

MathQueue: Reasoning Scheduling Helps Only Within a Narrow Budget, and Answer Extraction Decides Whether It Helps at All

Jonghyun Lee

Artificial Intelligence
Graduate School of Engineering
Yonsei University
devjh@yonsei.ac.kr

Byungjun Kim

Research & Administrative
Information Team
Yonsei University
kbjoz87@yonsei.ac.kr

Abstract

Chain-of-Thought (CoT) prompting treats every reasoning step as equally important, even though solving a mathematical problem depends largely on deciding which step to take first. We therefore formulate reasoning as a scheduling problem. **MathQueue** decomposes a problem into tasks, scores each task by how much subsequent work it enables, how much information it yields, and how likely it is to introduce an error, and then solves the tasks in that order. We evaluate this formulation on GSM8K with two models under a fixed limit on the number of tokens the model may generate, and find that it helps over a far narrower range of conditions than anticipated.

Executing each task in a separate model call, with a verifier that may trigger replanning, performs substantially worse than standard CoT (83% to 72%) at three times the generated tokens and, once the re-sent context of twelve calls is counted, thirteen times the total; the replanning loop accounts for most of the loss. A single-pass variant performs better, but for reasons unrelated to prioritisation: it terminates in the required answer format on every problem, whereas CoT does so on only 87%, and it generates a third fewer tokens with a quarter of the variance in length, yet conditional on an answer being delivered it is never more accurate than CoT.

Retaining this structure but suppressing the ranking leaves accuracy unchanged at loose budgets and costs 19 points at the single budget tight enough to make ordering consequential. Intervening on the scheduler directly rather than on the prompt then locates that effect: the task graphs the model generates are near-chains, so our priority rule reorders only 9 of 100 problems, and the gain therefore comes from committing to a ranked plan rather than from the particular scores we assign to tasks.

Whether any of this constitutes an improvement depends on a factor we had not expected

to matter: how the answer is parsed from a truncated generation. Scoring only well-formed answers favours MathQueue in all eight comparisons; reading the last number instead reverses six of them. We derive an identity explaining the reversal, adjudicate between the rules by manually inspecting every non-compliant generation, and confirm on a second model that what STRICT scoring rewards is the baseline’s format compliance rather than its reasoning. Budget-constrained evaluation of mathematical reasoning must therefore fix and report how the answer is read.

1 Introduction

Large language models solve mathematical word problems by generating a linear chain of intermediate steps (Wei et al., 2022; Nye et al., 2021). This makes an implicit assumption: every step deserves the same treatment. Human problem solving does not work this way. Expert solvers categorise a problem by the principle that governs it, whereas novices attend to surface features (Chi et al., 1981); recognising the governing relation is a high-value step, and multiplying two numbers afterwards is not. When the generation budget is finite, spending it in the wrong order is costly.

A large body of work adds structure to address this: search over branches and thought graphs (Yao et al., 2023a; Besta et al., 2024), sample and vote (Wang et al., 2023), decompose into sub-questions (Zhou et al., 2023; Khot et al., 2023), interleave reasoning with external actions (Yao et al., 2023b), or refine a draft with model-generated feedback (Madaan et al., 2023). All of them cost more than a single CoT pass, and several measure that cost carefully: Wang et al. (2023) equalise sample counts, Yao et al. (2023a) compare token consumption against 100 independent CoT trials, and Besta et al. (2024) fix the number of thoughts to equate cost. In every case, however, the budget is matched *post hoc* rather than imposed at decode time as a

constraint the method must plan against, and no method is scored on what it has produced when the budget is exhausted.

We take the scheduling view literally: a solver should enumerate candidate reasoning tasks, estimate each task’s priority, and execute in priority order subject to dependencies. We then ask, under a controlled token budget, whether this helps.

The answer depends on a decision we had not initially considered part of the method at all: how the final answer is read from a possibly truncated generation. Our contributions are:

1. We cast mathematical reasoning as priority-scheduled traversal of a reasoning-task directed acyclic graph (DAG) (§3), with an interpretable priority functional and a balance operator over reasoning types.
2. We report a **negative result**: a multi-turn reasoning queue with verification and dynamic replanning degrades GSM8K accuracy by 11 points at $\approx 3\times$ the generated and $\approx 13\times$ the total token cost, and an ablation localises the loss in the replanning loop (§6.1).
3. We identify a **plan-prefix floor**: single-pass scheduling spends $\Lambda \approx 127$ median tokens stating its plan before solving, so below $B \approx \Lambda$ it collapses to near-zero accuracy under every scoring rule (§6.3).
4. We separate **ordering from structure** with an ablation that plans without ranking, and find that ordering is beneficial only within the window in which the budget binds and the plan fits (§6.5). Intervening on Eq. 5 directly then **localises** that benefit: because generated task graphs are near-chains, it derives from committing to a ranked plan rather than from our particular task scores (Appendix F).
5. We show that for $B \geq 256$ the **extraction rule can determine the sign** of the comparison, reversing six of eight comparisons, and give an exact identity explaining why (§4, §6.4). We then **adjudicate** between the candidate rules by manual inspection of every non-compliant generation, which yields a validated truncation-aware extractor (Appendix D).
6. We establish **what the advantage consists of**: on a second model whose CoT baseline

already complies on 99% of problems it disappears and all three extractors agree, confirming that STRICT-scored gains track format compliance; and **budget-matched self-consistency** is either far worse than Math-Queue or equal to it at $3.8\times$ the generated tokens (Appendices G and H).

2 Related Work

Structured prompting. CoT (Wei et al., 2022) and scratchpads (Nye et al., 2021) externalise intermediate computation. Least-to-Most (Zhou et al., 2023) identifies all sub-problems in one pass and solves them in the easy-to-hard order fixed at decomposition time; Decomposed Prompting (Khot et al., 2023) relaxes this to recursive and other non-linear structures, generating each sub-question conditioned on previous answers. Both therefore order sub-problems by a rule, but neither rule refers to a generation budget. Kim et al. (2024) go further and dispatch a planned task DAG in parallel, optimising latency rather than what is produced under a token cap. In our notation these methods realise a traversal of the dependency graph that is unordered with respect to any explicit cost, whereas Math-Queue selects the linear extension that is greedy with respect to a stated priority functional.

Search and self-correction. Tree- and graph-structured search (Yao et al., 2023a; Besta et al., 2024) and self-consistency (Wang et al., 2023) improve solution quality by exploring several candidate paths, at a token cost that is a multiple of a single chain. Self-Refine (Madaan et al., 2023) adds iterative feedback from the model itself, but Huang et al. (2024) show that in the *intrinsic* setting, where no external feedback is available, accuracy on reasoning benchmarks typically *degrades* rather than improves, and that previously reported gains rely on oracle labels. Our verifier is intrinsic in exactly this sense, and our negative result is consistent with theirs: on the problems where it triggers replanning, final accuracy falls below what the same pipeline achieves with the loop disabled.

Efficient reasoning. Recent work studies the cost side of reasoning explicitly, by shortening outputs (Nayab et al., 2026) or allocating test-time compute (Snell et al., 2025); see Sui et al. (2025) for a survey. We contribute a budget-matched comparison in which the manipulated variable is the *order* of reasoning rather than its length or sampling count.

Answer extraction. Evaluating GSM8K-style benchmarks requires parsing a numeric answer out of free-form text. GSM8K marks its gold answers with a ##### delimiter and its reference implementation credits a completion only when that marker is present (Cobbe et al., 2021), which is the rule we call STRICT. Downstream practice relaxes it: Kojima et al. (2022) fall back to the last number when the answer trigger is absent, and the widely used harness of Biderman et al. (2024) offers both filters as interchangeable options, treating the choice as a neutral implementation detail. Scores and even model rankings are known to be sensitive to such surface choices (Sclar et al., 2024; Biderman et al., 2024). We identify a specific mechanism behind these reversals: once generations are truncated, the fallback systematically credits whichever method failed to deliver an answer.

3 MathQueue

3.1 Budget-constrained solving

Let $q \in \mathcal{Q}$ be a problem with gold answer $y^*(q)$, and let π_θ be an autoregressive model over token alphabet Σ . A *solver* produces a completion $u \in \Sigma^*$ from which an extractor recovers a prediction. Decoding is greedy ($T = 0$) under a completion budget $B \in \mathbb{N}$, so the realised output is the prefix $u_{1:\min(|u|, B)}$.

3.2 Reasoning task graph

Given q , the model emits reasoning tasks $\mathcal{T} = \{t_1, \dots, t_k\}$ with $3 \leq k \leq 6$ and a dependency relation $E \subseteq \mathcal{T} \times \mathcal{T}$, forming a directed graph $G = (\mathcal{T}, E)$ with parents $\text{pa}(t_i) = \{t_j : (t_j, t_i) \in E\}$. Each task carries a type $\tau(t_i)$ drawn from $\mathcal{K} = \{\text{PLN}, \text{CAL}, \text{VER}, \text{OTH}\}$ and a self-estimated feature vector

$$\phi(t_i) = (D_i, I_i, C_i, R_i)^\top \in [0, 1]^4, \quad (1)$$

where D is dependency (downstream work enabled), I is information gain, C is complexity, and R is error risk. Malformed output triggers one repair turn; persistent failure falls back to a fixed four-task chain.

3.3 Priority functional

Tasks are scored by a linear functional with weights $w = (\alpha, \beta, \gamma, \delta)^\top$:

$$P(t_i) = w^\top \phi(t_i) = \alpha D_i + \beta I_i + \gamma C_i + \delta R_i. \quad (2)$$

We fix $w = (0.35, 0.35, 0.10, 0.20)^\top$ a priori; since these are non-negative and sum to one, $P(t_i) \in [0, 1]$. Dependency and information gain dominate so that structure-revealing steps precede execution steps, while complexity is down-weighted so that a long calculation cannot monopolise the queue merely by being long.

3.4 Balance operator

To prevent trajectories consisting only of arithmetic, we track the type distribution of the completed set $\mathcal{D}_s$ after s steps,

$$\rho_s(\kappa) = \frac{|\{t \in \mathcal{D}_s : \tau(t) = \kappa\}|}{|\mathcal{D}_s|}, \quad \kappa \in \mathcal{K}, \quad (3)$$

and, writing $n_s = |\mathcal{D}_s|$, add a bonus to under-represented types:

$$b_s(t_i) = \begin{cases} \lambda_1 & \tau(t_i) = \text{VER}, n_s \geq 2, \\ & \rho_s(\text{CAL}) > \theta_c \vee \rho_s(\text{VER}) < \theta_v, \\ \lambda_2 & \tau(t_i) = \text{PLN}, n_s \geq 3, \\ & \rho_s(\text{PLN}) < \theta_p, \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

with $\lambda_1 = 0.25$, $\lambda_2 = 0.20$, $\theta_c = 0.55$, $\theta_v = 0.15$, $\theta_p = 0.10$. When both clauses apply the maximum is taken, so $P(t_i) + b_s(t_i) \in [0, 1 + \lambda_1]$. This is a reasoning-domain analogue of load balancing across resource classes.

3.5 Selection rule

Let $\mathcal{F}_s = \{t_i \in \mathcal{T} \setminus \mathcal{D}_s : \text{pa}(t_i) \subseteq \mathcal{D}_s\}$ be the ready frontier. The scheduler executes

$$t^{(s)} = \arg \max_{t_i \in \mathcal{F}_s} [P(t_i) + b_s(t_i)], \quad (5)$$

falling back to $\mathcal{T} \setminus \mathcal{D}_s$ when $\mathcal{F}_s = \emptyset$, and halts after $S_{\max} = 6$ steps or when the queue empties.

Proposition 1 (Schedule validity). *If G is acyclic and every dependency reference resolves to a task in $\mathcal{T}$, then $\mathcal{F}_s \neq \emptyset$ whenever $\mathcal{D}_s \subsetneq \mathcal{T}$, and the sequence produced by Eq. 5 is a linear extension of G .*

Equation 5 is a list-scheduling rule in the sense of Graham (1966), and Proposition 1 restates a standard fact: a finite non-empty DAG restricted to $\mathcal{T} \setminus \mathcal{D}_s$ always contains a source, so greedy selection from the ready set yields a topological order (Kahn, 1962). We state it because the antecedent is consequential in practice: because task graphs are

model-generated, a cycle or an unresolved identifier triggers the fallback branch and the guarantee no longer holds, so Proposition 1 is conditional on well-formed output rather than an invariant of the system.

3.6 Instantiations

The same policy admits realisations with markedly different call complexity M , which is our main experimental variable.

MathQueue-Full (multi-turn). Each task is executed in its own call followed by a verifier call; if the verifier rejects, a replanning call inserts an alternative-strategy task. With S executed steps, V verifier calls and Π replanning calls, $M = 2 + S + V + \Pi$. Empirically $\bar{M} = 12.2$.

MathQueue-Lite (two-call). One call yields $(\mathcal{T}, E, \phi)$; the schedule is materialised offline by Eq. 5; a second call solves in that order. Here $M = 2$, and because $\mathcal{D}_0 = \emptyset$ the balance operator is inactive by Eq. 4.

MathQueue-Inline (single call). The policy is stated in the prompt and the model ranks and executes its own tasks in one generation, so $M = 1$.

4 Budget, Delivery, and Extraction

Our results depend on separating three factors that raw accuracy conflates: whether the reasoning is correct, whether an answer is *delivered* within the budget, and how the extractor behaves when it is not.

Definition 1 (Delivery length). *For solver π and problem q , write $u = u^\pi(q)$ for the unconstrained completion. The delivery length $L^\pi(q)$ is the smallest ℓ such that $u_{1:\ell}$ contains a well-formed final-answer marker, and $L^\pi(q) = \infty$ if the generation never emits one.*

Assumption 1 (Prefix consistency). *Under greedy decoding, imposing a budget truncates rather than alters the trajectory: $u^{\pi,B}(q) = u_{1:\min(|u|,B)}^{\pi,\infty}(q)$.*

We contrast three extractors applied to the *same* generation. STRICT accepts only a well-formed marker carrying a number. LENIENT prefers such a marker but otherwise falls back to the last number appearing anywhere in the text; this is the rule used by our pipeline and corresponds to the last-number filter of Biderman et al. (2024), whereas STRICT corresponds to their marker-only filter and to the reference implementation of Cobbe et al. (2021).

Neither rule is satisfactory, because each answers a different question: STRICT asks whether the solver delivered, LENIENT asks what number it last wrote. A generation can lack a marker for two reasons, and the fallback is trustworthy in only one. If it *ran to completion* without the required line, the solver has an answer and merely failed to format it, so its trailing number is that answer. If it was *cut off*, the trailing number is whatever quantity was being manipulated at truncation and does not constitute an answer. We therefore define a third rule, GATED, which accepts a marker and otherwise falls back to the last number only when the provider reports that generation stopped on its own. Appendix D validates the distinction manually: the fallback is correct on 36 of 36 completed generations and on only 62 of 381 truncated ones, so GATED retains the rescues that are justified and discards those that are incidental.

Under STRICT, a solver scores only when it both delivers within the budget and is correct, so writing $\mathcal{C}^\pi(q)$ for the event that the delivered answer is correct,

$$\text{Acc}_{\text{STRICT}}^\pi(B) = \Pr_q[\mathcal{C}^\pi(q) \wedge L^\pi(q) \leq B], \quad (6)$$

which is non-decreasing in B . STRICT is thus the product of a *delivery rate* $\Pr[L^\pi \leq B]$ and a *compliant accuracy* $\Pr[\mathcal{C}^\pi \mid L^\pi \leq B]$; reporting only the product lets a formatting advantage read as a reasoning advantage, so we report both factors (§6.5). LENIENT adds a *rescue term*: when no answer is delivered, the fallback may still return the gold value by chance, for instance because the running computation happened to end on it. Writing

$$\varepsilon^\pi(B) = \Pr_q[L^\pi(q) > B \wedge \text{fallback correct}] \geq 0, \quad (7)$$

we have $\text{Acc}_{\text{LENIENT}}^\pi(B) = \text{Acc}_{\text{STRICT}}^\pi(B) + \varepsilon^\pi(B)$, while GATED restricts that rescue term to untruncated generations,

$$\text{Acc}_{\text{GATED}}^\pi(B) = \text{Acc}_{\text{STRICT}}^\pi(B) + \varepsilon_{\text{fmt}}^\pi(B), \quad (8)$$

and therefore the following identity for the gap $\Delta(B)$ between two solvers.

Proposition 2 (Extraction identity). *For any two solvers π_1, π_2 , writing $\Delta_{(\cdot)}(B) = \text{Acc}_{(\cdot)}^{\pi_2}(B) - \text{Acc}_{(\cdot)}^{\pi_1}(B)$,*

$$\Delta_{\text{LENIENT}}(B) = \Delta_{\text{STRICT}}(B) - (\varepsilon^{\pi_1}(B) - \varepsilon^{\pi_2}(B)). \quad (9)$$

Equation 9 states that LENIENT scoring credits each solver in proportion to how often it *fails* to deliver. If one method is systematically more compliant, LENIENT scoring is biased against it by exactly the difference in rescue rates, and the measured sign may reverse. Crucially, ε depends on incidental properties of the text, including material echoed from the prompt, and is therefore not a stable property of the method.

MathQueue-Inline also pays a structural cost: it must state its ranked plan before solving, so its delivery length decomposes as $L^{\text{MQ}} = \Lambda + R$, where Λ is the plan prefix. This imposes a floor: if $B < \Lambda$ then no answer is delivered and $\text{Acc}_{\text{STRICT}}^{\text{MQ}}(B) = 0$, irrespective of the quality of the underlying reasoning.

5 Experimental Setup

Data and models. We sample $N = 100$ problems uniformly without replacement (seed 42) from the GSM8K test split (Cobbe et al., 2021) and use Solar Pro 3 (solar-pro3; hereafter Solar) through an OpenAI-compatible endpoint at temperature 0. All methods see identical problems. To test whether the effects are a property of one model we repeat the central comparison on gpt-oss-20b (OpenAI, 2025) served locally (Appendix G); that model streams a hidden reasoning channel whose tokens are charged against the same cap, so we log them separately.

Prompt versions. Both prompts originally illustrated the output format with a literal example, FINAL_ANSWER: 42. We found that truncated CoT generations re-quote this instruction, causing LENIENT to return 42 on 10 of 100 problems at $B = 256$. We therefore re-ran the entire sweep with the numeral removed. We report both versions, written LEGACY and CLEAN, because the difference between them is part of the finding. Appendix I summarises every prompt.

Conditions. E1 compares CoT, MathQueue-Full, and MathQueue-Full without replanning at $B = 2048$. E2 compares CoT and MathQueue-Inline at $B = 2048$. E3 sweeps budgets 128, 256, 512, 1024 and 2048 for both methods and both prompt versions. E4 re-scores every stored generation under all three extractors, so the extractor is isolated from the model. E5 adds an ablation, PLAN, which keeps MathQueue-Inline’s decompose–list–solve–verify structure and its final-answer format but is

explicitly instructed not to rank or prioritise the tasks; it is a prompt-level manipulation, differing from MathQueue-Inline in the ranking instruction and from CoT in having any explicit structure at all. E6 replaces that manipulation with an intervention on Eq. 5 itself: one decomposition per problem, two linear extensions of the same graph, identical solve prompt. E7 repeats E2, E4 and E5 on the second model at $B \in \{256, 512, 2048\}$. E8 adds a budget-matched self-consistency baseline (Wang et al., 2023) in which $k = 3$ samples are each capped at $\lfloor B/k \rfloor$ tokens, so the ensemble cannot outspend a single CoT pass of B . E6–E8 are reported in Appendices F–H.

Metrics. We report accuracy under each extractor, mean completion tokens $\bar{c}$, mean calls $\bar{M}$, latency, truncation rate, *marker rate*, the share of generations from which a well-formed answer can be read, and *compliant accuracy*, accuracy restricted to those generations. Appendix J lists the decoding settings and the contents of the per-problem logs.

Significance. Paired comparisons use the exact conditional McNemar test (McNemar, 1947; Dietterich, 1998). With discordance counts n_{10}, n_{01} and $m = n_{10} + n_{01}$, the two-sided p -value is

$$p = \min \left(1, 2^{1-m} \sum_{i=0}^{\min(n_{10}, n_{01})} \binom{m}{i} \right). \quad (10)$$

6 Results

6.1 E1: agentic scheduling degrades accuracy

The fully agentic queue is harmful. At $B = 2048$ on the LEGACY prompt under LENIENT scoring, MathQueue-Full reaches 72.0 against CoT’s 83.0, a loss of 11 accuracy points, while issuing $\bar{M} = 12.2$ calls per problem ($p = 0.019$). Removing dynamic replanning recovers 8 of the 11 lost points (80.0; against CoT $p = 0.61$), which localises most of the degradation in the verify-and-replan loop rather than in prioritisation.

The cost of the loop is larger than generated tokens alone suggest. MathQueue-Full produces $2.9\times$ as many completion tokens as CoT, but each of its 12.2 calls re-sends the accumulated context, so it consumes $35\times$ the *input* tokens (5859 against 167) and $12.6\times$ the total (7013 against 558). Under any pricing scheme that charges for input, the orchestration rather than the reasoning dominates the cost; Appendix E reports both sides for every condition.

The replanning subset makes this precise. Let $\mathcal{R} = \{q : \Pi(q) > 0\}$; here $|\mathcal{R}| = 16$ with 2.8 insertions on average. Restricted to $\mathcal{R}$, MathQueue-Full attains 43.8%, CoT 62.5%, and the identical pipeline with the loop disabled 68.8%. Because the no-replan variant differs only in whether the loop may be invoked, the verifier is not merely selecting difficult problems; it degrades trajectories that the same scheduler would otherwise have solved.

6.2 E2: single-pass scheduling is compact and compliant

Compiling the policy into one generation removes the orchestration cost. At $B = 2048$ on the CLEAN prompt, MathQueue-Inline uses 33.4% fewer completion tokens than CoT (269 against 404) and is markedly more predictable: the standard deviation of completion length falls from 339 to 84 tokens, a $4\times$ reduction, and the maximum from 2048 (i.e. reaching the cap) to 549 (Appendix B). It also delivers an answer far more reliably. Even at this budget, where only 2% of CoT generations are truncated, CoT emits a well-formed final answer on 87% of problems against MathQueue’s 100%; the remainder terminate in commentary on the requested format without producing the required line. Accuracy is deferred to Table 1, since it depends on the extractor.

Prompt length partly offsets the saving: the scheduling instructions lengthen the prompt from 165 to 281 tokens, so the reduction in *total* tokens is only 3.3% ($569 \rightarrow 550$). The saving is concentrated on the generated side, which dominates latency and, under most pricing schemes, cost.

6.3 E3: the plan-prefix floor

Sweeping the budget reveals a cost that the unconstrained comparison hides. MathQueue-Inline must state its ranked plan before solving; we locate the plan/solution boundary in all 100 traces and estimate Λ at a median of 127 tokens (mean 141, IQR 111–151). At $B = 128$ the plan alone consumes the entire budget on 50% of problems, and the plan-prefix floor becomes binding: MathQueue reaches 0% (STRICT) or 5% (LENIENT) against CoT’s 7% and 27%. This is the one regime in which both scoring rules agree that scheduling is harmful, and it constitutes a substantive limitation of the method: the scheduling overhead must be amortised before ordering can be of any benefit.

B	STRICT			GATED			LENIENT		
	CoT	MQ	Δ	CoT	MQ	Δ	CoT	MQ	Δ
128	7	0	−7	8	0	−8	27	5	−22
256	42	51	+9	49	51	+2	66	54	−12
512	69	83	+14	78	83	+5	83	84	+1
1024	78	85	+7	86	85	−1	87	85	−2
2048	77	84	+7	88	84	−4	88	84	−4

Table 1: E3/E4 accuracy (%) on the CLEAN prompt under the three extractors. STRICT and LENIENT disagree on the sign at three of the four budgets with $B \geq 256$; under the validated GATED rule the gap is within 5 points of zero throughout and never significant ($p \geq 0.27$). LEGACY results are in Appendix A and the corresponding curves in Appendix B.

6.4 E4: extraction decides the sign

For $B \geq 256$ the conclusion depends entirely on how the answer is read (Table 1). Under STRICT MathQueue leads at every budget in this range, in all eight comparisons across both prompt versions, by +3 to +14 points; under LENIENT it trails in most, by as much as −12. The two rules disagree on the sign at six of those eight combinations, agreeing only where the LENIENT margin is within 6 points of zero.

Which rule to prefer can be settled empirically rather than by convention, because each makes a verifiable claim about individual generations. We manually inspected every generation carrying no marker and asked whether its trailing number is nonetheless the answer the solver had reached: pooled over budgets and methods, the fallback is correct on 36 of 36 generations that ran to completion and on 62 of 381 that were truncated (Appendix D). Under the resulting GATED rule of Eq. 8, MathQueue’s advantage falls from between +7 and +14 points to between −4 and +5, and is not significant at any budget. Most of what STRICT credited to scheduling was CoT being penalised for correct answers in the wrong format.

Proposition 2 accounts for this exactly. Taking π_1 to be CoT and π_2 MathQueue, so that Δ is signed as in Table 1: at $B = 2048$ on the CLEAN prompt, $\varepsilon^{\text{CoT}} = 88 - 77 = 11$ points while $\varepsilon^{\text{MQ}} = 84 - 84 = 0$, so Eq. 9 gives $\Delta_{\text{LENIENT}} = +7 - (11 - 0) = -4$, which is the value we measure. The rescue term is large for CoT and zero for MathQueue for a reason evident in the delivery rates (Appendix C): above $B = 256$ MathQueue delivers a marker on every problem, so the fallback has nothing to rescue, whereas CoT

plateaus at 87% even when it is not truncated. At $B = 256$ the rescued items can be counted directly: exactly 24 CoT generations deliver no answer yet have the gold value as their trailing number, matching $\epsilon^{\text{CoT}} = 66 - 42 = 24$ points, against 3 for MathQueue. Appendix D presents one such case, in which CoT computes the correct profit and then expends its remaining budget on commentary about the required output format instead of stating the answer.

The instability is not confined to the choice of extractor. Under LENIENT scoring at $B = 256$, deleting the numeral from the format example alone moves the gap from +6.0 to -12.0, an 18-point change produced by a single token of prompt text with the model, problems and method held fixed. Under STRICT scoring the same change moves the gap only from +13.0 to +9.0.

6.5 E5: ordering versus structure

MathQueue differs from CoT in two respects simultaneously: it imposes a plan, and it orders that plan by priority. The experiments reported so far therefore cannot attribute the effect to either component. Table 2 separates them.

At loose budgets the ordering policy contributes nothing. At $B = 2048$, PLAN reaches 85.0 against MathQueue’s 84.0 ($p = 1.000$), and at $B = 512$ the gap is +2.0 in MathQueue’s favour ($p = 0.754$). Both variants deliver a well-formed answer far more reliably than CoT (93–100% against 75–87%), and both exceed it under STRICT scoring, by +7 to +14 points. Any advantage that scheduling confers in the unconstrained regime is therefore attributable to *structure*, that is, to committing to a task list and terminating in the required format, rather than to the priority functional.

At $B = 256$ this relationship reverses. Ranking is worth +19.0 points over the same structure left unranked ($p = 0.003$), the largest and most significant effect we measure for the ordering policy, and it is the one effect in the paper that is invariant to the extraction rule: the same +19.0 holds under GATED ($p = 0.003$). Unranked planning is in fact *inferior* to standard CoT in this regime (-10.0 STRICT, -17.0 GATED): it incurs the cost of writing a plan and then executes it in an arbitrary order, so it is truncated on 66% of problems and delivers an answer on only 34%, against MathQueue’s 44% and 57%.

This is precisely the window that the analysis of §4 predicts. Ordering can matter only when part of

B	Method	STRICT	GATED	LENIENT	Mark.	Compl.	$\bar{c}$
256	CoT	42.0	49.0	66.0	44	95.5	223
	PLAN	32.0	32.0	43.0	34	94.1	240
	MQ	51.0	51.0	54.0	57	89.5	229
512	CoT	69.0	78.0	83.0	75	92.0	317
	PLAN	81.0	81.0	81.0	93	87.1	300
	MQ	83.0	83.0	84.0	98	84.7	268
2048	CoT	77.0	88.0	88.0	87	88.5	404
	PLAN	85.0	85.0	85.0	100	85.0	300
	MQ	84.0	84.0	84.0	100	84.0	269

Table 2: E5 ablation (CLEAN prompt). PLAN keeps the structure but drops the ranking; Compl. is accuracy among generations that delivered a marker. Ranking is worth +19.0 points over unranked structure at $B = 256$ under both STRICT and GATED ($p = 0.003$), but +2.0 ($p = 0.754$) at 512 and -1.0 ($p = 1.000$) at 2048. MathQueue has the lowest compliant accuracy at every budget.

the trajectory will be truncated, which requires B to bind, and it can help only when the plan itself fits, which requires $B > \Lambda$. Outside ($\Lambda, \approx 512$) the priority functional is inert. We treat this window as a preliminary estimate: it rests on one binding budget, one dataset and one seed, and its upper edge is bounded only between 256 and 512. Intervening on Eq. 5 directly, by reordering an identical generated graph rather than by altering the prompt, localises the mechanism further: the graphs the model produces are near-chains, so within this window ranking operates chiefly by committing the model to one ordered plan rather than by selecting among genuinely competing tasks (Appendix F).

The compliant accuracies in Table 2 delimit what structure provides. At every budget MathQueue is the *least* accurate of the three methods conditional on having delivered an answer (84.0–89.5% against CoT’s 88.5–95.5%); its STRICT advantage is produced entirely by delivering answers at all. Scheduling in this setting improves the reliability with which a solution is emitted, not the quality of the solution.

7 Analysis

Structure yields delivery; ordering yields time.

The ablation separates the two effects cleanly. Structure is responsible for the robust, budget-independent gains: a 100% marker rate against CoT’s 87%, 33% fewer completion tokens, and a 4× tighter length distribution. The priority functional adds nothing to this once the budget is loose, and contributes substantially (+19 points) precisely

where the budget compels the solver to choose what to address first. Both components of the scheduling formulation are substantive, but they operate in different regimes, and reporting only unconstrained accuracy would have revealed neither.

Why lenient scoring inverts the ranking. A last-number fallback is a reasonable heuristic for a completed generation and a poor one for a truncated generation, in which the last number is whatever quantity the model was manipulating when the budget ran out. Manual inspection quantifies the gap: the fallback is correct on 36 of 36 completed non-compliant generations and on 62 of 381 truncated ones. By Eq. 9 the unconditional form of the heuristic therefore awards credit to exactly the method that more often fails to deliver, which is why we adopt the finish-conditioned rule GATED as our primary measure and report all three. STRICT is not free of judgement either, since it penalises correct reasoning that terminates in the wrong format; GATED separates the two failure modes that STRICT and LENIENT each conflate.

Verification is not free. The verifier was intended to implement the dynamic-replanning component of the scheduling formulation, and it is precisely the component that fails: on $\mathcal{R}$, disabling it raises accuracy from 43.8% to 68.8%. This is consistent with evidence that LLMs cannot reliably self-correct reasoning without external signals (Huang et al., 2024). A scheduler that reorders work is inexpensive; one that also adjudicates correctness inherits the verifier’s error rate.

Evaluation under a budget. Comparisons that fix a generation budget are becoming common as test-time compute becomes a controlled variable, whether the budget is enforced by truncation or by intervening on the decoder (Snell et al., 2025; Muennighoff et al., 2025; Sui et al., 2025). Our results suggest they are not identifiable unless the answer-extraction rule is fixed and reported, because truncation makes the extractor a co-determinant of the score. We recommend reporting the marker rate and the compliant accuracy alongside accuracy, and conditioning any fallback on the finish reason. The second model supports the diagnosis: where the baseline already complies on 99% of problems, all three extractors agree to within a point and no method ordering survives (Appendix G).

8 Conclusion

We formalised mathematical reasoning as priority-scheduled traversal of a reasoning-task DAG and evaluated it under matched token budgets. The agentic realisation performs worst: -11 accuracy points at triple the generated and thirteen times the total token cost, with the replanning loop responsible for most of the loss. The single-pass realisation carries a fixed plan-prefix cost that causes it to collapse below $B \approx \Lambda$. Between that floor and the budget at which truncation ceases to bind, priority ordering is worth $+19$ points over the same plan left unranked under both scoring rules that apply there; outside that window it confers no benefit, and every remaining gain derives from structure, which raises delivery to 100% at a third fewer tokens without improving accuracy conditional on delivery. Whether any of this manifests as an accuracy improvement depends on the answer-extraction rule, which reverses the sign of six of eight comparisons and whose choice we settle by manual validation rather than by convention. The scope over which the scheduling formulation is useful is thus narrower than it initially appears, and the methodological implication is broader: budget-constrained evaluation of mathematical reasoning must fix and report how the answer is read.

Limitations

Our study is deliberately narrow. All experiments use one dataset (GSM8K); grade-school word problems may understate the value of scheduling on problems where strategy search is the difficulty, and we have not verified that these effects transfer to MATH (Hendrycks et al., 2021) or to proof-oriented benchmarks. The two models we use differ in scale, provider and reasoning interface simultaneously, so the second serves to test whether a mechanism reappears, not to establish a population of models. With $N = 100$ per condition, individual budget points are underpowered: only 4 of the 10 STRICT comparisons reach $p < 0.05$, and of the eight at $B \geq 256$ where MathQueue leads, only two do. Our claim of a consistent STRICT advantage therefore rests on the sign being positive in 8 of 8 comparisons rather than on any single test, and under GATED the corresponding advantage is within 5 points of zero. We use a single seed and greedy decoding for the main comparisons, so we cannot separate method effects from decoding variance.

Several measurements are indirect. The plan prefix Λ is estimated by locating a section header and converting a character offset into tokens using a per-example ratio, not by counting tokens directly. Assumption 1 may be violated by provider-side nondeterminism. Stored traces are capped at 4000 characters, which affected 11 generations whose STRICT scores are therefore lower bounds. E1 pre-dates the prompt correction and can only be scored LENIENT, because MathQueue-Full’s answer arrives through a separate finalisation call rather than a single generation; we did not re-run it under the CLEAN prompt, so its -11 point result is not directly comparable to Table 1.

The method itself is only partly validated. The features ϕ are self-reported by the model and never checked against ground-truth step importance, the weights w were fixed a priori and never tuned, and we did not ablate the balance operator, which is inactive in MQ-Lite by construction. The E5 ablation isolates ordering from structure at the prompt level, so PLAN and MathQueue-Inline differ slightly in prompt length (250 vs. 281 tokens) as well as in ranking; the graph-level intervention of Appendix F removes that confound but can only reorder the graphs the model happens to generate, which are near-chains, so it bounds the effect of the selection rule on *these* graphs rather than of graph-based scheduling in general. Both were run at three budgets, so the window boundary between 256 and 512 is coarse. Our search baseline is budget-matched self-consistency at a single budget split (Appendix H); tree search and verifier-guided decoding remain untested.

Ethics Statement

This work evaluates arithmetic reasoning on a public benchmark (GSM8K) and involves no human subjects, no personal data, and no user-facing generation. The principal risk we identify is over-reliance on truncated model reasoning: under a tight budget both methods emit confident answers with incomplete justification, and part of MathQueue’s advantage is that it *always* produces a clean final answer, which may make incomplete reasoning appear more trustworthy than it is. Our finding that a last-number fallback can invert a method ranking also implies that published gains on truncated-generation benchmarks may partly reflect parsing heuristics; we release per-problem logs so that any reported number can be recom-

puted under an alternative rule. Cost figures come from provider-returned token counts.

References

- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefer. 2024. [Graph of thoughts: Solving elaborate problems with large language models](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.
- Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sid Black, Jordan Clive, Anthony DiPofi, Julien Etzaniz, et al. 2024. [Lessons from the trenches on reproducible evaluation of language models](#). *Computing Research Repository*, arXiv:2405.14782.
- Micheline T. H. Chi, Paul J. Feltovich, and Robert Glaser. 1981. [Categorization and representation of physics problems by experts and novices](#). *Cognitive Science*, 5(2):121–152.
- 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. 2021. [Training verifiers to solve math word problems](#). *Computing Research Repository*, arXiv:2110.14168.
- Thomas G. Dietterich. 1998. [Approximate statistical tests for comparing supervised classification learning algorithms](#). *Neural Computation*, 10(7):1895–1923.
- Ronald L. Graham. 1966. [Bounds for certain multiprocessing anomalies](#). *Bell System Technical Journal*, 45(9):1563–1581.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.
- Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. [Large language models cannot self-correct reasoning yet](#). In *International Conference on Learning Representations*.
- Arthur B. Kahn. 1962. [Topological sorting of large networks](#). *Communications of the ACM*, 5(11):558–562.
- Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. [Decomposed prompting: A modular approach for solving complex tasks](#). In *International Conference on Learning Representations*.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2024. [An LLM compiler for parallel function calling](#). In *International Conference on Machine Learning*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. [Large language models are zero-shot reasoners](#). In *Advances in Neural Information Processing Systems*.

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. 2023. [Self-refine: Iterative refinement with self-feedback](#). In *Advances in Neural Information Processing Systems*.

Quinn McNemar. 1947. [Note on the sampling error of the difference between correlated proportions or percentages](#). *Psychometrika*, 12(2):153–157.

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

Sania Nayab, Giulio Rossolini, Marco Simoni, Andrea Saracino, Giorgio Buttazzo, Nicolamaria Manes, and Fabrizio Giacomelli. 2026. [Concise thoughts: Impact of output length on LLM reasoning and cost](#). *Information Sciences*, 757:123939.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. [Show your work: Scratchpads for intermediate computation with language models](#). *Computing Research Repository*, arXiv:2112.00114.

OpenAI. 2025. [gpt-oss-120b & gpt-oss-20b model card](#). *Computing Research Repository*, arXiv:2508.10925.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. [Quantifying language models’ sensitivity to spurious features in prompt design](#). In *International Conference on Learning Representations*.

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

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. [Stop overthinking: A survey on efficient reasoning for large language models](#). *Transactions on Machine Learning Research*.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. [ReAct: Synergizing reasoning and acting in language models](#). In *International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). In *International Conference on Learning Representations*.

A Legacy-prompt results

Table 3 reports the sweep on the LEGACY prompt, whose format example contained the numeral 42. Comparing with Table 1 isolates the effect of that single token: under STRICT scoring the gap changes by at most 8 points and never changes sign for $B \geq 256$, whereas under LENIENT scoring it changes by up to 18 points and changes sign at $B = 256$.

B	STRICT			LENIENT		
	CoT	MQ	Δ	CoT	MQ	Δ
128	20.0	0.0	−20.0	28.0	3.0	−25.0
256	51.0	64.0	+13.0	60.0	66.0	+6.0
512	75.0	81.0	+6.0	83.0	82.0	−1.0
1024	79.0	82.0	+3.0	83.0	82.0	−1.0
2048	77.0	82.0	+5.0	84.0	82.0	−2.0

Table 3: Accuracy (%) on the LEGACY prompt.

B Budget and length curves

Figure 2 plots accuracy against the budget under both boundary extractors, and Figure 1 the completion-length distributions at $B = 2048$.

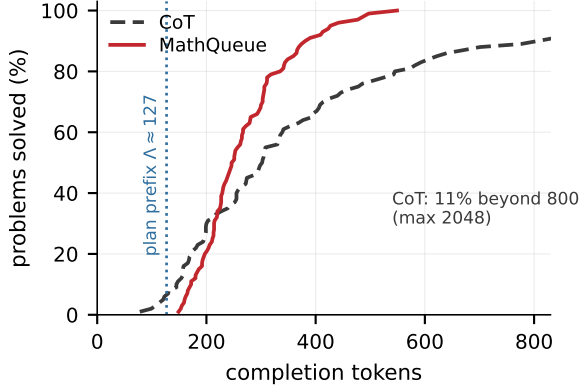


Figure 1: Completion lengths at $B = 2048$ (CLEAN). Scheduling concentrates generation into a narrow band and eliminates the long tail, at the price of a floor near the plan prefix Λ .

B	marker %		trunc. %		$\bar{c}$	
	CoT	MQ	CoT	MQ	CoT	MQ
128	8	0	95	100	127	128
256	44	57	61	44	223	229
512	75	98	20	2	317	268
1024	87	100	7	0	387	265
2048	87	100	2	0	404	269

Table 4: Delivery behaviour on the CLEAN prompt. Marker rate is the share of generations yielding a well-formed answer. Above $B = 256$ MathQueue always delivers, while CoT plateaus at 87% even without truncation.

C Delivery behaviour

Table 4 gives the quantities entering Eq. 9. Two features drive the results of §6.4. First, MathQueue’s marker rate reaches 100% from $B = 1024$ upwards, so its rescue term ε is zero and its STRICT, GATED and LENIENT scores coincide. Second, CoT’s marker rate stops at 87% even where only 2% of its generations are truncated, so its non-delivery is a formatting failure rather than a budget failure, and the rescue term stays large.

D Validating the GATED extractor

We examined every generation in the CLEAN sweep from which STRICT recovers nothing, and recorded whether the trailing number of the text is the value the solver had in fact arrived at. Table 5 reports the outcome. The split is close to categorical: when the model terminated of its own accord, the answer was present but unmarked in all 36 cases, whereas under truncation the trailing number was the gold value in 16% of cases, a rate consistent with a partial computation coinciding with the answer. The

Non-marker generations	n	correct	prec.
Terminated on its own	36	36	1.00
Truncated at the cap	381	62	0.16

Table 5: Manual validation of the last-number fallback, pooled over the five budgets and both methods on the CLEAN prompt. Precision is the share of non-marker generations whose trailing number is the gold answer. Conditioning on the finish reason separates a reliable rescue from an unreliable one, which is what Eq. 8 does.

composition of the non-marker cases follows the same pattern by budget: of CoT’s 56 at $B = 256$, 49 were truncated and 24 carried the gold value; of its 13 at $B = 2048$, 11 did. MathQueue never produces a completed non-compliant generation at any budget.

Figure 3 shows a case that LENIENT rescues and GATED does not.

E Full cost accounting

Study	Method	in	out	total	$\bar{M}$	s
E1, 2048	CoT	167	391	558	1.0	2.9
	MQ-Full	5859	1154	7013	12.2	13.4
	no replan	5183	1021	6204	11.5	12.1
E5, 256	CoT	165	223	388	1.0	1.9
	PLAN	250	240	490	1.0	2.2
	MQ-Inline	281	229	510	1.0	2.1
E5, 2048	CoT	165	404	569	1.0	3.1
	PLAN	250	300	550	1.0	2.4
	MQ-Inline	281	269	550	1.0	2.8

Table 6: Mean prompt, completion and total tokens per problem, calls $\bar{M}$ and wall-clock seconds. The budget B caps completion tokens only, so input cost is unconstrained by construction, which is what makes the multi-turn realisation costly.

Table 6 reports both sides of the cost. The budget constrains completion tokens, so a method that issues many calls pays an input cost the budget does not see: MathQueue-Full re-sends the accumulated transcript on each of its 12.2 calls and consumes $35\times$ CoT’s input tokens. Single-pass scheduling avoids this entirely; its longer prompt adds a fixed 116 tokens, so at $B = 2048$ its total cost is comparable to CoT’s (550 against 569) despite generating a third fewer tokens.

F E6: intervening on the selection rule

The E5 ablation manipulates the prompt, so it cannot distinguish the priority functional from the instruction to write a plan. We therefore intervene on

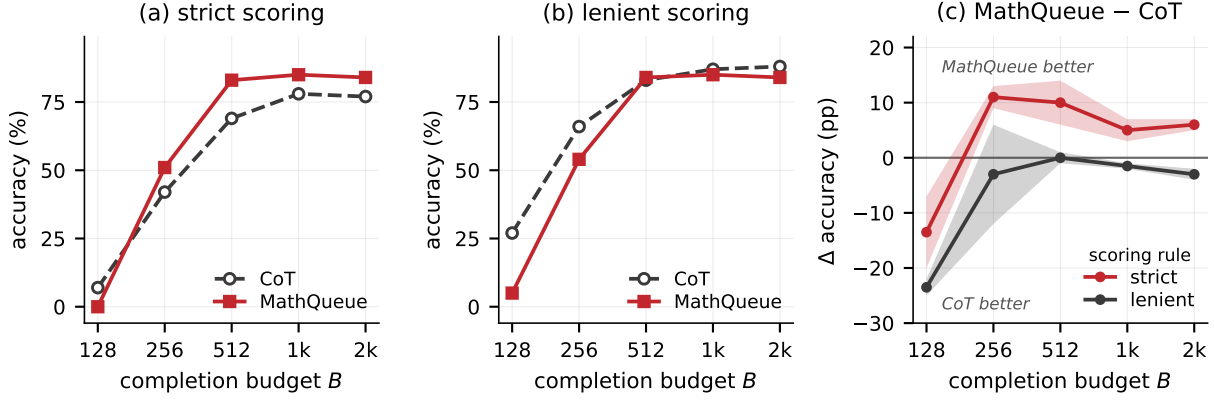


Figure 2: Accuracy against completion budget. (a) Under STRICT extraction MathQueue dominates CoT for $B \geq 256$. (b) Under LENIENT extraction the ordering reverses everywhere except at $B = 512$, where the two curves nearly coincide. (c) The gap $\Delta(B)$ for both prompt versions and both extractors: the extractor (colour) separates the curves cleanly, while the prompt version (solid vs. dotted) does not. At $B = 128$, where the plan prefix exceeds the budget, both rules agree that scheduling is harmful.

Problem (gold = 8): *Rose bought five dozens of eggs for \$2.40 a dozen. She will sell it for \$1 for 3 eggs. How much will be Rose’s profit?*

CoT, truncated at $B = 256$: ... *Profit = revenue – cost = \$20 – \$12 = \$8.00. Thus profit = \$8.00. But we need to show step-by-step reasoning. Also ensure final answer is numeric, maybe with dollar sign? The format says*

STRICT and GATED: nothing delivered $\Rightarrow$ wrong.

LENIENT: trailing number 8.00 $\Rightarrow$ correct.

MathQueue, complete in 242 tokens: ... *4. Profit: \$20.00 – \$12.00 = \$8.00, followed by the required final-answer line.*

All three rules: correct.

Figure 3: One of the 24 problems at $B = 256$ where LENIENT scoring rescues a CoT generation that never delivered an answer. Whether the rescue succeeds depends on which number the truncation happens to fall after.

Eq. 5 itself. For each problem we generate the task graph once and then execute it twice at $B = 512$, once in the order the priority functional selects and once in the order the model emitted, with an otherwise identical solve prompt from which all ranking language has been removed. Everything except the order is thus held fixed.

The intervention is possible on fewer problems than expected. Analysing the 100 graphs shows that 54 are chains admitting exactly one valid order, the median number of linear extensions over all graphs is 1, and the largest ready set encountered at any point is 3. The priority order coincides with the model’s emitted order on 91 problems, and on 6 of the 9 remaining ones the disagreement is not attributable to the priority scores. Re-decomposition

at execution time reproduced two genuinely distinct orders on 5 of the 9 pairs, and in every case they differ by a single transposition of adjacent tasks.

The outcome is unchanged on all of them. Both orders yield 44.4% STRICT and 55.6% GATED accuracy on the paired subset, the predictions agree token-for-token on the reordered pairs, and no problem changes from correct to incorrect or the reverse. This does not contradict §6.5; it locates the +19 points there. Within the window where the budget binds, what the ranking supplies is a commitment to a single ordered plan, and on graphs of this depth that commitment does not depend on which of the admissible orders is chosen. A benefit from the particular scores ϕ would require problems whose dependency structure leaves genuine choice, which GSM8K decompositions rarely do.

G E7: transfer to a second model

We repeat the E5 comparison on gpt-oss-20b, a model that differs from Solar in scale, provider and interface, and that streams a hidden reasoning channel whose tokens count against B (88 mean reasoning tokens for MathQueue against 58 for CoT at $B = 256$). Table 7 reports the result.

Two of the paper’s claims reappear and one does not. The plan-prefix floor reappears, displaced upwards: because reasoning tokens are billed against the cap, the effective budget is smaller, and $B = 256$ now lies below the floor, where MathQueue reaches 25% against CoT’s 66% and is truncated on 74% of problems. The value of ranking over

B	Method	STRICT	GATED	LENIENT	Mark.	Tr.
256	CoT	66	66	72	73	27
	PLAN	18	18	22	20	80
	MQ	25	25	32	26	74
512	CoT	86	86	86	99	1
	PLAN	86	86	86	95	5
	MQ	82	82	85	93	7
2048	CoT	85	85	85	100	0
	PLAN	90	90	90	100	0
	MQ	88	88	88	100	0

Table 7: E7: gpt-oss-20b on the same 100 problems and the CLEAN prompt. The baseline complies on 99–100% of problems from $B = 512$, so all three extractors agree and the STRICT advantage of §6.4 does not arise. Hidden reasoning tokens are charged against B , which moves the plan-prefix floor upwards.

unranked structure reappears in the same regime, at +7 points. What does not reappear is the STRICT advantage of §6.4, and the reason is the one our analysis predicts: this model’s CoT baseline already delivers a well-formed answer on 99% of problems at $B = 512$ and 100% at $B = 2048$, leaving no compliance gap for structure to close. Consistently with that, all three extractors agree to within a point at every budget. The STRICT-scored advantage on Solar is therefore not a property of scheduling as such but of the compliance ceiling of the baseline it is measured against.

H E8: budget-matched self-consistency

B	Method	STRICT	GATED	Deliv.	out
512	SC ($k=3$)	45.0	48.0	51	484
	MQ	83.0	83.0	98	268
2048	SC ($k=3$)	84.0	84.0	93	1024
	MQ	84.0	84.0	100	269

Table 8: E8: self-consistency with $k = 3$ samples at temperature 0.7, each capped at $\lfloor B/k \rfloor$ tokens so that the total generation budget matches. Deliv. is the share of problems on which a majority answer could be formed; out is mean completion tokens.

Sampling several trajectories and taking a majority vote is the natural alternative use of a fixed budget. We match the budget by drawing $k = 3$ samples at temperature 0.7, each capped at $\lfloor B/k \rfloor$ tokens (170 at $B = 512$, 682 at $B = 2048$), and take the majority over delivered answers. Table 8 reports the comparison.

The two regimes separate sharply. At $B = 512$ the per-sample cap of 170 tokens is below the

length of a typical solution, so most samples are truncated and no majority can be formed on 49% of problems; self-consistency reaches 45.0% against MathQueue’s 83.0% ($p < 10^{-10}$). At $B = 2048$ each sample has sufficient room, and self-consistency matches MathQueue exactly (84.0% against 84.0%, $p = 1.000$) while generating $3.8\times$ as many tokens and taking $2.9\times$ as long. Within a tight budget, therefore, allocating tokens to one structured trajectory dominates spreading them over several unstructured ones, and once the budget is loose the two are equivalent in accuracy but not in cost. Neither regime provides evidence that sampling recovers what ordering provides.

I Prompts

Decomposition. The model returns JSON containing 3–6 tasks, each with id, description, task_type, depends_on and the four components of ϕ in $[0, 1]$, with instructions not to solve the problem yet and to include at least one VER task.

Inline scheduling. The single-call prompt instructs the model to (i) decompose the problem, (ii) rank tasks by dependency and information gain, stating explicitly that complexity must not dominate the order, (iii) solve in ranked order, (iv) verify once at the end, and (v) avoid long calculations before the governing relation is fixed. It requires a short ranked task list followed by the solution and a final answer line.

CoT baseline. The baseline asks for step-by-step reasoning and the same final-answer format, with no scheduling instructions. In the LEGACY version both prompts ended with the example FINAL_ANSWER: 42; in the CLEAN version the numeral is replaced by a placeholder, which the extractor ignores.

The paragraphs above summarise the prompts; the templates themselves are released verbatim as plain-text files alongside the logs, one per method, so that no wording has to be reconstructed from this description.

J Reproducibility

Models and endpoints. Solar Pro 3 is served as solar-pro3 on the OpenAI-compatible endpoint api.upstage.ai/v1; the provider does not expose an immutable version tag, so every record carries the response identifier and timestamp returned by the API. The second model is openai/

gpt-oss-20b served locally behind an OpenAI-compatible endpoint, whose reasoning_tokens field is logged separately and counted against the same cap. Each run writes a run_meta.json recording the model string, endpoint, budget, seed, prompt version and library versions.

Decoding. Temperature is 0 for all main comparisons, at budgets of 128, 256, 512, 1024 and 2048 completion tokens, identical across methods within a study, with $S_{\max} = 6$ scheduling steps for multi-turn variants. Self-consistency (Appendix H) is the sole exception, at temperature 0.7 with $k = 3$. The budget is imposed as max_tokens at decode time rather than by post hoc truncation.

Problem sample. The 100 problems are drawn from the GSM8K test split by random.Random(42).sample, and the resulting indices are stored with the logs so that the sample can be reproduced or audited without re-running the sampler.

Records. Every problem yields one record containing the prediction, the gold answer, provider prompt/completion/total/reasoning token counts, wall-clock latency, the finish reason, the raw generation, and, for scheduling methods, the full task trace $(\mathcal{T}, E, \phi, \sigma)$ including the emitted order, the priority order and the executed order. Traces are stored uncapped for E6–E8; in the earlier sweeps they were truncated at 4000 characters, which affected 11 generations, as noted in the Limitations.

Offline scoring. All three extractors, the compliant accuracies and every p -value from Eq. 10 are recomputed from the stored generations by a single scoring script, so no number in this paper depends on scoring decisions made at generation time. This is what allows Table 1 to isolate the extractor from the model, and it allows any reader to re-derive the tables under a different extraction rule.