a correct answer that lost the vote, 93 where its pool held only wrong answers, and just 6 where it completed nothing; the reserve, present in both arms, keeps non-completion out of this comparison. All 199 involve narrowing: SCA+ retains 1.1–2.0 candidates per layer versus the ablation's 2.0–3.0, across an average of 5.6 layers per run. The narrowing saves 6.5% of the ablation's charged tokens (1,779 tokens on average) and raises per-candidate token investment by 44–60% (charged tokens per retained candidate, with generation output per candidate rising in the same proportion), concentrating generation on fewer but better-explored paths.

Both arms produce one correct terminal completion on average, and the count is identical in 193 of the 199 cases; what differs is how those completions enter the pool. SCA+'s arrive with higher verifier scores, the pool's highest correct-class score averaging 0.926 versus 0.755, and its narrow pools put less weight on wrong classes, so the vote margin $m(\mathcal{P})$ moves from both sides at once. The 93 wrong-only wins show the sharper form of the same channel: the ablation's correct completion never became a usable pool entry at all, consistent with the admission rule that a completion counts only once its required scoring finishes within the ceiling (Appendix C). The 28 losses are the mirror image: narrowing concentrated weight on a wrong class while the wider pool kept a winning correct one. Narrowing controls concentration; the verifier decides whether the concentrated branch is right, and here it is right in 199 of the 227 decisive runs. Table 6 summarises the same comparison across all ablation arms; wins are gross counts, so each row is read with its reverse direction. The no-completion-filter row shows the other failure layer: 164 of its 357 wins come from the ablation completing nothing at all.

Table 6: Trajectory-level attribution of SCA+ selection wins at 32,768 tokens in E5. Each row counts runs where SCA+ selects correctly and the ablation does not. $\overline{\Sigma\Delta W}$: mean cumulative width saved per run. $\overline{\Delta\mathrm{tok}}$: mean token saving. $\overline{\Delta\mathrm{corr}}$: mean extra correct completions.

| Ablation | Wins | $\overline{\Sigma\Delta W}$ | $\overline{\Delta\mathrm{tok}}$ | $\overline{\Delta\mathrm{corr}}$ |
|---|---|---|---|---|
| No gain model | 199 | 6.6 | +1,779 | +0.0 |
| No completion filter | 357 | 12.8 | +4,061 | +0.5 |
| SCA (no learned models) | 129 | 3.0 | +1,871 | +0.2 |
| Pooled statistics | 174 | 2.3 | +1,355 | +0.0 |
| Depth schedule | 238 | 2.6 | +1,957 | +0.2 |

These observations are consistent with the chain: narrowing → per-candidate budget concentration → stronger vote weight for the surviving branches' classes → correct selection when a surviving branch is right. The direction depends on the descendants those retained candidates produce. Attributing an individual width decision would require forked continuations from matched states, which we have not run; the ablation comparison isolates each component's contribution within the E5 setting.

# I EXTENDED RESULTS AND TRANSFER

The full comparisons separate three operating regimes: allocation on 7B planning, cost at 14B saturation, and discovery versus selection on mathematics. Method abbreviations match the main text; "no gain" takes the largest affordable width, and "no completion filter" removes the reserve constraint. Read accuracy and realized tokens together: a ceiling limits spending but does not require a method to use it all.

Table 7: E4: 7B planning at 32,768 tokens. Accuracy in percent; differences are SCA+ minus each row with 95% task-bootstrap intervals.

| Method | Accuracy | Tokens | Difference (pp) |
|---|---|---|---|
| Fixed $k$=1 | 70.28 | 18,957 | +17.87 [14.81, 20.83] |
| Fixed $k$=2 | 73.43 | 25,456 | +14.72 [12.13, 17.50] |
| Fixed $k$=5 | 56.20 | 27,402 | +31.94 [27.96, 35.93] |
| REBASE | 77.04 | 26,208 | +11.11 [8.61, 13.70] |
| REBASE+F | 79.07 | 24,179 | +9.07 [6.76, 11.39] |
| No gain model | 77.78 | 24,060 | +10.37 [8.06, 12.96] |
| No completion filter | 69.26 | 27,168 | +18.89 [15.93, 22.04] |
| Pooled statistics | 79.26 | 24,262 | +8.89 [6.85, 11.11] |
| Depth schedule | 76.30 | 23,995 | +11.85 [9.26, 14.54] |
| SCA | 81.67 | 22,519 | +6.48 [4.54, 8.61] |
| SCA+ | 88.15 | 20,108 | |

**Planning: preserving completion is not sufficient by itself.** Table 7 first compares fixed widths, then methods that redistribute work. REBASE+F improves on REBASE, but taking the largest reserve-admitted width (no gain model) remains below SCA+. The decisive comparison is whether affordable expansion is also useful.

Table 8: E6: 14B planning at 65,536 tokens. Accuracy in percent; differences are SCA+ minus each row with 95% task-bootstrap intervals.

| Method | Accuracy | Tokens | Difference (pp) |
|---|---|---|---|
| Fixed $k$=1 | 88.43 | 16,628 | +10.28 [7.96, 12.87] |
| Fixed $k$=2 | 96.85 | 33,398 | +1.85 [0.74, 3.06] |
| Fixed $k$=5 | 94.35 | 51,626 | +4.35 [2.78, 6.02] |
| REBASE | 96.11 | 37,362 | +2.59 [1.30, 3.98] |
| REBASE+F | 98.24 | 31,603 | +0.46 [−0.56, 1.57] |
| No gain model | 97.69 | 34,099 | +1.02 [0.09, 2.04] |
| No completion filter | 96.30 | 48,087 | +2.41 [1.30, 3.52] |
| Pooled statistics | 97.87 | 33,550 | +0.83 [−0.09, 1.76] |
| Depth schedule | 96.39 | 34,188 | +2.31 [1.30, 3.52] |
| SCA | 97.96 | 27,407 | +0.74 [−0.28, 1.76] |
| SCA+ | 98.70 | 22,974 | |

**At saturation, distinguish comparable accuracy from equal cost.** The SCA+–REBASE+F interval in Table 8 spans zero, so it does not establish higher accuracy. The stated non-inferiority comparison concerns the $-2\,\mathrm{pp}$ margin. The large token separation is the informative result in this regime; fixed width 1 is cheaper but remains substantially less accurate.

**Mathematics at the primary ceiling.** Table 9 includes all voting, tree-search and controller ablations. The no-PRM vote separates the cost of verification from the benefit of search, while the single-temperature vote isolates the sampling mixture. BG-stop spends fewer tokens than SCA+ but returns fewer correct answers. These are accuracy–cost tradeoffs, so the full frontier complements each paired difference.

Table 9: E5: 7B math at 32,768 tokens. Accuracy in percent; differences are SCA+ minus each row with 95% task-bootstrap intervals.

| Method | Accuracy | Tokens | Difference (pp) |
|---|---|---|---|
| Fixed $k$=1 | 68.68 | 14,696 | +23.68 [20.76, 26.74] |
| Fixed $k$=2 | 75.14 | 27,953 | +17.22 [14.79, 19.72] |
| Fixed $k$=5 | 59.10 | 29,077 | +33.26 [30.00, 36.60] |
| Vote, no PRM | 81.60 | 27,691 | +10.76 [8.68, 12.92] |
| Vote, single temperature | 77.78 | 27,671 | +14.58 [12.29, 17.01] |
| BoN + PRM vote | 83.33 | 27,755 | +9.03 [7.01, 11.11] |
| REBASE | 81.67 | 27,686 | +10.69 [8.54, 12.92] |
| REBASE+F | 84.31 | 27,278 | +8.06 [6.32, 10.00] |
| BG-stop | 85.21 | 22,725 | +7.15 [5.00, 9.44] |
| BG-MCTS | 87.64 | 30,578 | +4.72 [2.64, 6.81] |
| No gain model | 81.11 | 27,251 | +11.25 [9.10, 13.40] |
| No completion filter | 69.24 | 28,366 | +23.13 [20.28, 26.04] |
| Pooled statistics | 82.85 | 27,381 | +9.51 [7.50, 11.60] |
| Depth schedule | 77.99 | 26,880 | +14.38 [11.94, 16.88] |
| SCA | 86.32 | 26,784 | +6.04 [4.17, 8.06] |
| SCA+ | 92.36 | 25,666 | |

Table 10: E5: 7B math at 65,536 tokens. Accuracy in percent; differences are SCA+ minus each row with 95% task-bootstrap intervals.

| Method | Accuracy | Tokens | Difference (pp) |
|---|---|---|---|
| Vote, no PRM | 89.44 | 55,137 | +7.57 [5.69, 9.58] |
| BoN + PRM vote | 92.22 | 55,259 | +4.79 [3.26, 6.39] |
| REBASE+F | 93.33 | 49,282 | +3.68 [2.36, 5.00] |
| BG-stop | 90.42 | 42,412 | +6.60 [4.93, 8.33] |
| BG-MCTS | 94.86 | 62,857 | +2.15 [0.90, 3.47] |
| SCA+ | 97.01 | 42,438 | |

**A larger ceiling reduces, but does not remove, selection differences.** Table 10 compares each method at the same larger ceiling. Sampling and tree search improve with extra work, while SCA+ uses only 42,438 of the 65,536 available tokens on average. A within-ceiling comparison answers which policy uses the same allowance better; the cross-budget comparison below asks how much allowance is needed to reach comparable accuracy.

Table 11: Breakdowns on the confirmation panels. Top: E5 accuracy by MATH difficulty level at 32,768 tokens. Bottom: E4 (32,768) and E6 (65,536) accuracy by planning domain.

| Level | Problems | BoN | SCA+ |
|---|---|---|---|
| 1 | 6 | 94.4 | 100.0 |
| 2 | 78 | 90.6 | 98.3 |
| 3 | 102 | 89.5 | 95.4 |
| 4 | 144 | 85.2 | 92.8 |
| 5 | 150 | 73.1 | 86.4 |

| | Domain | Fixed $k$=2 | REBASE+F | SCA+ |
|---|---|---|---|---|
| E4 | Gripper | 71.7 | 75.8 | 84.7 |
| E4 | Logistics | 77.8 | 82.2 | 92.8 |
| E4 | Miconic | 70.8 | 79.2 | 86.9 |
| E6 | Gripper | 95.6 | 97.8 | 97.5 |
| E6 | Logistics | 98.6 | 98.9 | 99.4 |
| E6 | Miconic | 96.4 | 98.1 | 99.2 |

**Breakdowns qualify the aggregate.** Table 11 shows an E5 advantage at every difficulty level, but level 1 contains only six problems and should carry little weight on its own. The E4 domain-level gains are consistent in direction. At 14B, however, SCA+ is slightly below REBASE+F on Gripper (97.5% versus 97.8%), while leading on the other two domains. The aggregate saturation result should not be read as a gain in every domain.

**The stronger-model regime.** At 14B, compare horizontal separation at similar accuracy in Figure 14A before reading paired differences in B. Saturation makes token cost the informative axis.

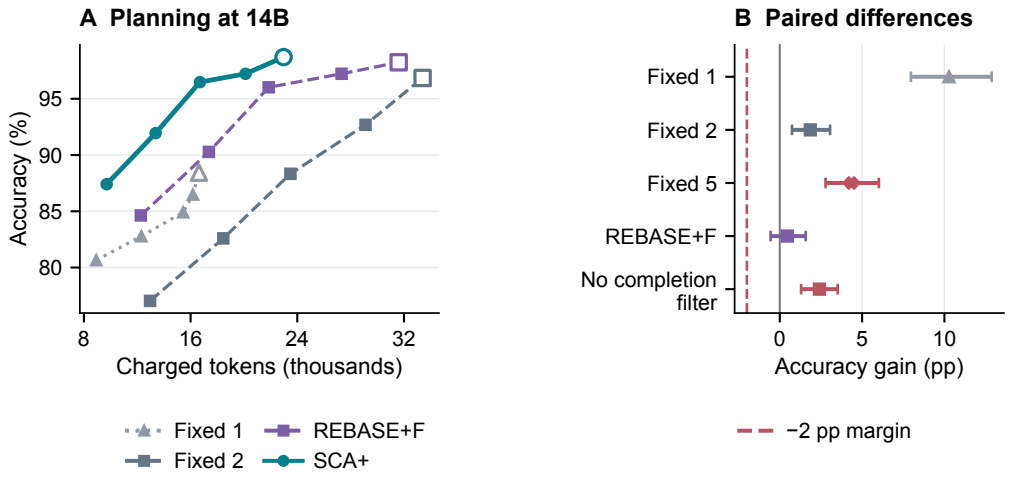


Figure 14: **E6: saturated accuracy changes the comparison to cost.** **A**: mean charged tokens versus accuracy; hollow points mark the 65,536-token ceiling. **B**: paired differences with 95% task-bootstrap intervals; the dashed line is the declared $-2$ pp non-inferiority margin. The E4 tasks with a 14B model and the same frozen controller.

**Cross-budget comparison.** SCA+ at 32,768 tokens against BoN at 65,536 tokens on the 480-problem panel: $+0.14$ pp $[-1.32, 1.67]$, 35 task wins, 36 losses and 409 ties, with 53.55% fewer tokens.

**Core panels.** On the 80-problem single-seed MATH core panel at 32,768 tokens, SCA+ solves 73, BG-stop 70, BG-MCTS 69, REBASE+F 67, BoN 66, strong vote 65, fixed $k$=2 60 and fixed $k$=5 47; at 65,536 tokens, 77, 71, 75, 74, 73, 71, respectively for the first six. Single-seed paired tests on 80 problems are underpowered (exact $p = 0.29$ against BG-MCTS at 32,768); the confirmation panel carries the inference.

**Where records end up.** At 32,768 tokens on the 1,440-run E5 panel: fixed $k$=5 leaves 373 runs without a usable answer, 108 with only wrong answers and 108 with a correct answer that was not selected; BoN leaves 0, 78 and 162; BG-MCTS 0, 37 and 141; SCA+ 2, 36 and 72. Table 13 gives the token composition behind these runs.

## J Token accounting and modeled query time

### J.1 Two token currencies

**Output-token accounting.** The BG-MCTS paper budgets generated output tokens only. Table 12 restates the E5 comparison in that currency. At 32,768 charged tokens SCA+ generates 9,731 output tokens and BG-MCTS 5,260; at 65,536, 16,088 against 8,855. Under output-only accounting BG-MCTS at the higher budget (94.86% with 8,855 output tokens) sits above SCA+ at the lower budget

(92.36% with 9,731), while under the total-token accounting used in this paper SCA+ leads at both budgets. The two accountings differ in whether verifier input, 15,289 tokens per run for BG-MCTS at 32,768, is charged.

Table 12: E5 confirmation panel restated in output tokens. Accuracy and mean charged tokens as in Tables 9 and 10; the last two columns show generated output and verifier input. Generation input is the remaining component, detailed in Table 13.

| Budget | Method | Accuracy (%) | Charged tokens | Output tokens | Verifier input |
|---|---|---|---|---|---|
| 32,768 | SCA+ | 92.36 | 25,666 | 9,731 | 7,187 |
| 32,768 | BG-MCTS | 87.64 | 30,578 | 5,260 | 15,289 |
| 32,768 | BG-stop | 85.21 | 22,725 | 4,337 | 11,363 |
| 32,768 | BoN | 83.33 | 27,755 | 10,405 | 7,766 |
| 32,768 | REBASE+F | 84.31 | 27,278 | 10,226 | 7,636 |
| 65,536 | SCA+ | 97.01 | 42,438 | 16,088 | 11,885 |
| 65,536 | BG-MCTS | 94.86 | 62,857 | 8,855 | 31,428 |
| 65,536 | BG-stop | 90.42 | 42,412 | 6,827 | 21,206 |
| 65,536 | BoN | 92.22 | 55,259 | 20,705 | 15,470 |
| 65,536 | REBASE+F | 93.33 | 49,282 | 18,463 | 13,803 |

Table 13: Mean token composition per run at 32,768 tokens, E5 confirmation panel.

| Method | Generation input | Generation output | Verifier input | Total |
|---|---|---|---|---|
| SCA+ | 8,748 | 9,731 | 7,187 | 25,666 |
| BG-MCTS | 10,029 | 5,260 | 15,289 | 30,578 |
| BG-stop | 7,025 | 4,337 | 11,363 | 22,725 |
| REBASE+F | 9,415 | 10,226 | 7,636 | 27,278 |
| BoN | 9,584 | 10,405 | 7,766 | 27,755 |
| Vote, no PRM | 13,278 | 14,413 | 0 | 27,691 |

**Where the charges differ.** Table 13 shows why output tokens alone reverse part of the comparison: BG-MCTS generates fewer tokens but spends more on verifier input. Total tokens treat each token as a logical charge regardless of kernel or caching efficiency. Neither this sum nor the output count is a direct measure of time, which motivates the separate service model below.

### J.2 From work items to modeled seconds

Table 14 translates the recorded work into modeled seconds under four service profiles. These are assumed execution costs; no hardware was timed. The following settings explain why their ordering differs from the token frontier.

**Service assumptions.** The profile assumes one 80GB-class GPU with both 7B models resident and one query admitted at a time. Generation prefill runs at 5,000 tokens/s, decoding at 80 tokens/s per sequence with context multiplier $1 + 0.035 \cdot \text{mean context}/4096$, and PRM scoring at 3,500 tokens/s. Concurrent decoding has multiplier $1+0.18(\text{active}-1)$. Four profiles vary capacity (serial or batch4), prefix caching (batch4 without cache), or PRM throughput (batch4 at half throughput).

**Execution and exclusions.** Within a query, BG-MCTS inherits parent prefixes, beam lanes reuse part of the preceding prefix, and sampling shares the prompt after its first wave. Caching changes modeled time while logical charges stay fixed. Generation precedes scoring, and decisions wait for the scores they require. Query time includes GPU service, CPU barriers and selection; loading, compilation, cross-query queueing and cold start are excluded. A deterministic lognormal factor per task, seed and budget ($\sigma = 0.035$, clipped to $[0.9, 1.1]$) is shared by methods and profiles.

Table 14: Modeled seconds per query at 32,768 tokens (E5). Top: service profiles. Bottom: batch4 components.

| Method | Serial | Batch4 | No cache | Slow PRM |
|---|---|---|---|---|
| SCA+ | 125.5 | 102.1 | 103.5 | 103.4 |
| BG-MCTS | 68.4 | 68.4 | 73.2 | 70.0 |
| BG-stop | 56.4 | 56.4 | 59.7 | 57.7 |
| REBASE+F | 131.8 | 80.0 | 81.5 | 81.4 |
| BoN | 135.4 | 65.4 | 65.9 | 67.6 |
| Vote, no PRM | 185.1 | 88.0 | 88.6 | 88.0 |

| Method | Prefill | Decode | PRM | Dispatch | Control |
|---|---|---|---|---|---|
| SCA+ | 0.99 | 99.44 | 1.35 | 0.21 | 0.08 |
| BG-MCTS | 0.06 | 66.38 | 1.59 | 0.31 | 0.08 |
| BG-stop | 0.06 | 54.68 | 1.32 | 0.26 | 0.06 |
| REBASE+F | 1.06 | 77.18 | 1.44 | 0.23 | 0.04 |
| BoN | 1.48 | 61.63 | 2.23 | 0.06 | 0.01 |
| Vote, no PRM | 2.05 | 85.87 | 0.00 | 0.04 | 0.01 |

**Reading the comparison.** Figure 15 changes only the cost axis: total tokens in A, serial time in B and batch4 time in C. Output decoding dominates the assumed profile, while verifier input is relatively cheap. Concurrent service benefits sampling more than the often-narrow SCA+ controller. Figure 16 shows how one query's work is scheduled.

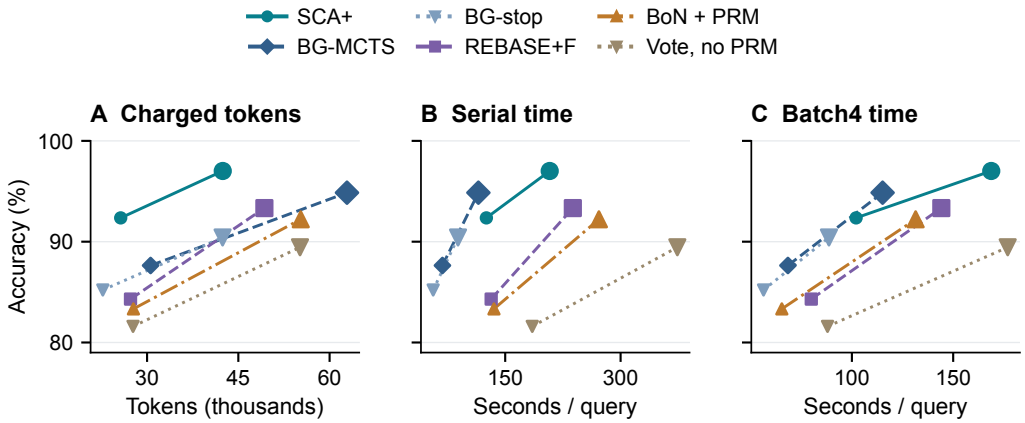


Figure 15: **Charged tokens and modeled query time on the MATH confirmation panel.** Small and large markers are the 32,768 and 65,536 ceilings. **A**: accuracy against mean charged model tokens. **B**: accuracy against modeled query time when one work item is served at a time under the assumed profile. **C**: the same with up to four concurrent work items.

**The schedule explains the profile dependence.** Figure 16 displays a single query; the table reports means. Concurrent sampling can occupy several decode lanes, whereas narrow search repeatedly waits for the scores needed to choose its next expansion. The same completed-answer objective can therefore favor different methods under a token ceiling and a latency ceiling. A deployment claim would need a measured service profile matched to the intended workload.

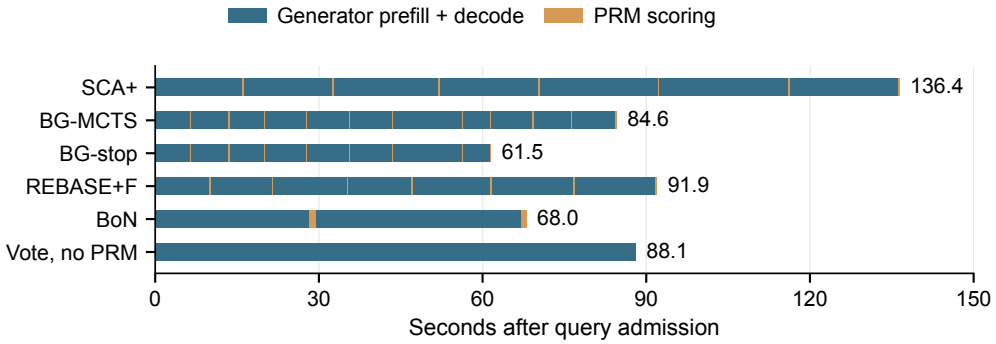


Figure 16: **One paired query under the batch4 profile.** Device intervals for MATH problem 0066, seed 0, 32,768 tokens; generator prefill and decode in blue, PRM scoring in orange. Endpoint labels give the modeled query times reported with the trace.