WHEN DOES LONGER REASONING HELP? PREDICTING MATHEMATICAL REASONING THROUGH DISCOVERY AND EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time compute can improve mathematical reasoning, but can short-budget runs predict how mathematical reasoning scales with additional compute? We introduce a *Discovery–Execution (DE)* framework that predicts the aggregate held-out scaling curves through a convolution of strategy *discovery* and conditional *execution*. From independent short-budget attempts and oracle-sketch-conditioned runs, the framework estimates cumulative success along held-out reasoning trajectories under alternate compute allocations. We evaluate four models on 35 fresh Olympiad problems and non-geometry problems from IMO-ProofBench Advanced. Under the DE framework, near-saturated execution predicts geometric scaling, as observed for the GPT models. For Claude Opus 4.8, incorporating measured execution substantially improves held-out forecasts over geometric extrapolation across one- and two-arm allocations. As a secondary application, regularized DE (R-DE) decisions to continue or restart yield lower average regret than the best model-specific retrospective policy. Together, these results show that measuring conditional execution provides information about longer reasoning that short-budget success rates do not always capture.

1 INTRODUCTION

Additional compute can improve mathematical reasoning through longer attempts or repeated sampling. Prior work finds benefits from both approaches, with their relative effectiveness varying across models, problems, and inference procedures (Snell et al., 2025; Wu et al., 2025a; Hasan et al., 2026). Long reasoning trajectories can complete computations that short attempts cannot (Mirtaheri et al., 2025), while broader search exposes alternatives missed by one trajectory (Wang et al., 2023; 2025). Can separate measurements predict aggregate scaling under these allocations without observing their outcomes?

We study this question through *strategy discovery* and *conditional execution*. A strategy specifies a proof's load-bearing constructions, claims, and dependencies; execution expands it into a valid proof. A solver could be bottlenecked by either, so two solvers with identical success rates may respond differently to additional compute. We show that short-budget runs and sketch-conditioned completion measurements can predict held-out scaling curves under different compute allocations. We call this framework *Discovery–Execution (DE)*.

In this paper, we approximate the time to strategy acquisition with a geometric distribution, but this is not a requirement for the general DE framework. Under this assumption, when execution is saturated, DE reduces to the *Simple Geometric (SG)* success curve. When execution improves with additional compute, the framework instead predicts different scaling curves and allocation tradeoffs. The prediction of longer reasoning outcomes therefore depends on both strategy acquisition and conditional execution, and cannot always be inferred from short-budget success probability alone.

To test our framework, we introduce **AOBench (Advanced Olympiad Benchmark)** – 35 hard, non-geometry problems from 2026 olympiads with human-checked reference proofs, solution sketches of at most 25 words, and explicit load-bearing steps. AOBench is designed to be comparable in size and difficulty to the 30-problem IMO-ProofBench Advanced benchmark (Luong et al., 2025). We replicate our study on all 22 non-geometry problems from that benchmark as well. We exclude

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

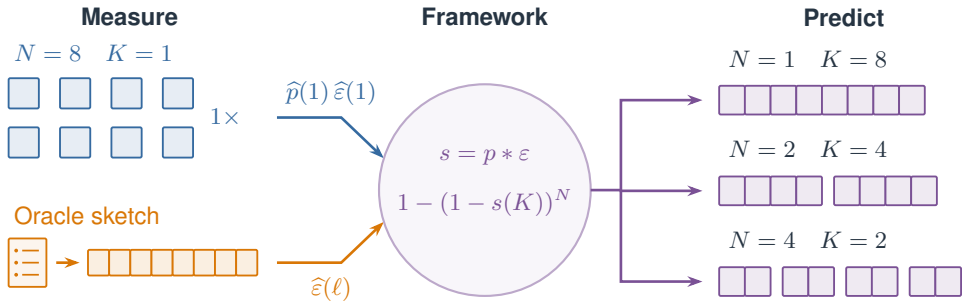


Figure 1: **Predicting the value of longer reasoning.** Short budget runs measure success within one block; runs supplied with an oracle strategy sketch measure execution. Together, these measurements estimate discovery and execution, whose convolution predicts held-out unaided success across compute allocations. Each tile is one block (200k output tokens); N independent attempts receive K blocks each, with $NK = 8$.

geometry problems because models can reduce these problems to extensive algebraic calculations whose size makes human audit impractical.

Figure 1 summarizes the measurement and prediction setup. A *Parallel-N* compute *allocation* divides a *budget* of $B = 8$ compute blocks equally among N independent *attempts*. The *run* succeeds if any attempt produces a correct solution. An *Oracle-Execution* trajectory initializes an attempt with a verified strategy sketch. We measure Parallel-8 and Oracle-Execution success rates to estimate discovery and execution without observing other allocations, and evaluate on Parallel-*N* trajectories for $N = 1, 2, 4$ (Figure 2). The model-generated proofs are graded for mathematical validity by GPT-5.6 Sol. A stratified sample of 50 proofs was also graded by an International Mathematical Olympiad (IMO) medalist panel to assess agreement with the automated audits.

Our results show when SG predicts aggregate scaling well and when measuring execution improves its predictions. We observe that the GPT models approach execution saturation early, so the framework predicts scaling close to SG, matching the observed curves. In contrast, Claude Opus benefits from additional compute during execution. Incorporating execution with R-DE reduces held-out prediction RMSE from 7.16 to 2.87 solved trials at $N = 1$ and from 9.16 to 2.96 at $N = 2$. A joint target-trial bootstrap supports the improvement across these allocations (Appendix D.4). Even within Opus, the same contrast holds: SG predictions fit well on problems with saturated first-block execution, while incorporating execution improves predictions on the unsaturated subset (Table 2).

As a secondary application, these measurements inform whether to extend an unfinished trajectory or start over. R-DE achieves modestly lower average measured regret than each model's retrospectively selected best fixed action, with the largest average gains at early budgets (Figure 3).

In summary, our contributions are:

- A predictive discovery–execution framework that expresses test-time mathematical reasoning as a convolution of strategy discovery and execution.
- Predictions of aggregate unaided scaling from separate measurements, testing when execution adds predictive information, with continue-or-restart decisions as an application.
- AOBench: 35 hard Olympiad problems with human-checked proofs and verified strategy sketches, evaluated alongside IMO-ProofBench Advanced across four models.

2 Related Work

Forecasting inference scaling. Repeated independent sampling can substantially increase problem coverage (Brown et al., 2024). Schaeffer et al. (2025) explain how heterogeneity in per-problem success probabilities connects exponential per-problem scaling to aggregate power laws and enables forecasting. Kazdan et al. (2025) develop a beta-binomial approach to predicting pass@k from limited samples. These works forecast repeated-sampling performance from the distribution of single-

attempt success probabilities. We ask when short-attempt success also predicts sequentially extended reasoning, and whether separately measured conditional execution adds predictive information.

Depth versus breadth. Test-time scaling can deepen one trajectory through self-critique, revision, or budget forcing (Madaan et al., 2023; Muennighoff et al., 2025), or broaden search through self-consistency and verifier-selected best-of-N (Wang et al., 2023; Snell et al., 2025; Wu et al., 2025a). Long chains can dominate parallel short chains in constructed settings (Mirtaheri et al., 2025), yet the compute-optimal allocation varies with difficulty (Snell et al., 2025; Wu et al., 2025a); learned exploration can extend scaling (Setlur et al., 2025), while continued reasoning can plateau or regress (Ghosal et al., 2025). Gu et al. (2026) find evidence that reduced exploration contributes to sequential sampling's disadvantage relative to parallel sampling in their settings. Reset-and-Discard reallocates repeated attempts across problems to improve coverage under a shared budget (Meir et al., 2026); our continue-or-restart application instead compares extending and restarting an attempt on the same problem.

Strategy generation and execution. PlanSearch diversifies natural-language plans before implementation and analyzes success conditioned on an idea, including benefits from short sketches (Wang et al., 2025). TTS-Uniform distributes inference across explicitly extracted strategies to mitigate selection bias (Wu et al., 2025b). In mathematical reasoning, Liang et al. (2026) distinguish strategy occurrence from model-relative executability, while Liang et al. (2025) separate high-level reasoning from formal proof generation. Targeted guidance can also unlock unaided failures in mathematics (Agrawal et al., 2024) and olympiad programming (Shi et al., 2024). In our work, oracle guidance was only used as an intervention to estimate compute-dependent execution, and then predict unaided allocations whose outcomes are not used for estimation.

Olympiad proofs and evaluation. Huang & Yang (2025) study a model-agnostic verification-and-refinement pipeline for IMO proofs. Luong et al. (2025) introduce IMO-Bench, including IMO-ProofBench for proof generation and IMO-GradingBench for evaluating proof grading. We use the non-geometry IMO-ProofBench Advanced problems alongside AOBench to evaluate predictions of unaided proof success as compute increases.

3 Frameworks for Test-Time Scaling

Let us fix a problem, model, and inference procedure, and let $B \in \mathbb{N}$ denote the available test-time compute budget, measured in discrete compute blocks (fixed token budgets in our experiments). We suppress the problem index in this section; a subscript n elsewhere identifies problem-specific quantities.

3.1 Simple Geometric (SG) Framework

Let q denote the probability that a fresh attempt solves the problem within one compute block. A simple extrapolation assumes that each additional block has success probability q, conditional on the trajectory remaining unsolved. The resulting success curve after K blocks is geometric:

$$s^{\mathrm{SG}}(K) = 1 - (1 - q)^K. \tag{1}$$

Equation 1 treats every new block as a fresh chance of success. However, an unfinished trajectory may already contain a viable strategy and need more compute to expand it into a rigorous proof. The SG framework does not account for this progress, so its predictions may be miscalibrated when the compute requirements increase. This motivates us to separate strategy acquisition from proof completion.

3.2 Discovery–Execution (DE) Framework

We model proof completion as strategy acquisition followed by execution. Let A denote the block in which a viable strategy is first acquired, and let E denote the number of blocks from acquisition to proof completion, counting the acquisition block as the first. We assume that, conditional on strategy acquisition, the execution does not depend on when the strategy was acquired. **(Assumption 1)**

Let

$$p(k) = \Pr(A = k), \qquad \varepsilon(\ell) = \Pr(E \le \ell \mid A = k),$$

denote the first-discovery distribution and cumulative execution probability, respectively. By Assumption 1, the conditional probability defining $\varepsilon(\ell)$ is the same for every k with $p(k) > 0$. When multiple viable strategies exist, $\varepsilon(\ell)$ represents the discovery-weighted average of their execution probabilities. A strategy acquired in block k has $K - k + 1$ blocks available for completion by block K. Therefore, the probability of solving within K blocks is

$$s(K) = (p * \varepsilon)(K) = \sum_{k=1}^{K} \underbrace{p(k)}_{\text{discovery}} \underbrace{\varepsilon(K - k + 1)}_{\text{execution}}. \tag{2}$$

The unaided success curve is thus a discrete convolution of strategy discovery and conditional execution. If an oracle strategy is provided in the initial prompt, then $p^{\text{oracle}}(1) = 1$ and $p^{\text{oracle}}(k) = 0$ for $k > 1$. The success curve therefore measures conditional oracle execution:

$$s^{\text{oracle}}(K) = \varepsilon^{\text{oracle}}(K). \tag{3}$$

Assuming $\varepsilon^{\text{oracle}}(\ell)$ approximates $\varepsilon(\ell)$ following natural strategy acquisition, we may use the oracle-execution measurement in the convolution to predict unaided success. **(Assumption 2)**

3.3 Geometric Distribution for Discovery

We now simplify the DE framework by assuming that *strategy acquisition* is memoryless, with probability $\alpha \in (0, 1]$ of first viable strategy acquisition. **(Assumption 3)** We do not impose a geometric form on execution. Under this geometric acquisition assumption,

$$p(k) = \alpha(1 - \alpha)^{k-1}, \qquad s(K) = \sum_{k=1}^{K} \alpha(1 - \alpha)^{k-1} \varepsilon(K - k + 1). \tag{4}$$

At the first block, fresh proof success has probability $q = s(1) = \alpha \cdot \varepsilon(1)$, which need not equal the acquisition probability. Also, q alone cannot determine the longer-trajectory curve under DE since the execution term also influences it.

We examine the robustness of assumptions 1-3 in Appendix E.

3.4 Optimal Compute Allocation

For a total budget $B \in \mathbb{N}$, assigning K blocks to each arm permits $N = \lfloor B/K \rfloor$ independent attempts. Among these equal-depth independent allocations, the optimal depth $K^\star$ maximizes the probability that at least one attempt solves the problem, conditional on the problem parameters:

$$S_B(K) = 1 - (1 - s(K))^{\lfloor B/K \rfloor}, \qquad K^\star \in \operatorname*{arg\,max}_{K \in \{1, \ldots, B\}} S_B(K). \tag{5}$$

Proposition 1 (saturated execution). Under geometric discovery, suppose execution is saturated within one block, $\varepsilon(1) = 1$. Then $q = \alpha$ and $s(K) = 1 - (1 - q)^K$. Consequently, all allocations of N independent attempts of K blocks each that use the full budget $NK = B$ have the same success probability:

$$1 - (1 - s(K))^N = 1 - (1 - q)^{NK} = 1 - (1 - q)^B. \tag{6}$$

Thus, the SG framework is a special case of DE under geometric discovery and saturated first-block execution. When execution is unsaturated, additional compute can also help complete an acquired strategy. The following proposition identifies a regime in which this favors longer trajectories.

Proposition 2 (unsaturated execution). Suppose $\varepsilon(1) < \varepsilon(\ell)$ for some $\ell \in \{2, \ldots, B\}$. Under geometric discovery, as $\alpha \to 0$, Eq. 5 is uniquely maximized to first order in α by $K^\star = B$ and $N^\star = 1$.

In the rare-discovery limit, splitting a budget provides no first-order increase in discovery opportunities. A long trajectory leaves more compute to execute an early discovery. Proofs are provided in Appendix B.

3.5 Parameter Estimation

We estimate fresh success $\widehat{q}$ from Parallel-8 and conditional completion $\widehat{\varepsilon}^{\text{oracle}}(k)$ from Oracle-Execution. Section 4 describes the measurement runs.

Discovery–Execution (DE). Under assumption 2, $\varepsilon^{\text{oracle}}(k) \approx \varepsilon(k)$ for $1 \le k \le B$, we estimate discovery by correcting fresh success for first-block oracle completion:

$$\widehat{\alpha} = \min\left\{1, \frac{\widehat{q}}{\widehat{\varepsilon}^{\text{oracle}}(1)}\right\}, \qquad \widehat{\varepsilon}^{\text{oracle}}(1) > 0. \tag{7}$$

When the empirical fresh rate exceeds first-block oracle completion, the constrained fit sets $\widehat{\alpha} = 1$ and jointly adjusts the execution estimate.

Regularized Discovery–Execution (R-DE). For $1 \le k \le B$, let $\pi^{(k)}$ denote the probability of first oracle completion in block k, and let $\pi^{(B+1)}$ denote the probability of no completion by block B. Then,

$$\varepsilon(\ell) = \sum_{k=1}^{\ell} \pi^{(k)}, \quad 1 \le \ell \le B$$

Let $\boldsymbol{\pi} = \left(\pi^{(1)}, \ldots, \pi^{(B+1)}\right)$. To stabilize estimates from few trials, we use the priors

$$\alpha \sim \text{Beta}(a, b), \qquad \boldsymbol{\pi} \sim \text{Dirichlet}(\boldsymbol{d}),$$

with shared parameters $(a, b, \boldsymbol{d})$ fitted across all 57 problems separately for each LLM, using only Parallel-8 and Oracle-Execution data. More details about parameter estimation are provided in Appendix C.

3.6 Continue or Restart

Proposition 1 shows that Simple Geometric assigns the same success probability to continuing an unfinished attempt for h more blocks or starting a fresh h-block attempt. Both DE and R-DE can distinguish these actions through the measured execution curve.

For a given problem, continuation contributes an unconditional success increment $s(k+h) - s(k)$, whereas restarting unfinished trajectories contributes $(1 - s(k))s(h)$. We compare the expected unconditional gains:

$$\Delta^{(k)}(h) = \mathbb{E}[s(k+h) - s(k) - (1 - s(k))s(h)]\,. \tag{8}$$

A positive expected advantage favors continuation, giving the per-problem policy:

$$\rho^{(k)}(h) = \begin{cases} \text{continue}, & \Delta^{(k)}(h) > 0, \\ \text{restart}, & \text{otherwise.} \end{cases}$$

4 Experiments

4.1 Datasets and Models

Datasets. AOBench contains 35 non-geometry problems from twelve olympiads and selection tests held in 2026: 9 algebra, 13 combinatorics, and 13 number theory. The set was fixed before model evaluation, and every problem has a human-audited reference proof, a verified strategy sketch, and three explicit load-bearing steps. We repeat the study on all 22 non-geometry problems from IMO-ProofBench Advanced: 8 algebra, 8 combinatorics, and 6 number theory (Luong et al., 2025). Appendix A provides details and the AOBench subtopic breakdown.

Models. We evaluate Muse Spark 1.2, GPT-5.4, GPT-5.5, and Claude Opus 4.8. We report results separately for each model. All models use the same harness and tools, with requests routed to their respective provider APIs.

4.2 HARNESS AND COMPUTE

Each attempt uses an isolated Docker container and model session. Solvers receive the problem and condition-specific prompt, can execute local code, and cannot access reference solutions, other runs, web tools, or other agents. Networking is restricted to provider endpoints (Appendix A.2).

A compute block ($1\times$) is a budget of 200k provider-reported output tokens, including reasoning and generated tool calls. Sequential trajectories alternate critique and revision in one persistent Self-Refine session (Madaan et al., 2023), up to $8\times = 1.6$M tokens. We evaluate the last complete proof at each cumulative block boundary. Early termination is allowed after a minimum of ten rounds and if two consecutive self-critiques find no gap in the generated proof.

4.3 MEASUREMENTS

Parallel-N ($N \in \{1, 2, 4, 8\}$). The total budget is $B = 8$ blocks. Parallel-N allocates N independent attempts, each running a fresh trajectory for $K = B/N$ blocks.

Oracle-Execution ($N = 1, K = 8$). An eight-block trajectory receives a verified sketch of one reference strategy, limited to 25 words.

We use three runs of **Parallel-8 and Oracle-Execution measurements** to estimate our parameters q and ε. From Parallel-8, three runs of 8 fresh attempts give 24 one-block proof outcomes per problem. Their solve fraction gives the fresh success estimate $\widehat{q}$. From Oracle-Execution, the fraction solved by block $1 \le k \le B$ estimates $\varepsilon^{\mathrm{oracle}}(k)$.

Parallel-N runs with $N \neq 8$ are **unaided target runs**, and their outcomes are never used to fit our framework parameters. We evaluate $N = 1, 2$ for all four models and $N = 4$ for Muse and GPT-5.5.

Continue or Restart. At checkpoints $k = 1, \ldots, 7$, we allocate one additional block ($h = 1$) to continuation if R-DE's predicted $\Delta_n^{(k)}(1) > 0$ in Equation 8, and to restart otherwise. We evaluate regret (Equation 9) using observed continuations and estimated restart outcomes. Restart outcomes are estimated from six Parallel-2 trajectories per problem whose outcomes are not used to fit any predictor.

4.4 ALTERNATIVE PREDICTORS

We define three alternative predictors as follows. All predictors use only measurement-run outcomes.

SG. The saturated-execution special case predicts scaling from fresh success alone. For u solved attempts among $m = 24$ Parallel-8 attempts (3 seeds for each run), the finite-bank estimate is

$$\widehat{s}^{\mathrm{SG}}(K) = 1 - \frac{\binom{m-u}{K}}{\binom{m}{K}}.$$

The numerator is zero when $m - u < K$.

R-SG. To test whether regularization alone explains the gains, we place a shared $\mathrm{Beta}(a, b)$ prior on fresh success q. For each model, we fit a, b by maximizing the beta-binomial marginal likelihood of its 57 problems' Parallel-8 counts.

SCT. Sketch Curve Transfer adds the sketch-conditioned curve's improvement beyond the first block to fresh success, scaled by the initially unsolved fraction.

$$\widehat{s}^{\mathrm{SCT}}(K) = \widehat{q} + (1 - \widehat{q})\left[\widehat{\varepsilon}^{\mathrm{oracle}}(K) - \widehat{\varepsilon}^{\mathrm{oracle}}(1)\right].$$

SCT tests whether the unaided curve can be predicted by translating the sketch-conditioned curve to fresh success.

4.5 Evaluation

Model-generated proofs are graded by GPT-5.6 Sol using a near-binary rubric with only four allowed scores: 7 for a complete, rigorous proof; 6 or 5 for an otherwise correct proof with exactly one or two minor local defects, respectively; and 0 otherwise. Scores 5–7 count as solved. An IMO medalist panel independently audited 50 distinct proofs and agreed with automated judgments on solved status in 48/50 cases (96.0%) and reference-step presence in 140/150 cases (93.3%). Appendix F gives the protocol and detailed results.

Our primary target is the aggregate cumulative solved-trial count across problems at each compute checkpoint. We report RMSE between predicted and observed counts, with 171 trials per complete curve (lower is better). The Average row reports the square root of the mean model-specific squared RMSE. Execution-subset comparisons use the same calculation on each subset.

For continue-or-restart decisions, let C and R be the observed continuation and estimated restart counts of additional solved trials, and let $V(\rho^{(k)}(h))$ be the count under the selected action. Measured regret is

$$g = \max\{C, R\} - V\left(\rho^{(k)}(h)\right). \tag{9}$$

We sum regret across problems and average over checkpoints and then equally over models. We compare R-DE with always continuing, always restarting, and best fixed, which retrospectively selects each model's better constant action. Lower regret means fewer additional successes lost relative to the better observed action on each problem.

5 Results

5.1 When does execution matter?

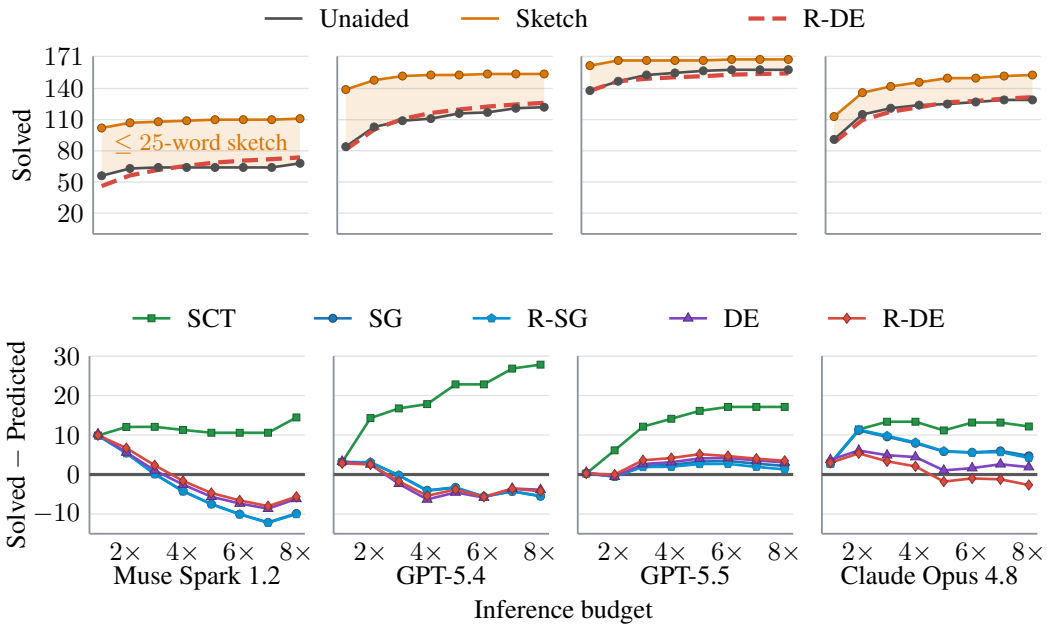


Figure 2: **Measured execution and held-out Parallel-1 scaling** ($N = 1$)**.** Top: sketch-conditioned and unaided successes, with dashed R-DE predictions. Bottom: difference between observed and predicted solved counts. Each condition has 171 trajectories across 57 problems (3 seeds each).

The predictive value of measuring execution varies across models (Figure 2). GPT-5.5 completes 162/171 oracle trajectories within one block and 168/171 within eight. With little further execution

Table 1: Prediction RMSE (solved-trial counts) across 57 problems and 171 trials per model. Bold marks the lowest error within each row and allocation.

	$N = 1$					$N = 2$				
Model	SG	R-SG	SCT	DE	R-DE	SG	R-SG	SCT	DE	R-DE
Muse	8.26	8.31	11.52	6.54	**6.25**	5.61	**5.58**	10.49	6.26	**5.58**
GPT-5.4	3.96	3.98	20.46	4.25	**3.93**	2.83	2.83	9.97	2.32	**2.16**
GPT-5.5	2.40	**1.86**	13.80	3.04	3.66	3.59	**3.03**	6.47	3.88	3.46
Opus	7.16	7.20	11.81	3.68	**2.87**	9.16	9.10	5.48	6.70	**2.96**
Average	5.94	5.92	14.84	4.57	**4.36**	5.84	5.73	8.39	5.11	**3.76**

gain (Proposition 1), SG closely predicts its unaided scaling (RMSE 2.40), as well as GPT-5.4's (Table 1).

Opus's oracle successes rise from 113/171 to 153/171 over eight blocks. Incorporating execution improves its held-out forecasts, reducing SG's RMSE of 7.16 to 3.68 under DE and 2.87 under R-DE. R-SG gives 7.20, so regularization alone does not explain the improvement. Flattening the measured execution curve also worsens Opus's predictions (Appendix E.6).

Table 2: Opus prediction RMSE (solved-trial counts), split by first-block execution saturation. Bold marks row minima. DE helps when execution is not saturated.

N	Execution	Trials	SG	DE
1	Saturated	87	**2.40**	2.65
	Unsaturated	84	5.50	**1.86**
2	Saturated	87	**2.17**	2.50
	Unsaturated	84	8.25	**5.22**

Within Opus, SG predicts well when all three oracle trajectories solve in one block, and DE offers no improvement (Table 2). On the remaining 28 problems, DE reduces RMSE from 5.50 to 1.86 at $N = 1$ and from 8.25 to 5.22 at $N = 2$.

5.2 Predicting across alternate compute allocations

At $N = 2$, R-DE has the lowest across-model RMSE: 3.76 versus 5.84 for SG and 5.73 for R-SG (Table 1). For Opus, R-DE reduces RMSE from SG's 9.16 to 2.96, while R-SG remains best for GPT-5.5.

Combining $N = 1$ and $N = 2$, Opus's RMSE falls from 8.22 under SG and 8.20 under R-SG to 2.91 under R-DE. Both reductions have 95% paired target-bootstrap percentile intervals above zero (Appendix D.4).

For Muse, R-DE closely predicts final $N = 2$ success (77.3 predicted versus 77 observed), with the largest gap at the first checkpoint (Appendix D).

Transfer without test-problem sketches. Across 50 splits, we learn shared priors from 38 problems and predict individual four-block unaided trajectories on 19 held-out problems using only their Parallel-8 outcomes. For Opus, R-DE reduces mean solved-count RMSE from R-SG's 6.24 to 5.04, or 3.40 with test-problem oracle measurements. Muse also improves, GPT-5.4 is nearly tied, and GPT-5.5 favors R-SG (Appendix E.7).

5.3 Choosing whether to continue or restart

Across seven checkpoints, R-DE's mean measured next-block regret is 1.50, versus 1.79 for each model's retrospectively selected best fixed action, 4.30 for always continuing, and 2.13 for always restarting (Table 3). R-DE modestly improves on best fixed for both GPT models and Opus, but not Muse.

Table 3: Mean next-block regret (solved trials lost), $k = 1, \ldots, 7$. Best fixed selects each model's better constant action retrospectively. Bold marks row minima.

Model	Always continue	Always restart	Best fixed	R-DE
Muse Spark 1.2	9.74	**0.57**	**0.57**	1.57
GPT-5.4	3.14	3.62	3.14	**2.29**
GPT-5.5	2.10	1.19	1.19	**0.76**
Claude Opus 4.8	2.24	3.14	2.24	**1.38**
Average	4.30	2.13	1.79	**1.50**

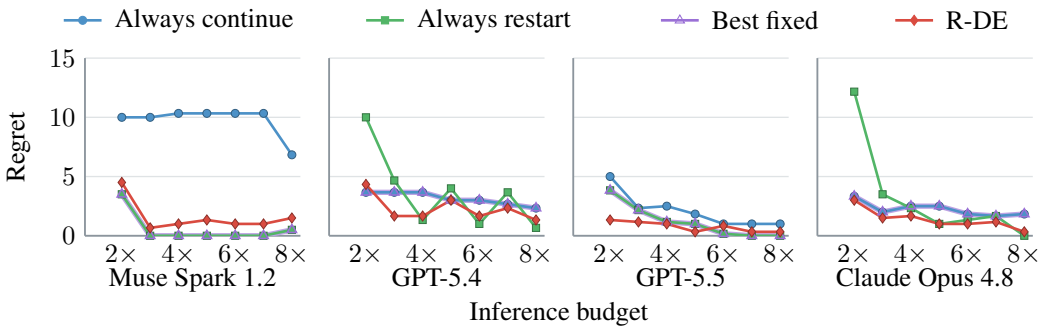


Figure 3: **Next-block regret across inference budgets.** Solved trials lost; lower is better. Budgets include the additional block.

R-DE's largest average gains over best fixed occur at $3\times$ and $4\times$ (Figure 3; Appendix D.5), avoiding much of the initial restart cost for Opus and GPT-5.4.

6 Discussion and Limitations

R-DE's aggregate forecasts show that conditional execution adds predictive information beyond short-budget success. SG remains competitive near first-block saturation, whereas Opus benefits from modeling completion over longer budgets. Gains vary across models and problems (Appendix D.3). Muse's sketch-timing sensitivity and preference for reduced late discovery reveal limits of the assumptions (Appendix E). DE provides a predictive decomposition rather than an account of internal reasoning.

Main forecasts use problem-specific oracle sketches, but execution information also transfers across problems (Appendix E.7). The 57-problem sample and three oracle trajectories per problem limit precision, particularly for individual execution curves and small policy differences.

7 Conclusion

We introduced a Discovery–Execution framework that models mathematical proof success as a convolution of strategy discovery and conditional execution. Measurements from short unaided attempts and Oracle-Execution trajectories predict aggregate scaling under held-out compute allocations without fitting the target outcomes. When execution is certain, the framework recovers geometric scaling, while incorporating compute-dependent execution improves beyond geometric extrapolation. The same measurements inform continue-or-restart decisions, yielding lower average regret than each model's best fixed action. Together, these results show that short-budget success alone does not determine the value of longer reasoning: measuring conditional execution provides additional information for predicting scaling and informing compute allocation.

AI USE STATEMENT

OpenAI Codex was used to propose and refine hypotheses, critique experimental design, assist with the formulation and analysis of the mathematical framework and arguments, implement and debug the experimental harness and analysis code, analyze and interpret results, survey related literature, and prepare the manuscript. Language models also serve as solver and grading instruments in the experiments, as described in the paper. The authors reviewed all AI-assisted work and take responsibility for the final text, claims, code, and artifacts.

REPRODUCIBILITY STATEMENT

We will publicly release the experimental harness, audit and framework-fitting code, and AOBench, including its reference proofs, strategy sketches, and load-bearing steps. Section 4 describes the models, compute budgets, and evaluation protocol. Appendices A, B, C, and F provide dataset details, theoretical proofs, estimation procedures, and audit protocols, respectively.

REFERENCES

Vansh Agrawal, Pratham Singla, Amitoj Singh Miglani, Shivank Garg, and Ayush Mangal. Give me a hint: Can LLMs take a hint to solve math problems? *arXiv preprint arXiv:2410.05915*, 2024. URL https://arxiv.org/abs/2410.05915.

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. URL https://arxiv.org/abs/2407.21787.

Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. Does thinking more always help? mirage of test-time scaling in reasoning models. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/fc067ac218430c409d6f65403328f740-Abstract-Conference.html.

Xiangming Gu, Soham De, Larisa Markeeva, Petar Veličković, and Razvan Pascanu. Understanding performance gap between parallel and sequential sampling in large reasoning models. *arXiv preprint arXiv:2604.05868*, 2026. URL https://arxiv.org/abs/2604.05868.

Adib Hasan, Daniel Schaffield, Akashnil Dutta, and Tarik Adnan Moon. AutoFyn technical report: Non-parametric expert iteration for long-horizon agents, 2026. URL https://arxiv.org/abs/2609.05446.

Yichen Huang and Lin F. Yang. Winning gold at IMO 2025 with a model-agnostic verification-and-refinement pipeline. In *MATH-AI: The 5th Workshop on Mathematical Reasoning and AI at NeurIPS*, 2025. URL https://openreview.net/forum?id=svz4xDjRC1.

Joshua Kazdan, Rylan Schaeffer, Youssef Allouah, Colin Sullivan, Kyssen Yu, Noam Levi, and Sanmi Koyejo. Efficient prediction of Pass@k scaling in large language models. *arXiv preprint arXiv:2510.05197*, 2025. URL https://arxiv.org/abs/2510.05197.

Weida Liang, Yiyou Sun, Shuyuan Nan, Chuang Li, Dawn Song, and Kenji Kawaguchi. Strategy executability in mathematical reasoning: Leveraging human–model differences for effective guidance. *arXiv preprint arXiv:2602.22583*, 2026. URL https://arxiv.org/abs/2602.22583.

Zhenwen Liang, Linfeng Song, Yang Li, Tao Yang, Feng Zhang, Haitao Mi, and Dong Yu. Decoupling reasoning from proving: A new framework for tackling olympiad-level mathematics. In *MATH-AI: The 5th Workshop on Mathematical Reasoning and AI at NeurIPS*, 2025. URL https://openreview.net/forum?id=5Oc8TKWFWO.

Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Huiyi Hu, Henryk

Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu Hoang Trinh, Quoc V Le, and Junehyuk Jung. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35418–35442. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1794. URL https://aclanthology.org/2025.emnlp-main.1794/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2303.17651.

Sagi Meir, Tommer D. Keidar, Noam Levi, Shlomi Reuveni, and Barak Hirshberg. More bang for the buck: Improving the inference of large language models at a fixed budget using reset and discard (ReD). *arXiv preprint arXiv:2601.21522*, 2026. URL https://arxiv.org/abs/2601.21522.

Parsa Mirtaheri, Ezra Edelman, Samy Jelassi, Eran Malach, and Enric Boix-Adserà. Let me think! a long chain-of-thought can be worth exponentially many short ones. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2505.21825.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Rylan Schaeffer, Joshua Kazdan, John Hughes, Jordan Juravsky, Sara Price, Aengus Lynch, Erik Jones, Robert Kirk, Azalia Mirhoseini, and Sanmi Koyejo. How do large language monkeys get their power (Laws)? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 53132–53176. PMLR, 2025. URL https://proceedings.mlr.press/v267/schaeffer25a.html.

Amrith Setlur, Matthew Y. R. Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for LLMs. *arXiv preprint arXiv:2506.09026*, 2025. URL https://arxiv.org/abs/2506.09026.

Quan Shi, Michael Tang, Karthik Narasimhan, and Shunyu Yao. Can language models solve olympiad programming? In *Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2404.10952.

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. URL https://arxiv.org/abs/2408.03314.

Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. Planning in natural language improves LLM search for code generation. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.03733.

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

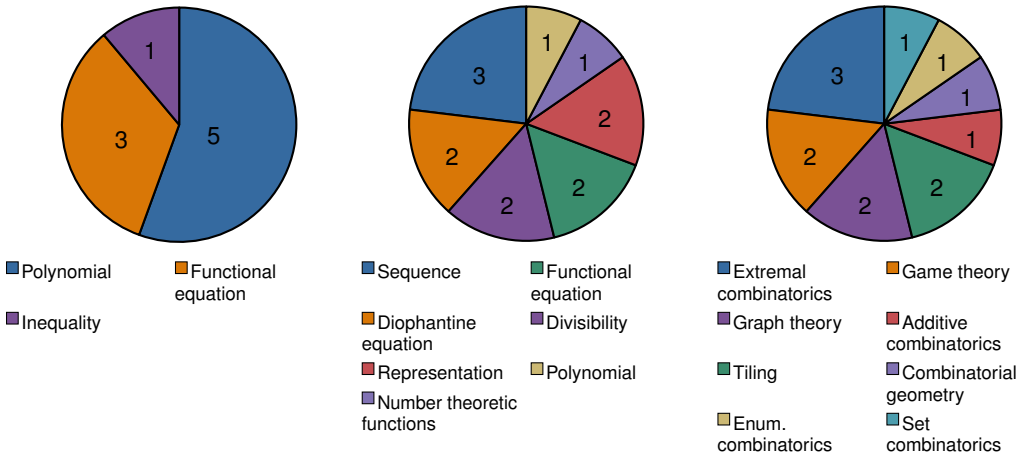
Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *International Conference on Learning Representations*, 2025a. URL https://arxiv.org/abs/2408.00724.

Zongqian Wu, Baoduo Xu, Tianyu Li, Zhu Sun, Xiaofeng Zhu, and Lei Feng. Mitigating strategy-selection bias in reasoning for more effective test-time scaling. *arXiv preprint arXiv:2509.17905*, 2025b. URL https://arxiv.org/abs/2509.17905.

A Dataset and protocol details

A.1 Datasets

AOBench was fixed before model evaluation and contains 35 non-geometry problems from competitions held in 2026. The sources are the China TST (12 problems), Iran TST (5), All-Russian MO (3), IMO (3), Korea FKMO (2), Romanian Master of Mathematics (2), Romania TST (2), Serbian MO (2), APMO (1), ELMO Shortlist (1), Serbia TST (1), and USAMO (1). Each record retains the original source and a complete checked reference solution. The problems were selected by the IMO medalist panel to be comparable in difficulty and size to the IMO-ProofBench Advanced dataset. No problem was removed based on model performance. Geometry was excluded before evaluation because the extensive algebraic calculations produced by models make human audit impractical.



(a) Algebra ($n = 9$) (b) Number theory ($n = 13$) (c) Combinatorics ($n = 13$)

Figure 4: AOBench subtopics. Slice labels give problem counts; geometry is excluded from both benchmark cohorts.

The replication set contains all 22 algebra, combinatorics, and number-theory problems in the Advanced split of IMO-ProofBench. Geometry was excluded by the same pre-specified rule. The resulting domain counts are 8 algebra, 8 combinatorics, and 6 number theory.

A.2 Harness and stopping protocol

The Python harness uses Claude Agent SDK version 0.1.56, with automatic context compaction configured at 900k tokens. Solvers execute code in a local scratch directory; WebSearch and WebFetch tools are blocked. Appendix G gives a full worked example.

After a trajectory stops, later budget checkpoints retain its last complete proof. Cumulative success records whether any evaluated checkpoint has a passing proof; an incorrect terminal proof does not erase an earlier success. Among 171 trajectories per model and arm, self-converged runs with no passing checkpoint number 38, 22, 0, and 4 for unaided Muse, GPT-5.4, GPT-5.5, and Opus, respectively, and 27, 7, 1, and 0 for their oracle arms.

B PROOFS AND DERIVATIONS

B.1 PROOF OF PROPOSITION 1

Since ε is cumulative, saturated first-block execution, $\varepsilon(1) = 1$, implies $\varepsilon(\ell) = 1$ for every $\ell \geq 1$. Substituting geometric discovery into Eq. 2 gives

$$s(K) = \sum_{k=1}^{K} \alpha(1-\alpha)^{k-1} = 1 - (1-\alpha)^K.$$

At one block, $q = \alpha\varepsilon(1) = \alpha$. For N independent attempts with $NK = B$, the probability that at least one solves is therefore

$$1 - (1 - s(K))^N = 1 - (1-q)^{NK} = 1 - (1-q)^B,$$

which is independent of the allocation. □

Extension to flat, unsaturated execution. More generally, if $\varepsilon(\ell) = c \in (0, 1]$ for every $\ell \geq 1$, then $K = 1$, $N = B$ is an optimal allocation, and is uniquely optimal when $c < 1$. This distinguishes saturated execution from execution that remains flat below one. Substituting geometric discovery and constant execution probability into Eq. 2 gives

$$\begin{aligned} s(K) &= c \sum_{k=1}^{K} p(k) \\ &= c\alpha \sum_{k=1}^{K} (1-\alpha)^{k-1} \\ &= c\left[1 - (1-\alpha)^K\right]. \end{aligned}$$

Let $N_K = \lfloor B/K \rfloor$. Consequently, the success probability is

$$1 - \left[1 - c + c(1-\alpha)^K\right]^{N_K}.$$

By convexity of $x \mapsto x^K$,

$$\begin{aligned} 1 - c + c(1-\alpha)^K &= (1-c)1^K + c(1-\alpha)^K \\ &\geq \left[(1-c)1 + c(1-\alpha)\right]^K \\ &= \left[1 - c + c(1-\alpha)\right]^K. \end{aligned}$$

Raising both sides to N_K yields

$$\left[1 - c + c(1-\alpha)^K\right]^{N_K} \geq \left[1 - c + c(1-\alpha)\right]^{KN_K}.$$

Because $KN_K \leq B$ and the bracketed quantity lies in $[0, 1]$,

$$\left[1 - c + c(1-\alpha)\right]^{KN_K} \geq \left[1 - c + c(1-\alpha)\right]^B.$$

For $0 < c < 1$ and $K > 1$, the inequality is strict; subtracting both sides from one proves that $K = 1$ uniquely maximizes success. If $c = 1$, equality holds in the convexity step, and the allocation has success probability

$$1 - (1-\alpha)^{K\lfloor B/K \rfloor}.$$

This equals the $K = 1$ value $1 - (1-\alpha)^B$ whenever $K\lfloor B/K \rfloor = B$, and is otherwise weakly smaller. □

B.2 PROOF OF PROPOSITION 2

Under geometric discovery, Eq. 4 gives

$$s(K) = \sum_{k=1}^{K} \alpha(1-\alpha)^{k-1}\varepsilon(K-k+1).$$

For fixed K, as $\alpha \to 0$, $(1-\alpha)^{k-1} = 1 + O(\alpha)$, so

$$s(K) = \alpha \sum_{\ell=1}^{K} \varepsilon(\ell) + O(\alpha^2). \tag{10}$$

Let $N_K = \lfloor B/K \rfloor$. Since $s(K) = O(\alpha)$,

$$\begin{aligned} S_B(K) &= 1 - \big(1 - s(K)\big)^{N_K} \\ &= \alpha N_K \sum_{\ell=1}^{K} \varepsilon(\ell) + O(\alpha^2). \end{aligned} \tag{11}$$

Define

$$\bar{\varepsilon}(K) = \frac{1}{K} \sum_{\ell=1}^{K} \varepsilon(\ell).$$

By definition, $\varepsilon(K)$ is non-decreasing: completing a proof within K blocks also constitutes completing it within $K+1$ blocks. Consequently,

$$\bar{\varepsilon}(K+1) - \bar{\varepsilon}(K) = \frac{\varepsilon(K+1) - \bar{\varepsilon}(K)}{K+1} \geq 0.$$

Moreover, the assumed strict improvement implies $\bar{\varepsilon}(K) < \bar{\varepsilon}(B)$ for every $K < B$. Since $N_K K \leq B$, for every $K < B$,

$$N_K K \bar{\varepsilon}(K) < B\bar{\varepsilon}(B).$$

Hence the first-order coefficient in Eq. 11 is uniquely maximized by $K = B$, for which $N_B = 1$. □

C Framework estimation and prediction

For problem n, let u_n of m_n independent one-block unaided attempts succeed. Let $c_n^{(j)}$ count oracle trajectories first solving in block j, with category $B+1$ denoting no success through block B. Write $r_n = \sum_{j=1}^{B+1} c_n^{(j)}$. Our experiments use $m_n = 24$, $r_n = 3$, and $B = 8$. Success is cumulative: a passing proof at any earlier checkpoint counts as success thereafter. All fits below use these measurement runs, never the target unaided outcomes. We write $S_n(N, K)$ for the probability that at least one of N independent K-block arms succeeds.

C.1 Simple Geometric (SG)

Under SG, each block is an independent opportunity with success probability q_n, so $S_n^{\mathrm{SG}}(N, K) = 1 - (1-q_n)^{NK}$. Rather than substitute $\widehat{q}_n = u_n/m_n$, we use the finite-sample pass@NK estimator

$$\widehat{S}_n^{\mathrm{SG}}(N, K) = 1 - \frac{\binom{m_n - u_n}{NK}}{\binom{m_n}{NK}}, \qquad NK \leq m_n. \tag{12}$$

The ratio is the fraction of size-NK subsets of observed attempts containing only failures; its numerator is zero when $m_n - u_n < NK$. For independent identically distributed short attempts, this estimates $1 - (1-q_n)^{NK}$ without plug-in bias. SG uses no oracle observations and assigns equal success probability to allocations with the same total budget.

C.2 Discovery–Execution (DE)

DE separates one-block success into discovery and execution, $q_n = \alpha_n \varepsilon_n(1)$. Define the empirical execution curve $\widehat{\varepsilon}_n(\ell) = r_n^{-1} \sum_{j=1}^{\ell} c_n^{(j)}$. We estimate discovery by dividing the short-attempt success rate by first-block oracle completion, subject to $0 \leq \alpha_n \leq 1$:

$$\widehat{\alpha}_n = \begin{cases} 0, & \widehat{q}_n = \widehat{\varepsilon}_n(1) = 0, \\ \widehat{q}_n / \widehat{\varepsilon}_n(1), & \widehat{q}_n \leq \widehat{\varepsilon}_n(1), \quad \widehat{\varepsilon}_n(1) > 0, \\ 1, & \widehat{q}_n > \widehat{\varepsilon}_n(1). \end{cases} \tag{13}$$

We denote the adjusted execution curve by $\varepsilon_n^\star$. In the first two cases, $\varepsilon_n^\star(\ell) = \widehat{\varepsilon}_n(\ell)$. When both first-block rates are zero, discovery is unidentified; choosing zero is an implementation convention and yields zero predicted unaided success.

In the third case, no $\alpha_n \in [0,1]$ matches the fresh success rate while retaining the empirical oracle curve. We set $\widehat{\alpha}_n = 1$, so fresh and oracle first-block outcomes estimate the same execution probability. Pooling their counts gives

$$\varepsilon_n^\star(1) = \frac{u_n + c_n^{(1)}}{m_n + r_n}. \tag{14}$$

Changing first-block completion also changes the fraction of trajectories available to solve later. We preserve the observed completion rate *among trajectories unsolved after block one*. Of the $r_n - c_n^{(1)}$ such oracle trajectories, $\sum_{j=2}^{\ell} c_n^{(j)}$ solve by block ℓ. We estimate the conditional completion probability as

$$\delta_n(\ell) = \frac{\sum_{j=2}^{\ell} c_n^{(j)}}{r_n - c_n^{(1)}} = \frac{\widehat{\varepsilon}_n(\ell) - \widehat{\varepsilon}_n(1)}{1 - \widehat{\varepsilon}_n(1)}. \tag{15}$$

Applying this conditional probability to the probability remaining after block one gives the adjusted cumulative execution probability:

$$\varepsilon_n^\star(\ell) = \underbrace{\varepsilon_n^\star(1)}_{\text{solved in block one}} + \underbrace{\big(1 - \varepsilon_n^\star(1)\big)\delta_n(\ell)}_{\text{solved later, by block } \ell}, \qquad 1 \le \ell \le B. \tag{16}$$

This preserves the relative frequencies of later completion and noncompletion and keeps the curve nondecreasing. Predictions then use the fitted discovery–execution convolution:

$$\widehat{s}_n(K) = \sum_{j=1}^{K} \widehat{\alpha}_n(1-\widehat{\alpha}_n)^{j-1}\varepsilon_n^\star(K-j+1), \qquad \widehat{S}_n^{\mathrm{DE}}(N,K) = 1 - \big(1 - \widehat{s}_n(K)\big)^N. \tag{17}$$

C.3 Regularized Simple Geometric (R-SG)

R-SG models each problem's success probability using a shared Beta prior. For problem n, the prior and likelihood of u_n successes in m_n short attempts are

$$q_n \sim \mathrm{Beta}(a,b), \qquad L_n(q_n) = \binom{m_n}{u_n} q_n^{u_n}(1-q_n)^{m_n - u_n}. \tag{18}$$

For each LLM, we determine $(\widehat{a}, \widehat{b})$ to maximize the product of expected likelihoods across all 57 problems. Writing $\mathrm{B}(\cdot,\cdot)$ for the beta function,

$$\begin{aligned}(\widehat{a},\widehat{b}) &= \arg\max_{a,b>0} \sum_{n=1}^{57} \log \mathbb{E}_{q_n \sim \mathrm{Beta}(a,b)}[L_n(q_n)] \\ &= \arg\max_{a,b>0} \sum_{n=1}^{57} \left[\log \mathrm{B}(a+u_n, b+m_n-u_n) - \log \mathrm{B}(a,b)\right].\end{aligned} \tag{19}$$

We optimize $\log a, \log b \in [-12, 12]$ with SciPy.

With the fitted prior fixed, conjugacy between the Beta prior and binomial likelihood gives the posterior in closed form:

$$q_n \mid u_n, m_n \sim \mathrm{Beta}\Big(\widehat{a} + u_n, \widehat{b} + m_n - u_n\Big). \tag{20}$$

Averaging geometric success over this posterior gives the final estimator:

$$\begin{aligned}\widehat{S}_n^{\text{R-SG}}(N,K) &= 1 - \mathbb{E}\big[(1-q_n)^{NK} \mid u_n, m_n\big] \\ &= 1 - \frac{\mathrm{B}(\widehat{a}+u_n, \widehat{b}+m_n-u_n+NK)}{\mathrm{B}(\widehat{a}+u_n, \widehat{b}+m_n-u_n)} \\ &= 1 - \prod_{j=0}^{NK-1} \frac{\widehat{b}+m_n-u_n+j}{\widehat{a}+\widehat{b}+m_n+j}.\end{aligned} \tag{21}$$

C.4 Regularized Discovery–Execution (R-DE)

R-DE uses the same empirical-Bayes approach as R-SG, but learns separate distributions for discovery and execution. Let $\pi_n^{(j)}$ be the probability that execution first completes in block j, with category $B+1$ denoting noncompletion within the budget. Then $\varepsilon_n(\ell) = \sum_{j=1}^{\ell} \pi_n^{(j)}$. We place independent priors on discovery probability and execution outcomes:

$$\alpha_n \sim \mathrm{Beta}(a, b), \qquad \boldsymbol{\pi}_n \sim \mathrm{Dirichlet}(\boldsymbol{d}). \tag{22}$$

The two types of measurement runs inform different parts of this model. A short unaided attempt succeeds only if discovery and execution both succeed within the first block, with probability $\alpha_n \pi_n^{(1)}$. Oracle runs supply the strategy and therefore observe execution directly: $c_n^{(j)}$ counts trajectories that first complete in block j. Combining these observations gives the likelihood, up to coefficients depending only on the data:

$$L_n(\alpha_n, \boldsymbol{\pi}_n) \propto [\alpha_n \pi_n^{(1)}]^{u_n} [1 - \alpha_n \pi_n^{(1)}]^{m_n - u_n} \prod_{j=1}^{B+1} [\pi_n^{(j)}]^{c_n^{(j)}}. \tag{23}$$

As in R-SG, we fit the shared prior by maximizing the product of expected likelihoods across the 57 problems for each LLM. Here both the Beta and Dirichlet shapes are fitted jointly:

$$(\widehat{a}, \widehat{b}, \widehat{\boldsymbol{d}}) = \arg\max_{a,b,\boldsymbol{d}} \sum_{n=1}^{57} \log \mathbb{E}_{\substack{\alpha_n \sim \mathrm{Beta}(a,b) \\ \boldsymbol{\pi}_n \sim \mathrm{Dirichlet}(\boldsymbol{d})}} [L_n(\alpha_n, \boldsymbol{\pi}_n)]. \tag{24}$$

For the main fit, the Dirichlet distribution is defined on the execution categories observed in the pooled oracle measurements. We fit positive shapes for those categories and set the probabilities of all other categories to zero. Zero entries in $\boldsymbol{d}$ are bookkeeping for these omitted coordinates, not parameters of a full-dimensional Dirichlet density; Dirichlet expressions below are understood on the retained coordinates. We optimize the retained log-shapes with SciPy using the same bounds as R-SG. The transfer experiment in Appendix E.7 instead retains all $B+1$ categories, allowing outcomes absent from the prior-fitting problems to occur on held-out problems.

With the fitted prior fixed, each problem's observations $\mathcal{D}_n = (u_n, m_n, \boldsymbol{c}_n)$ determine its posterior. The oracle update is immediate: execution follows $\mathrm{Dirichlet}(\widehat{\boldsymbol{d}} + \boldsymbol{c}_n)$ before incorporating the short attempts. The short-attempt update is less direct because a failure does not reveal whether discovery or execution failed.

To express this uncertainty, let h_n count the failed short attempts in which discovery succeeded but execution did not finish. If this count were known, discovery would have $u_n + h_n$ successes and $m_n - u_n - h_n$ failures. First-block execution would have u_n successes and h_n failures in addition to the oracle observations. Conjugacy then gives independent conditional posteriors:

$$\begin{aligned} \alpha_n \mid h_n, \mathcal{D}_n &\sim \mathrm{Beta}\Big(\widehat{a} + u_n + h_n, \widehat{b} + m_n - u_n - h_n\Big), \\ \pi_n^{(1)} \mid h_n, \mathcal{D}_n &\sim \mathrm{Beta}\left(\widehat{d}_1 + c_n^{(1)} + u_n, \sum_{j=2}^{B+1} (\widehat{d}_j + c_n^{(j)}) + h_n\right). \end{aligned} \tag{25}$$

Among outcomes unfinished after block one, the relative execution probabilities $(\pi_n^{(j)}/(1-\pi_n^{(1)}))_{j>1}$ retain the Dirichlet distribution with shapes $(\widehat{d}_j + c_n^{(j)})_{j>1}$, independently of the two Beta variables for fixed h_n. These proportions come from the oracle update because short attempts do not observe later completion. Since h_n is unknown, the full posterior averages these conditional distributions over $h_n = 0, \ldots, m_n - u_n$, weighted by each count's posterior probability given the observations.

Prediction now follows the same principle as R-SG: average the success formula over the posterior. For each possible h_n, we average DE success over discovery and execution uncertainty, then average those predictions over h_n:

$$\widehat{S}_n^{\text{R-DE}}(N, K) = \mathbb{E}_{h_n \mid \mathcal{D}_n}\Big[\mathbb{E}_{\alpha_n, \boldsymbol{\pi}_n \mid h_n, \mathcal{D}_n}\Big[1 - \big(1 - s_n(K; \alpha_n, \boldsymbol{\pi}_n)\big)^N\Big]\Big]. \tag{26}$$

Here $s_n(K; \alpha_n, \boldsymbol{\pi}_n)$ is the single-arm DE success probability from Eq. 4. The result is the posterior predictive probability that at least one of N arms solves problem n within K blocks each. The inner expectation is evaluated exactly using closed-form Beta and Dirichlet moments; the outer expectation is a finite weighted sum over the possible failure counts.

D ADDITIONAL PREDICTION RESULTS

D.1 PREDICTIONS ACROSS COMPUTE ALLOCATIONS

Figure 5 shows the Parallel-2 held-out curves underlying Table 1. Each trial succeeds if either of its two independent trajectories produces a passing proof within the per-arm budget. Predictions use the same Parallel-8 and Oracle-Execution measurement runs that fit Parallel-1. For Muse, R-DE predicts 77.3 final successes against 77 observed; the largest gap is at the first checkpoint, with 57.7 predicted against 68 observed.

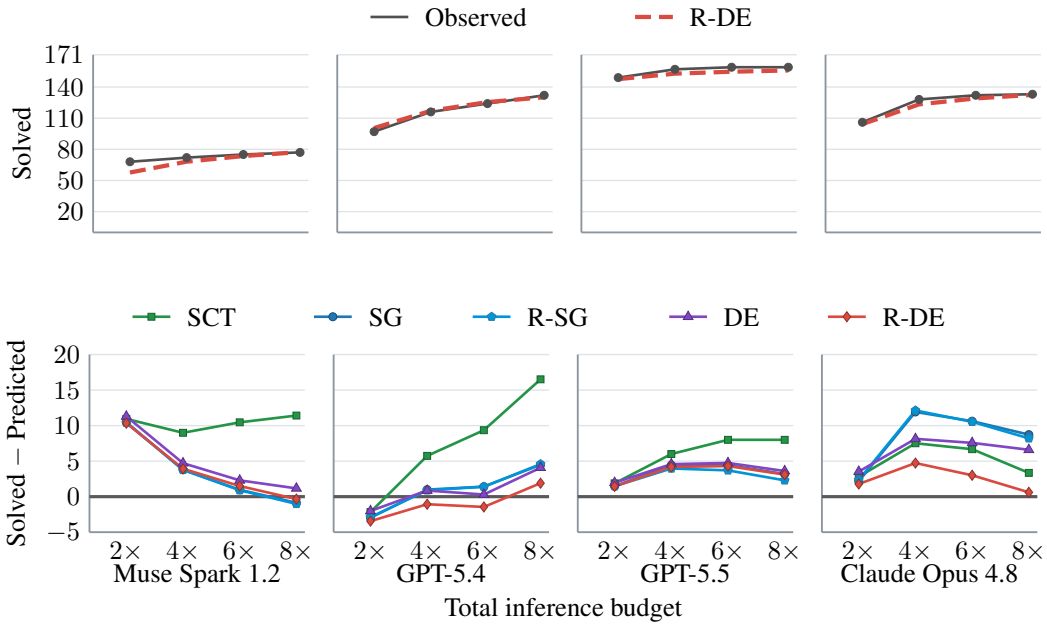


Figure 5: **Predicting two-arm success ($N = 2$).** Top: observed successes and R-DE predictions. Bottom: observed minus predicted solved counts. Each curve represents 171 allocation trials across 57 problems; the horizontal axis gives total compute across both arms.

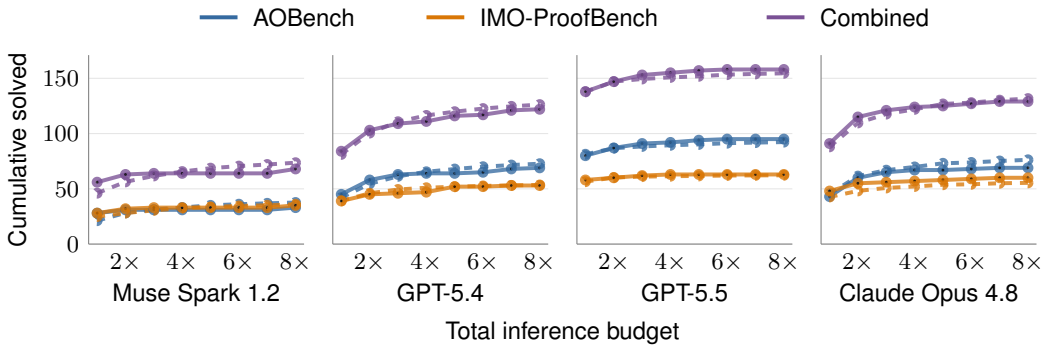
Table 4 extends the comparison to $N = 4$ arms of $K = 2$ blocks each for Muse and GPT-5.5. Each model contributes three allocation trials per problem, and success requires at least one of the four arms to solve. All predictors retain their measurement-run fits. For GPT-5.5, R-SG has the lowest error; for Muse, all predictors substantially underestimate success, including R-DE, which predicts 78.5 final successes against 93 observed. The additional allocation therefore exposes a transfer failure despite Muse's close final prediction at $N = 2$.

Table 4: Prediction RMSE in solved-trial counts for $N = 4$, evaluated across both checkpoints of four arms with two blocks each. Each model has 57 problems and 171 allocation trials. Predictions use the unchanged intervention fit. Bold marks row minima.

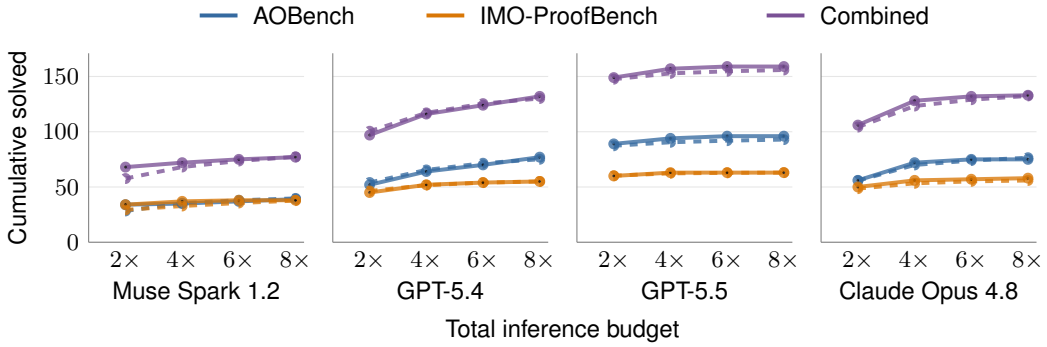
Model	SG	R-SG	SCT	DE	R-DE
Muse Spark 1.2	17.06	16.97	19.40	18.77	**16.64**
GPT-5.5	1.85	**1.14**	1.71	2.27	1.57

D.2 Dataset-separated predictions

Figure 6 separates the $N = 1$ and $N = 2$ results into AOBench and IMO-ProofBench. Each panel uses the same fitted parameters as the pooled evaluation and sums predictions and observed successes only over the indicated dataset. AOBench contributes 105 trials and IMO-ProofBench 66.



(a) $N = 1$: one arm, eight blocks.



(b) $N = 2$: two arms, four blocks each.

Figure 6: **Dataset-separated scaling.** Solid lines show observed successes; dashed lines show R-DE predictions. Colors identify each dataset and their combined total.

D.3 Problem-level prediction

Table 5 compares predictions with the six individual trajectories from Parallel-2 per problem at blocks 1–4, predicting each trajectory with $N = 1$. For each problem and checkpoint, we compare the predicted solved count with the observed count out of six. We square these errors, average over all 57×4 problem–checkpoint pairs, and take the square root. Fits use the same Parallel-8 and Oracle-Execution measurements as the main predictions.

Table 5: Problem-level RMSE in solved-trajectory counts, using the six individual Parallel-2 trajectories per problem over blocks 1–4. Bold marks row minima.

Model	SG	R-SG	SCT	DE	R-DE
Muse Spark 1.2	1.27	1.23	1.26	1.19	**1.09**
GPT-5.4	1.22	1.21	1.51	1.20	**1.16**
GPT-5.5	0.71	**0.69**	1.10	0.74	0.71
Claude Opus 4.8	0.93	0.92	1.31	0.92	**0.86**
Average	1.06	1.04	1.31	1.03	**0.97**

R-DE has the lowest problem-level RMSE for Muse, GPT-5.4, and Opus, while R-SG performs best for GPT-5.5. Problem-level improvements are smaller than aggregate improvements: for Opus, R-DE reduces RMSE from SG's 0.93 to 0.86 solved trajectories.

D.4 Prediction accuracy across allocations

We compare the fitted predictors across the two target allocations using

$$\mathrm{RMSE}_{\mathrm{combined}} = \sqrt{\frac{\mathrm{RMSE}_{N=1}^2 + \mathrm{RMSE}_{N=2}^2}{2}}. \tag{27}$$

Each allocation has 171 trials across 57 problems. Allocation-specific MSE averages over eight checkpoints for $N = 1$ and four for $N = 2$, so Eq. 27 gives the allocations equal weight. This exploratory comparison follows the allocation-specific analysis in Table 1.

We hold the 57 problems and all fitted predictions fixed and perform 100,000 paired target-trial bootstrap replicates (random seed 1234). Within each problem, we sample three allocation trials with replacement, retaining each trial's complete cumulative success history. An $N = 1$ trial is one eight-block trajectory; an $N = 2$ trial is a recorded pair evaluated through four blocks per arm, successful if either arm solves. Every predictor is evaluated against the same resampled outcomes. Resampling preserves any shared-trajectory dependence across allocations, while separate target trials are resampled independently.

Table 6: **Combined held-out prediction accuracy across $N = 1$ and $N = 2$.** Equal-allocation RMSE in solved-trial counts. Reductions are baseline minus R-DE, so positive values favor R-DE; brackets give 95% paired target-bootstrap percentile intervals.

	Combined RMSE			RMSE reduction relative to baseline	
Model	SG	R-SG	R-DE	SG minus R-DE	R-SG minus R-DE
Muse	7.06	7.08	5.93	1.13 $[-0.05, 1.80]$	1.15 $[-0.06, 1.84]$
GPT-5.4	3.44	3.45	3.17	0.27 $[-0.85, 1.26]$	0.28 $[-0.84, 1.27]$
GPT-5.5	3.05	2.52	3.56	-0.51 $[-0.77, 0.18]$	-1.04 $[-1.27, -0.02]$
Opus	8.22	8.20	2.91	5.31 $[2.29, 5.95]$	5.29 $[2.29, 5.92]$

For Opus, R-DE lowers combined RMSE by 5.31 relative to SG and 5.29 relative to R-SG, with both percentile intervals above zero (Table 6). The corresponding comparisons for Muse and GPT-5.4 include zero, while GPT-5.5 favors R-SG over R-DE.

Pointwise 95% percentile intervals quantify target-sampling uncertainty with the benchmark and fitted predictors fixed, without multiplicity adjustment. With three trials per problem and allocation, unanimous outcomes have zero bootstrap variance. Appendix E.7 separately assesses sensitivity to prior-fitting problems.

D.5 Continue-or-restart evaluation

At each checkpoint $k = 1, \ldots, 7$, the continuation count C_n is the number of the problem's three Parallel-1 trajectories that first achieve a passing proof at $k + 1$. The restart count R_n estimates additional successes among trajectories still unsolved at k, using the six individual trajectories from Parallel-2 for that problem. These restart trajectories are separate from the continuation trajectories and are not used to fit the predictors. R-DE selects an action using its fitted expected advantage; regret compares the selected count with $\max\{C_n, R_n\}$, sums across problems, and averages across checkpoints, as in Eq. 9.

These are descriptive regret comparisons; the additional supplement reports uncertainty analyses.

E ROBUSTNESS AND TRANSFER

E.1 ASSUMPTION 1: EXECUTION INDEPENDENT OF ACQUISITION TIME

We compare no sketch, a sketch supplied at the start, and a sketch supplied after an unaided prefix. For each model, we select problem–seed trials whose recorded $3\times$ proof in the no-sketch arm scores below 5/7. No sketch and late sketch continue the same native prefix session for one additional block; start sketch uses the corresponding oracle $1\times$ checkpoint.

Table 7: AOBench sketch-timing comparison: solved/selected trajectories (score $\geq 5/7$), with $\Delta\% = 100(\text{late} - \text{start})/\text{start}$.

Model	No sketch	Start sketch	Late sketch	$\Delta\%$
Muse Spark 1.2	1/71	34/71	17/71	-50.0%
GPT-5.4	8/45	30/45	29/45	-3.3%
GPT-5.5	3/12	11/12	11/12	0.0%
Claude Opus 4.8	3/48	20/48	20/48	0.0%

The GPT start- and late-sketch counts differ by at most one solved trajectory, while Opus solves 20 trajectories in either sketch condition versus 3 without a sketch. Muse instead solves 34 trajectories with a sketch at the start and 17 with a late sketch, showing that its conditional completion is more sensitive to this change in context.

E.2 ASSUMPTION 2: EXECUTION ACROSS VIABLE STRATEGIES

To assess whether the execution measurement depends on which reference strategy is supplied, we replace the original oracle sketch with a sketch of a human-checked AI-generated alternative proof on 38 problems: 24 from AOBench and 14 from IMO-ProofBench. Both sketches contain at most 25 words and use the same prompt format. Each condition has three standalone $1\times$ runs per problem for Muse, GPT-5.5, and Opus, giving 114 trajectories per model and condition. Both conditions are graded by GPT-5.6 Sol against their respective reference solutions; success means any valid proof scoring at least 5/7.

Table 8: Solved $1\times$ trajectories on 38 matched problems (24 AOBench, 14 IMO-ProofBench), with three seeds each. $\Delta\% = 100(\text{alternate} - \text{original})/\text{original}$.

Model	Original sketch	Alternate sketch	$\Delta\%$
Muse Spark 1.2	58/114	66/114	$+13.8\%$
GPT-5.5	109/114	111/114	$+1.8\%$
Claude Opus 4.8	88/114	82/114	-6.8%

Out of 114 trajectories per condition, Muse solves 58 with the original sketch and 66 with the alternate sketch; GPT-5.5 solves 109 and 111, respectively, and Opus solves 88 and 82.

E.3 ASSUMPTION 3: GEOMETRIC DISCOVERY

We test sensitivity to reduced late discovery by replacing geometric discovery with

$$p_n^{(\gamma)}(k) = \alpha_n\big((1-\alpha_n)\gamma\big)^{k-1}, \qquad 0 \leq \gamma \leq 1. \tag{28}$$

The case $\gamma = 1$ recovers the original framework; smaller values reduce late discovery. We hold the measurement-fitted R-DE posterior fixed and fit one γ per model to first-solve times from the 171 Parallel-1 trajectories, jointly integrating each problem's three trajectories over its shared parameters. We use 16,384 posterior draws per problem, with seeds 1234–1237 for Muse, GPT-5.4, GPT-5.5, and Opus, respectively.

Table 9: Geometric-discovery sensitivity on all 57 problems. $\Delta \log L$ is the improvement over $\gamma = 1$; the final column is the largest absolute change in predicted solved trajectories across the eight checkpoints.

Model	Trajectories	$\widehat{\gamma}$	$\Delta \log L$	Max. count change
Muse Spark 1.2	171	0.692	7.58	12.48
GPT-5.4	171	1.000	0.00	0.00
GPT-5.5	171	0.972	0.19	0.79
Claude Opus 4.8	171	0.974	0.12	0.86

Within this family, the GPT and Opus fits remain close to geometric discovery: $\widehat{\gamma}$ lies between 0.972 and 1, changing aggregate predictions by fewer than one solved trajectory. Muse instead favors reduced late discovery ($\widehat{\gamma} = 0.692$), with a maximum change of 12.48 solved trajectories.

E.4 Sketch-content control

To distinguish useful strategic content from simply adding text, we assign each problem the next problem's sketch in a sorted within-domain list, with cyclic wraparound. This preserves the sketch multiset and prompt wrapper. We compare no sketch, shuffled sketch, and the correct oracle sketch on all 35 AOBench problems, with three standalone $1\times$ runs per condition.

Table 10: Sketch-content control on all 35 AOBench problems and three seeds. The last three columns count solved standalone $1\times$ trajectories (score $\geq 5/7$); seeds are counted individually.

Model	Trajectories	No sketch	Shuffled sketch	Oracle sketch
Muse Spark 1.2	105	24	13	48
GPT-5.4	105	41	38	81
GPT-5.5	105	83	77	104
Claude Opus 4.8	105	45	41	73

Correct oracle sketches increase solved-trajectory counts for all four models; shuffled sketches do not improve over no sketch.

E.5 First-block consistency

We compare the 24 short unaided attempts per problem with the first checkpoint of the three eight-block unaided trajectories. Across all 57 problems, these give 1,368 short and 171 long-run outcomes per model. Table 11 reports the percentage scoring at least 5/7 in each condition and their difference in percentage points. Every problem contributes the same number of trials within each condition, so these percentages also equal the averages of the problem-level success rates.

Table 11: First-block success across all 57 problems: 1,368 short runs and 171 long runs per model. Differences are long minus short, in percentage points (pp), computed before rounding.

Model	Short (%)	Long, block 1 (%)	Difference (pp)
Muse Spark 1.2	27.0	32.7	+5.8
GPT-5.4	47.3	49.1	+1.8
GPT-5.5	80.6	80.7	+0.1
Claude Opus 4.8	51.6	53.2	+1.6

Muse has the largest observed difference, with long-run first-block success 5.8 percentage points above short-run success. The other models differ by 0.1–1.8 percentage points.

E.6 Execution ablations

We ablate ordinary DE using the same measurement runs and held-out $N = 1$ trajectories. *Flat execution* retains each problem's fitted discovery rate but sets every execution checkpoint to its fitted first-block value. *Shared execution* pools the model's 171 oracle trajectories into one empirical curve $\bar{\varepsilon}(k)$ and sets each problem's discovery rate to $\min\{1, \widehat{q}_n/\bar{\varepsilon}(1)\}$. The shared curve remains fixed across problems; target outcomes are used only for evaluation.

Table 12: Execution ablations: $N = 1$ RMSE in solved-trajectory counts across eight checkpoints, with 171 held-out trajectories per model. Lower is better; bold marks row minima.

Model	DE	Flat execution	Shared execution
Muse Spark 1.2	6.54	**5.22**	17.17
GPT-5.4	**4.25**	4.35	8.41
GPT-5.5	**3.04**	5.31	5.67
Claude Opus 4.8	**3.68**	19.47	21.20

Flattening execution substantially worsens Opus's predictions but improves Muse's, so measured execution gains do not transfer uniformly. Shared execution performs worse for every model. This ablation also constrains first-block fit: for Opus, 25 of 57 short success rates exceed $\bar{\varepsilon}(1)$. Its error therefore reflects the common execution level as well as the loss of problem-specific temporal variation.

E.7 Transfer across problems

We split the 57 problems into 38 for prior fitting and 19 for evaluation, repeating this 50 times with the same splits for all four models (random seed 1234). Each split evaluates aggregate $N = 1$ predictions at blocks 1–4 on the six individual Parallel-2 trajectories per test problem, giving 114 test trajectories. Table 13 compares shared priors fitted to all 57 problems with priors fitted only to the 38 training problems, then tests prediction without oracle observations from the evaluation problems. Target outcomes are never used for fitting.

Without test oracle measurements, all counts $\boldsymbol{c}_n$ in Appendix C, including noncompletion, are zero; predictions still average over the joint discovery–execution posterior. R-SG learns its Beta prior from the training problems' short runs.

Table 13: **Transfer across problems.** Mean $\pm$ SD of solved-count RMSE over 50 splits, each using the six individual Parallel-2 trajectories for each of 19 test problems (114 trajectories), evaluated at blocks 1–4 with $N = 1$. All conditions except the first R-DE column fit priors on the 38 training problems.

	R-DE			R-SG
Model	All 57 + oracle	Train 38 + oracle	Train 38, no oracle	Train 38
Muse Spark 1.2	4.57 ± 1.90	4.60 ± 1.90	4.62 ± 2.08	5.05 ± 2.13
GPT-5.4	4.19 ± 2.15	4.16 ± 2.05	4.08 ± 1.92	4.12 ± 2.12
GPT-5.5	3.78 ± 1.83	3.54 ± 1.66	3.56 ± 1.53	2.95 ± 1.41
Claude Opus 4.8	3.62 ± 1.47	3.40 ± 1.46	5.04 ± 2.67	6.24 ± 2.09

Restricting prior fitting to 38 problems changes mean R-DE test RMSE by less than 0.25 solved trajectories for every model. Without test oracle measurements, R-DE improves on R-SG for Muse and Opus, is close for GPT-5.4, and performs worse for GPT-5.5. Opus benefits further from its own oracle measurements, reducing RMSE from 5.04 to 3.40. Execution information can therefore transfer to problems without supplied sketches. Standard deviations describe variation across overlapping splits, not uncertainty from independent replications.

F AUDIT PROTOCOL AND HUMAN VALIDATION



Figure 7: Automated versus human proof scores for all 50 completed audits. Percentages are normalized within each row; n gives the number of proofs in that row.

The proof judge receives the problem, submitted proof, and reference solution, without the solver identity or condition. Alternate proofs are allowed; the reference is not a required route. Scores ≥ 5 count as solved: minor local defects are allowed, but load-bearing gaps and missing cases fail. A separate tool-free state judge checks a frozen three-step outline, marking a step present only when its mechanism and role are explicit. These labels are diagnostic, not inputs to discovery estimation.

An IMO medalist panel reviewed 50 anonymized proofs in disjoint sets. Each set contains seven automated score-0 proofs and six each with scores 5, 6, and 7; score-0 proofs are also stratified by reference-step presence. Auditors see the problem, reference solution, three-step outline, and submitted proof, but not solver identities or automated scores. We measure human–automated agreement on this stratified sample.

Solved status agrees on 48/50 proofs, giving 100.0% precision, 94.7% recall, and 97.3% F1 against human labels. Thirty proof scores agree exactly; in every disagreement, the automated judge assigns the lower score (Figure 7). Reference-step presence agrees on 140/150 labels; these diagnostic labels are not used to fit the frameworks.

G A WORKED EXAMPLE

We reproduce a GPT-5.5 Oracle-Execution trajectory on Romania TST 2026, Problem 7 (seed 3), item T21 in the human audit. The full reference proof, submitted proof, and audit judgments appear below with typesetting changes only.

Problem. Do there exist a positive integer N and an infinite sequence $a_1, a_2, a_3, \ldots$ of positive integers such that

$$a_{n+1} = \frac{a_1 a_2 \cdots a_n}{a_1 + a_2 + \cdots + a_n} \quad \text{for all } n \geq N\,?$$

Supplied sketch. Set $S_n = \sum_{i \leq n} a_i$; derive $S_{n+1} \mid S_n^3$ and stabilize prime supports. For p, persistent $v_p(a_{n+1}) \leq v_p(S_n)$ forces unbounded 2-divisibility; once reversed, the inequality persists. Combine stabilized valuations against growth.

Reference outline (audit only).

1. Split the accumulated sum and the newest term by their gcd into coprime parts to constrain the next sum's prime support.
2. Rule out the persistent-equality branch by iterating its linear valuation recursion and examining divisibility constraints on a fixed exponent
3. Induct forward a strict valuation inequality between the new term and the accumulated quantity, uniformly over the stabilized prime set.

Reference proof (audit only). The answer is **No**.

For $n \geq 1$, let $S_n = a_1 + a_2 + \cdots + a_n$ and $P_n = a_1 a_2 \cdots a_n$.

Suppose there exist such N and $(a_n)_{n\geq 1}$ satisfying $a_{n+1} = \frac{P_n}{S_n}$, i.e., $a_{n+1}S_n = P_n$ for all $n \geq N$.

Then for all $n \geq N$,

$$a_{n+2}S_{n+1} = P_{n+1} = a_{n+1}P_n = a_{n+1}^2 S_n. \tag{1}$$

Fix $n \geq N$. Let $g = \gcd(S_n, a_{n+1})$, so $S_n = gx$ and $a_{n+1} = gy$ for positive integers x, y with $\gcd(x, y) = 1$. Then (1) gives $a_{n+2}g(x+y) = g^3xy^2$, so $a_{n+2}(x+y) = g^2xy^2$, implying $x+y \mid g^2$. Hence $S_{n+1} = g(x + y) \mid g^3 \mid S_n^3$.

So for all $n \geq N$, $S_{n+1} \mid S_n^3$. In particular, every prime dividing S_{n+1} also divides S_n. Letting A_n denote the set of primes dividing S_n, we get $A_n \supseteq A_{n+1}$ for all $n \geq N$. Since A_N is finite, there exists $M \geq N$ such that $A_M = A_{M+1} = \cdots =: \mathcal{P}$.

Fix $p \in \mathcal{P}$. Applying ν_p to (1):

$$\nu_p(a_{n+2}) + \nu_p(S_{n+1}) = 2\nu_p(a_{n+1}) + \nu_p(S_n). \tag{2}$$

Lemma. There exists $n_0 \geq M$ such that $\nu_p(a_{n_0+k+1}) > \nu_p(S_{n_0+k})$ for all integers $k \geq 0$.

Proof. First we find $n_0 \geq M$ with $\nu_p(a_{n_0+1}) > \nu_p(S_{n_0})$. Assume for contradiction that $\nu_p(a_{n+1}) \leq \nu_p(S_n)$ for all $n \geq M$. If $\nu_p(a_{t+1}) < \nu_p(S_t)$ for some $t \geq M$, then $\nu_p(S_{t+1}) = \nu_p(a_{t+1})$, and (2) at $n = t$ gives $\nu_p(a_{t+2}) = \nu_p(a_{t+1}) + \nu_p(S_t) > \nu_p(S_{t+1})$, contradicting the assumption. So $\nu_p(a_{n+1}) = \nu_p(S_n)$ for all $n \geq M$. Then (2) becomes $2\nu_p(S_{n+1}) = 3\nu_p(S_n)$, giving $2^l\nu_p(S_{M+l}) = 3^l\nu_p(S_M)$ for all $l \geq 0$. Hence $2^l \mid \nu_p(S_M)$ for all l, which is impossible since $\nu_p(S_M)$ is a fixed positive integer. So such n_0 exists.

Now by induction: if $\nu_p(a_{n_0+k+1}) > \nu_p(S_{n_0+k})$, then $\nu_p(S_{n_0+k+1}) = \nu_p(S_{n_0+k})$, and (2) gives $\nu_p(a_{n_0+k+2}) = 2\nu_p(a_{n_0+k+1}) > 2\nu_p(S_{n_0+k}) = 2\nu_p(S_{n_0+k+1}) > \nu_p(S_{n_0+k+1})$, completing the induction. ■

Applying the lemma for each $p \in \mathcal{P}$ (a finite set), choose $M' \geq M$ such that $\nu_p(a_{n+1}) > \nu_p(S_n)$ for all $n \geq M'$ and all $p \in \mathcal{P}$. Then $\nu_p(S_{n+1}) = \nu_p(S_n)$ for all such n and p. Since $\mathcal{P}$ is the set of all prime divisors of S_n (and of S_{n+1}), the numbers S_n and S_{n+1} have the same prime factorization, so $S_n = S_{n+1}$, meaning $a_{n+1} = 0$. This contradicts (a_n) being a sequence of positive integers.

Therefore, no such N and $(a_n)_{n\geq 1}$ exist.

Model-generated proof. The critique found no genuine gap. Here is the complete standalone solution.

Final Solution

No such sequence exists.

Assume, for contradiction, that there exist a positive integer N and an infinite sequence of positive integers $a_1, a_2, a_3, \ldots$ such that

$$a_{n+1} = \frac{a_1 a_2 \cdots a_n}{a_1 + a_2 + \cdots + a_n} \tag{1}$$

for every $n \geq N$.

Let $S_n = a_1 + a_2 + \cdots + a_n$. For $m \geq 0$, define $X_m = S_{N+m}, \qquad Y_m = a_{N+m+1}$. Then

$$X_{m+1} = S_{N+m+1} = S_{N+m} + a_{N+m+1} = X_m + Y_m. \tag{2}$$

Applying (1) with $n = N + m$, we get $Y_m = a_{N+m+1} = \frac{a_1 a_2 \cdots a_{N+m}}{S_{N+m}} = \frac{a_1 a_2 \cdots a_{N+m}}{X_m}$. Thus

$$a_1 a_2 \cdots a_{N+m} = X_m Y_m. \tag{3}$$

Applying (1) with $n = N+m+1$, and using (3), gives $Y_{m+1} = a_{N+m+2} = \frac{a_1 a_2 \cdots a_{N+m+1}}{S_{N+m+1}} = \frac{X_m Y_m^2}{X_{m+1}}$. Hence

$$X_{m+1} Y_{m+1} = X_m Y_m^2. \tag{4}$$

In particular,

$$X_{m+1} \mid X_m Y_m^2. \tag{5}$$

Since $X_{m+1} = X_m + Y_m$, we have $Y_m \equiv -X_m \pmod{X_{m+1}}$. Therefore $X_m Y_m^2 \equiv X_m(-X_m)^2 = X_m^3 \pmod{X_{m+1}}$. Because $X_{m+1} \mid X_m Y_m^2$, this implies

$$X_{m+1} \mid X_m^3. \tag{6}$$

Let $\mathcal{P}$ be the set of prime divisors of $X_0 Y_0$. This set is finite. We claim that every prime divisor of every $X_m Y_m$ belongs to $\mathcal{P}$.

The claim is true for $m = 0$ by definition. Suppose it is true for some m. If a prime p divides X_{m+1}, then by (5), $p \mid X_m Y_m^2$. Since p is prime, $p \mid X_m$ or $p \mid Y_m$, so $p \mid X_m Y_m$. By the induction hypothesis, $p \in \mathcal{P}$.

If a prime p divides Y_{m+1}, then by (4), $p \mid X_m Y_m^2$. Again $p \mid X_m$ or $p \mid Y_m$, so $p \mid X_m Y_m$, hence $p \in \mathcal{P}$. Therefore every prime divisor of $X_{m+1} Y_{m+1}$ belongs to $\mathcal{P}$. By induction, every prime divisor of every $X_m Y_m$ belongs to $\mathcal{P}$.

Now fix a prime $p \in \mathcal{P}$. Define $e_m = v_p(X_m), \qquad f_m = v_p(Y_m)$. Taking p-adic valuations in (4), we obtain $e_{m+1} + f_{m+1} = e_m + 2f_m$, or equivalently,

$$f_{m+1} = e_m + 2f_m - e_{m+1}. \tag{7}$$

We prove that e_m is eventually constant.

First suppose $f_m > e_m$. Write $X_m = p^{e_m} u, \qquad Y_m = p^{f_m} v$, where $p \nmid u$ and $p \nmid v$. Then $X_{m+1} = X_m + Y_m = p^{e_m}\left(u + p^{f_m - e_m} v\right)$. Since $f_m - e_m > 0$, the integer $p^{f_m - e_m} v$ is divisible by p, while u is not divisible by p. Hence $p \nmid u + p^{f_m - e_m} v$, so $e_{m+1} = e_m$. Using (7), $f_{m+1} = e_m + 2f_m - e_{m+1} = 2f_m$. Thus $f_{m+1} = 2f_m > e_m = e_{m+1}$. Therefore, once $f_m > e_m$, the inequality $f_t > e_t$ holds for every later index t, and e_t is constant from that point onward.

Next suppose $f_m < e_m$. Write $X_m = p^{e_m} u, \qquad Y_m = p^{f_m} v$, where $p \nmid u$ and $p \nmid v$. Then $X_{m+1} = X_m + Y_m = p^{f_m}\left(p^{e_m - f_m} u + v\right)$. Since $e_m - f_m > 0$, the integer $p^{e_m - f_m} u$ is divisible by p, while v is not divisible by p. Hence $p \nmid p^{e_m - f_m} u + v$, so $e_{m+1} = f_m$. Using (7), $f_{m+1} = e_m + 2f_m - e_{m+1} = e_m + f_m$. Since $e_m > f_m \geq 0$, $f_{m+1} = e_m + f_m > f_m = e_{m+1}$. Thus after one step we enter the previous case, so e_t is constant for all sufficiently large t.

Now suppose $e_m = f_m = 0$. Then (7) gives $f_{m+1} = -e_{m+1}$. Both e_{m+1} and f_{m+1} are nonnegative integers, so $e_{m+1} = f_{m+1} = 0$. Thus once $e_m = f_m = 0$, the same equality persists forever, and e_t is constant from that point onward.

It remains to consider the case $e_m = f_m = E > 0$. Since both X_m and Y_m are divisible by p^E, their sum X_{m+1} is divisible by p^E. Hence $e_{m+1} = E + h$ for some integer $h \geq 0$. From (7),

$$f_{m+1} = E + 2E - (E + h) = 2E - h. \tag{8}$$

If $f_{m+1} \neq e_{m+1}$, then at index $m+1$ one of the two strict cases applies, and e_t is eventually constant.

Therefore, the only possible way for e_t not to become eventually constant is for equality $e_t = f_t > 0$ to persist forever from some point onward. Suppose this happens from an index m_0. Write $e_{m_0+r} =$

$f_{m_0+r} = E_r > 0 \qquad (r \geq 0)$. For each $r \geq 0$, since $e_{m_0+r} = f_{m_0+r} = E_r$, there exists an integer $h_r \geq 0$ such that $e_{m_0+r+1} = E_r + h_r$. By (8), $f_{m_0+r+1} = 2E_r - h_r$. Because equality persists, $E_r + h_r = 2E_r - h_r$. Thus $E_r = 2h_r$. Therefore $E_{r+1} = e_{m_0+r+1} = E_r + h_r = E_r + \frac{E_r}{2} = \frac{3E_r}{2}$. By induction, $E_r = \left(\frac{3}{2}\right)^r E_0 = \frac{3^r E_0}{2^r}$ for every $r \geq 0$. Since E_r is an integer and $\gcd(3^r, 2^r) = 1$, this implies $2^r \mid E_0$ for every $r \geq 0$, impossible for a fixed positive integer E_0. Hence equality $e_t = f_t > 0$ cannot persist forever.

Combining all cases, for each prime $p \in \mathcal{P}$, the sequence $v_p(X_m) = e_m$ is eventually constant.

Since $\mathcal{P}$ is finite, there exists an index M such that for every $p \in \mathcal{P}$, the value $v_p(X_m)$ is constant for all $m \geq M$. No prime outside $\mathcal{P}$ divides any X_m. Therefore the full prime factorization of X_m is the same for all $m \geq M$, so X_m is constant for all $m \geq M$.

But by (2), $X_{m+1} = X_m + Y_m$, and Y_m is a positive integer. Hence $X_{m+1} > X_m$ for every $m \geq 0$, contradicting the eventual constancy of X_m.

Therefore no such positive integer N and infinite sequence of positive integers exist.

AI correctness audit. Score: 7/7.

> The proof is complete and correct. It derives the recurrence $X_{m+1}Y_{m+1} = X_m Y_m^2$, confines all prime factors to a fixed finite set, and rigorously shows via p-adic valuation cases that every $v_p(X_m)$ eventually stabilizes. Finite prime support then makes X_m eventually constant, contradicting $X_{m+1} = X_m + Y_m$ with $Y_m > 0$.

AI step audit.

1. **Step 1: absent.** The submission constrains prime support by deriving divisibility such as $X_{m+1} \mid X_m^3$ and a finite prime set, but it never splits X_m and Y_m by their gcd into coprime parts. Thus it uses a materially different mechanism rather than recognizing the specified gcd/coprime-parts step.
2. **Step 2: present.** The submission explicitly isolates the persistent equality case $e_m = f_m = E > 0$, assumes it persists, derives $E_{r+1} = 3E_r/2$, and concludes $2^r \mid E_0$ for all r, which is impossible. This matches the valuation-recursion and fixed-exponent divisibility obstruction.
3. **Step 3: present.** The submission explicitly proves that once $f_m > e_m$, then $e_{m+1} = e_m$ and $f_{m+1} = 2f_m > e_{m+1}$, so the strict valuation inequality propagates forward. It then applies eventual constancy uniformly over the finite prime set P to force eventual constancy of X_m.

Human audit. Auditor B assigned 7/7 ("Sets up a recurrence and goes a slightly different way. Correct") and marked step 1 absent and steps 2 and 3 present ("Step 1: does constrain prime support but differently Step 2: Idea present Step 3: Idea present").