TRIAGE: Evaluating Prospective Metacognitive Control in LLMs under Resource Constraints

Anonymous Authors[1]

Abstract

Deploying language models as autonomous agents requires more than per-task accuracy: when an agent faces a queue of problems under a finite token budget, it must decide which to attempt, in what order, and how much compute to commit to each, all before any execution feedback is available. This is the prospective form of metacognitive control studied for decades in human cognition, yet whether language models possess it remains untested. We introduce TRIAGE, an evaluation framework in which a model receives a task pool and a token budget calibrated to its own baseline cost, and commits to a single ordered plan that jointly encodes selection, sequencing, and per-problem allocation. Plans are scored against an oracle with full knowledge of the model's solvability and cost on each problem, yielding a triage efficiency ratio on a common scale. We evaluate frontier and open-source models, with and without reasoning enabled, across competition mathematics, graduate-level science, code generation, and expert multidisciplinary knowledge, and find that current language models exhibit substantial gaps in prospective metacognitive control, revealing a previously unmeasured capability dimension with direct implications for resource-efficient agent deployment.

1. Introduction

Scaling test-time compute has become a major paradigm for improving LLM reasoning (Snell et al., 2025), but it has also made compute allocation a central unsolved problem: current reasoning models lack reliable *prospective control* over their own compute expenditure, and adaptive allocation per problem can yield substantial efficiency gains over uniform budgets (Snell et al., 2025; Alomrani et al., 2025). Reasoning models generate thousands of tokens even for trivial problems (Chen et al., 2024) and underthink hard ones, prematurely abandoning promising reasoning paths (Wang et al., 2025b). *Unconstrained agents*, similarly, can consume thousands of tokens on agentic tasks and still fail, exhibiting analysis paralysis and wasted compute that correlates with decreased performance (Cuadron et al., 2025).

This creates a concrete deployment problem: agents built on these models face real resource constraints - token budgets, tool-call limits, latency windows - and must decide which tasks to attempt before committing resources. A resource-rational agent should allocate finite compute to the tasks where its expected return is highest (Lieder & Griffiths, 2020). In practice, this means an LLM facing a queue of problems under a token budget should abstain from attempting ones it cannot solve, spend less on easy ones, and safely commit sufficient resources to each attempted problem so that under-allocation does not expend budget on partial or incomplete solutions that yield no return - the same strategy a student applies on a timed exam - the capability that Nelson & Narens (1990) formalized as *metacognitive control*: regulating effort allocation based on judgments of one's own knowledge state.

Recent work suggests LLMs have some of this capability in isolation: large language models can predict whether they will answer a question correctly (Kadavath et al., 2022), yet reasoning-trained models are worse - not better - at knowing when to abstain (Kirichenko et al., 2025). Barkan et al. (2025) found that frontier models are systematically overconfident about their own capabilities, and that this overconfidence does not diminish with scale or reasoning augmentation. The critical capability for an LLM planner is not just predicting token cost or knowing when to abstain from a single question, but managing a portfolio: given a set of tasks and a shared budget, selecting which to attempt, estimating cost, and ordering execution to maximize value. This is the problem that many real-world agentic deployments face - coding agents working through backlogs of issues, research agents decomposing queries into sub-questions of varying difficulty, and planning agents budgeting compute

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

across subtasks of a larger goal - all require cross-task prioritization under finite compute. Existing per-task budgeting methods estimate token cost for a single problem (Han et al., 2025a; Li et al., 2025), and external batch schedulers rank or route requests for throughput (Jin et al., 2023; Fu et al., 2024). However, whether LLMs can themselves perform the joint optimization - assessing their own feasibility, cost, and priority across a heterogeneous task set under a shared budget - remains untested, and answering it is a prerequisite for building agents that can plan and optimize their own resource use rather than relying on external orchestration.

We introduce TRIAGE, an evaluation framework for **prospective metacognitive control** under resource constraints. The construct has three components. **Metacognitive control** is the use of self-knowledge to regulate behavior, graded by the consequences of the model's plan rather than by verbalized confidence (Ackerman, 2026). **Prospective** means all decisions are committed before any task is attempted, distinguishing TRIAGE from retrospective measures of confidence or post-hoc calibration (Kadavath et al., 2022; Koriat, 1997). **Resource constraints** refers to a finite shared budget - output tokens by default - that the model's behavior consumes (Han et al., 2025a). In TRIAGE, a model receives a set of problems, point values per problem, and a token budget calibrated to its own baseline cost on the set. It returns a single ordered plan specifying which problems to attempt, how many tokens to allocate to each, and in what order. This design operationalizes three foundational control functions from Nelson & Narens (1990) in their prospective form: selection (which problems to attempt), allocation (how much compute to commit per problem), and termination (where to commit budget cutoffs ex ante). TRIAGE evaluates the control output directly rather than asking the model to verbalize its underlying judgments - appropriate for LLMs, where self-reports are an unreliable signal of internal state and metacognition is better measured behaviorally (Ackerman, 2026). Just as human learners plan effort allocation under time pressure (Son & Metcalfe, 2000), an autonomous agent must jointly optimize them to maximize expected utility under finite inference limits.

We evaluate 20 models - 4 standard LLMs and 16 reasoning-augmented LLMs, evaluated both with and without thinking enabled - across four domains: competition mathematics (AIME 2024–2025), graduate-level science (GPQA Diamond; Rein et al., 2023), code generation (LiveCodeBench; Jain et al., 2024), and multidisciplinary expert knowledge (Humanity's Last Exam; Phan et al., 2025b).

We find that current language models exhibit substantial and uneven gaps in prospective metacognitive control. Strong self-assessment in one domain rarely transfers to another, extended reasoning improves task-level accuracy without improving triage quality, and reasoning-trained models are less likely to recognize unsolvable items when they appear in a task pool. The gap between advisory and enforced regimes is wide across budget levels, indicating that models can sometimes choose what to attempt but rarely commit budgets that they themselves can honor. Together, these experiments characterize a previously unmeasured capability dimension with direct implications for resource-efficient agent deployment.

2. Related Work

2.1. Metacognition in Large Language Models

Metacognition, the capacity to monitor and regulate one's own cognitive processes (Nelson & Narens, 1990), has become a central concern in LLM evaluation. Research has progressed through three levels of investigation: metacognitive *knowledge* (can the model identify what a task requires?), metacognitive *monitoring* (can the model judge whether it will succeed?), and metacognitive *control* (can the model act on those judgments to regulate its behavior?). Existing work has made progress on the first two; the third remains untested because no current evaluation forces a model to commit, in advance, to a portfolio of tasks under a shared budget – a regime where control becomes operationally distinct from monitoring.

Metacognitive knowledge and monitoring. Kadavath et al. (2022) provided the foundational result on monitoring, showing that LLMs can predict which questions they will answer correctly, with calibration improving at scale. Didolkar et al. (2024) demonstrated that LLMs also possess metacognitive *knowledge*: given a math problem, frontier models can assign meaningful skill labels and use skill-matched exemplars to improve reasoning, establishing that models can assess task properties. Xiong et al. (2024) systematically benchmarked confidence elicitation methods and found that LLMs are consistently overconfident when verbalizing confidence, though both calibration and failure prediction improve with scale. Wang et al. (2025a) formalized the separation of metacognitive ability from raw cognitive ability via their DMC framework, finding that stronger LLMs exhibit stronger metacognition and that enhancing metacognition alleviates hallucination. At the mechanistic level, Ji-An et al. (2025) demonstrated that LLMs can monitor and control a low-dimensional subspace of their internal activations via a neurofeedback paradigm – direct evidence of metacognitive access to internal states, though limited to concurrent activations rather than prospective task-level judgments.

These results are counterbalanced by evidence of systematic failure. Kapoor et al. (2024) showed that verbalized confidence is poorly calibrated for open-ended generation and that a classifier trained on internal features substantially

outperforms prompting – implying that metacognitive monitoring is latent in representations but not reliably accessible through natural outputs. Griot et al. (2025) found that LLMs provide confident answers to medical questions even when correct options are entirely absent, revealing a disconnect between stated confidence and actual capability. Ackerman (2026) used behavioral paradigms where the model must decide before answering whether to attempt or defer (the Delegate Game) and found only rudimentary, context-dependent metacognition that depends on training regimen rather than scale. Steyvers & Peters (2025) organize these mixed findings by distinguishing metacognitive *sensitivity* (ranking own performance across items) from *calibration* (matching confidence to success rate), noting that the two can vary independently and that neither is reliably strong across models and domains.

From monitoring to control. The work above measures whether models can assess their own capabilities. Real deployment requires *acting* on those assessments to allocate resources. The closest prior work to ours is Barkan et al. (2025), who explicitly distinguish in-advance from after-the-fact confidence and test prospective self-assessment on coding and agentic tasks (SWE-Bench Verified). They find that frontier models are systematically overconfident, and that neither scale nor reasoning augmentation improves prediction accuracy. Their resource-acquisition experiment—accepting or declining sequential work contracts at varying cost—is the nearest existing analog to budget-constrained decision-making. But this evaluation design—sequential, independent accept/reject decisions—by construction cannot expose the combinatorial problem of jointly selecting, allocating, and ordering under a shared budget, where attempting one problem reduces what remains for all others. The design choice, not the model, bounds what the experiment can measure.

AbstentionBench (Kirichenko et al., 2025) scales the monitoring question to a benchmark setting, testing whether models abstain from unanswerable questions across 20 datasets and finding that reasoning fine-tuning degrades abstention by $\sim$24%. Their unanswerable variants of existing benchmarks inform our unsolvable-injection experiment. But abstention there is a property of the *question*—some are answerable, others are not. In TRIAGE, the relevant property is the *model–task–budget triple*: a problem may be answerable in principle but not by this model under this budget.

Across this literature, a consistent picture emerges: LLMs possess metacognitive knowledge and some monitoring ability, but both are contested in degree and fragile across domains. More critically, all existing evaluations operate one task at a time. The portfolio-level version of the problem is harder—accurate monitoring of any one task is necessary but not sufficient, because the model must also compare feasibility and cost across tasks while respecting a shared constraint. Whether current models can make this transition from single-task monitoring to portfolio-level resource allocation is what TRIAGE is designed to test.

2.2. Compute Allocation for LLM Reasoning

The test-time compute scaling paradigm (Snell et al., 2025) has made compute allocation a practical problem. Reasoning models overthink trivial problems (Chen et al., 2024), underthink hard ones (Wang et al., 2025b), and in agentic settings exhibit analysis paralysis that correlates with decreased performance (Cuadron et al., 2025). Existing methods address this waste, but split on who allocates and at what granularity, and neither side covers the full problem.

On the self-assessed side, TALE (Han et al., 2025a) has the LLM estimate its own token cost before solving a problem, and SelfBudgeter (Li et al., 2025) trains the model to emit a budget before its chain of thought. Both are prospective and self-assessed, but per-task: the model decides how long to spend, never whether to attempt at all, and there is no shared budget linking decisions across tasks. On the systems side, external predictors schedule LLM inference for throughput—via length-bucket classification (Jin et al., 2023), listwise ranking (Fu et al., 2024), or interval prediction under uncertainty (Chen et al., 2025). These operate at the portfolio level, but the predictor is external to the LLM, the objective is latency, and the model's own judgment of what it can solve plays no role.

Existing benchmarks test related but distinct capabilities. PlanBench (Valmeekam et al., 2023) evaluates whether LLMs can produce valid plans in classical planning domains, with automated ground-truth validation via external solvers—but tests plan *execution*, not prospective self-assessment of which plans will succeed. τ-bench (Yao et al., 2024) introduces tool-call interactions as a measurable resource axis for agent evaluation, and GAIA (Mialon et al., 2024) provides a heterogeneous, difficulty-leveled agent benchmark with validated ground truth. Both motivate the agentic extension of TRIAGE but neither tests whether the agent can triage its own workload under finite resources.

These two literature meet at a measurement gap. Single-task confidence benchmarks isolate monitoring; external schedulers isolate allocation; neither evaluates the joint capability that deployment actually requires. TRIAGE is designed to close this gap as an evaluation. The model produces a plan—selecting tasks, estimating costs, and ordering execution—which we score by simulating its consequences against pre-computed ground truth, yielding a single, falsifiable measure of the joint product of metacognitive monitoring accuracy and planning quality under a binding budget. The construct is what is new: a portfolio-level test in which accurate per-

task self-assessment is necessary but not sufficient, because the model must also compare feasibility and cost across tasks while respecting a shared constraint.

3. TRIAGE Framework

3.1. Prospective Metacognitive Control

Effective problem-solving under constraint requires more than the capacity to execute individual tasks; it requires metacognitive control over how that capacity is deployed, and for an agent operating under a finite execution budget, that control is exercised as prospective planning: a commitment, made before any problem is attempted, to which problems will be pursued, in what order, and how much of the budget each will receive. Cognitive psychology distinguishes an *object level*, where a task is executed, from a *meta level*, which monitors the object level and exerts control over it (Nelson & Narens, 1990; Koriat & Goldsmith, 1996). Two control functions of the meta level, what to attempt and how long to persist on what is attempted, are the levers behind human study-time allocation (Son & Metcalfe, 2000; Metcalfe & Kornell, 2005) and test-taking strategy (Ackerman & Goldsmith, 2011). We benchmark these levers in language models.

Metacognitive judgments partition by their position in the execution timeline (Dunlosky & Metcalfe, 2009): prospective, formed before execution; concurrent, formed during; and retrospective, formed after. We focus on the prospective class, Ease-of-Learning judgments in the laboratory (Underwood, 1966; Leonesio & Nelson, 1990), initial test plans in applied settings, for three independent reasons.

Methodological isolation. Once feedback is allowed during execution, observed performance reflects an unidentifiable mixture of prospective judgment quality and adaptive replanning; the two cannot be separated post-hoc. Studying prospective judgments without intervening feedback is the established technique for isolating the construct in human work (Mazzoni & Cornoldi, 1993; Son, 2004).

Deployment realism. A growing class of LLM-driven systems commits to a task-level plan over a heterogeneous set of subtasks before any one is attempted. Deep research agents decompose a query into sub-questions and dispatch them across parallel sub-agents, each running with its own context window and token budget that the orchestrator cannot revise once dispatched (Lin et al., 2026). Coding agents process queues of bug reports or pull-request issues under per-issue token budgets, where wasted attempts on infeasible items measurably degrade overall throughput (Jimenez et al., 2024). In both cases the quality of the upfront plan, what to attempt, in what order, with what allocation, determines whether the run completes usefully or exhausts its budget on the wrong items.

Construct priority. Concurrent strategies are downstream of prospective competence (Thiede & Dunlosky, 1999): quantifying what online feedback adds presupposes a measurement of the floor it acts on. The blind regime supplies that floor.

Tokens as the control variable. Classical study-allocation work uses time as the meta-level control variable. In autoregressive language models, output tokens are the natural unit of expended effort: generation time, compute, and monetary cost are all approximately proportional to token count, and the central paradigm of test-time compute scaling treats per-problem token allocation as the system's primary lever for adapting to problem difficulty (Snell et al., 2025; Han et al., 2025b). Tokens are therefore the operational analog of time in study-allocation theory, with the additional property of being exactly measurable.

Distributional self-knowledge and binding plans. Common agent pipelines generally decouple planning from execution into separate inference calls over identical weights (Xu et al., 2023). The planner does not introspect on an ongoing computation; it predicts the behavior of a same-weights solver under standard execution conditions. We treat this as the deployment-relevant form of self-modeling. The plan is moreover *operationally binding*: at execution time, per-problem token allocations from the plan can be enforced as binding constraints on the solver, converting the plan from a forecast into a self-commitment and testing whether the model can satisfy a budget it set for itself; we study this enforced setting alongside the unconstrained one as a secondary regime (§3.2).

Four primitives. Prospective metacognitive control under a finite execution budget reduces to four jointly exercised judgments: (i) *feasibility*, whether the model can solve the problem? (ii) *cost*, how many tokens will it take? (iii) *selection*, which subset to attempt? and (iv) *sequencing*, in what order, given that budget exhaustion truncates the tail? Existing LLM benchmarks isolate slices of this set: per-item confidence calibration covers (i) (Barkan et al., 2025); abstention covers a hard form of (i) without a budget (Kirichenko et al., 2025); per-task budget control covers (ii) for a single item (Li et al., 2025). TRIAGE measures the joint exercise of all four primitives under a finite budget.

3.2. Problem Formulation

We formalize TRIAGE for a single *task pool*, a finite set of problems sharing a domain and a budget, the natural unit faced by an agent when it begins a job.

Setup. A task pool $P = \{p_1, \ldots, p_n\}$ contains finitely many problems. For each problem p_i and a fixed solver model M: $v_i \in \mathbb{R}_{>0}$ is the point value, observable to M; $y_i \in \{0, 1\}$ indicates whether M solves p_i under standard execution, unobservable to M at planning time; and $c_i \in \mathbb{R}_{>0}$ is the output-token cost M incurs when attempting p_i, also unobservable at planning time. Both y_i and c_i are properties of the (problem, solver) pair, determined empirically by running M on p_i once at temperature 0.

Budget. The budget is $B(M, \alpha) = \alpha \cdot \sum_{i=1}^{n} c_i$ with $\alpha \in (0, 1]$. The factor $\alpha < 1$ forces selection by setting the budget below the solver's full baseline cost; $\alpha = 1$ supplies the full baseline budget, in which selection pressure is reduced but the model must still identify which items it can solve, since omissions still forfeit value.

Action. The model produces an ordered plan $\pi = (i_1, \ldots, i_k)$ with $0 \leq k \leq n$, indexing a selected subset $S = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, together with per-problem allocations $a_{i_j} \in \mathbb{R}_{\geq 0}$ satisfying $\sum_j a_{i_j} \leq B$. The plan thus jointly encodes selection, sequencing, and forecast.

Execution. Plans are executed in plan order under one of two regimes, which differ in whether the per-problem allocations a_i are binding.

Unconstrained regime (U). The solver runs each problem to its natural completion at the cost c_i; only the global budget B binds. Execution proceeds through π until reaching an item whose true cost exceeds the remaining budget, at which point execution stops.

Constrained regime (E). Each problem i_j is executed with a hard token limit a_{i_j} at the solver: if the solver completes within a_{i_j}, the outcome is the result under that limit; otherwise the attempt scores zero. The full allocation a_{i_j} is deducted from the remaining budget regardless of outcome, and execution proceeds while the remaining budget covers the next item's allocation.

The two regimes isolate distinct components of the meta-level control problem (Nelson & Narens, 1990). Regime U probes *prospective monitoring*: the planner's predictions of (y_i, c_i) surface only through which items it selects and in what order, since allocations are advisory; truncation discipline comes entirely from ordering quality. Regime E probes *prospective control* in the strict sense: per-problem allocations are binding, so the model is held to commitments it makes to itself. We write $V_M(\pi)$ for the achieved value at termination under whichever regime is in force, and report results in both.

Triage efficiency ratio. We benchmark the model's achieved value against an oracle that has the information the model lacks. The *oracle* is a planner with full knowledge of (y_i, c_i) for every i; its achievable value is $V_{\text{oracle}} = \max_{x \in \{0,1\}^n} \sum_{i=1}^{n} v_i \, y_i \, x_i$ subject to $\sum_{i=1}^{n} c_i \, x_i \leq B$, a 0–1 knapsack: each problem is either included in the plan ($x_i = 1$) or omitted ($x_i = 0$), with no fractional attempts permitted (Kellerer et al., 2004). The *random* reference V_{random} is the expected value of a feasible plan obtained without any self-knowledge, by drawing problems uniformly without replacement subject to remaining-budget feasibility; we report the mean over 10^3 such samples. The triage efficiency ratio normalizes the model's achieved value between these two reference points: $\eta_M = \big(V_M(\pi) - V_{\text{random}}\big) / \big(V_{\text{oracle}} - V_{\text{random}}\big)$, placing models on a 0–1 scale, where $\eta = 0$ corresponds to no use of self-knowledge and $\eta = 1$ to the value attainable with perfect self-knowledge. This is the standard skill-score normalization used in forecasting (Murphy, 1973) and in selective-prediction evaluation, where system performance is normalized against an oracle that observes the labels the system must predict (Geifman & El-Yaniv, 2017).

The main protocol fixes $v_i = 1$ due to leakage: any v_i correlated with empirical difficulty would let the model read difficulty off the prompt rather than estimate it from problem content, and language models are known to exploit such surface cues when present (Koriat, 1997; Tang et al., 2023).

Under uniform v the oracle simplifies further: V_{oracle} equals the maximum number of solvable items ($y_i = 1$) whose costs fit within B. Let $S^\star = \{i : y_i = 1\}$ and let $i_{(1)}, i_{(2)}, \ldots$ index $S^\star$ in ascending order of c_i; then $V_{\text{oracle}} = \max\{m : \sum_{j=1}^{m} c_{i_{(j)}} \leq B\}$.

The model receives only problem text and must form its plan without observing (y_i, c_i). Differences in η between models therefore reflect differences in their implicit predictions of (y_i, c_i), not differences in optimization: the oracle, the random reference, and the execution rule are identical across models. This is the sense in which TRIAGE measures metacognition, the model's prediction of its own future behavior, with the oracle playing the role of an ideal forecaster, as in selective-prediction evaluation more broadly (Geifman & El-Yaniv, 2017).

4. Experimental Setup

Models. We evaluate 20 models spanning frontier and open-source families (Qwen 2.5/3/3.5, GPT-OSS, Claude, DeepSeek, Gemini, GLM, GPT-5, Kimi), with reasoning enabled and disabled where supported. Full model identifiers, providers, parameter counts, and reasoning configurations are listed in Table 4 (Appendix E).

Datasets and task pools. We construct task pools from four domains: competition mathematics (AIME 2024–

2025), graduate-level science (GPQA Diamond (Rein et al., 2024)), code generation (LiveCodeBench (Jain et al., 2024), filtered to contests from 2025 onward), and multidisciplinary expert knowledge (Humanity's Last Exam (Phan et al., 2025a), excluding image-bearing items and topic categories overlapping with the other three datasets). Each dataset is partitioned into independent task pools of $n_{\text{chunk}} = 30$ problems to fit within planner context limits and avoid the long-context degradation reported in prior work (Liu et al., 2024). Each pool has its own budget, oracle, and random reference, and η is averaged across pools within each $(\text{dataset}, \alpha)$ cell.

To probe sensitivity to infeasible items, we additionally inject unsolvable problems from Abstention-Bench (Kirichenko et al., 2025) (GPQA-Abstain and MMLU-Math-Abstain) into pools at varying ratios.

Phase 0: baseline measurement. We first run each model on every problem once, in an isolated session at temperature 0, to record its baseline output-token cost c_i and correctness y_i. These two quantities serve two purposes: c_i calibrates the per-pool budget $B_{\text{local}} = \lfloor \alpha \sum_i c_i \rfloor$ given to the planner in Phase 1, and (c_i, y_i) together provide the ground truth against which the oracle and η are computed.

Phase 1: planning. The planner receives problem texts, point values ($v_i = 1$), the per-pool budget $B_{\text{local}} = \lfloor \alpha \sum_{i \in \text{pool}} c_i \rfloor$, and instructions to return an ordered plan with per-problem token allocations summing to at most B_{local}. The full prompt template is given in Appendix D. A prompt sensitivity analysis is available in D.2 and shows robustness under prompt variations.

Phase 2: execution. Plans are executed under regime U as the main condition and regime E as a secondary condition (§3.2), reusing Phase 0 outcomes when allocations are non-binding so that observed differences in η reflect planning quality rather than solver stochasticity.

Budget conditions. We sweep $\alpha \in \{0.25, 0.5, 0.75, 1.0\}$ to vary selection pressure from severe to slack. Main results report $\alpha = 1.0$.

Metric. We report triage efficiency η (§3.2). The random reference V_{random} is estimated by drawing 10^3 uniform shuffles of the gradeable items per pool and executing each under regime-U rules - the same execution rule used to score model plans, ensuring η measures the value added by self-knowledge over an unannotated random plan.

For the unsolvable-injection probe, we additionally report *waste rate* $W = \sum_{i \in \mathcal{S} \cap \mathcal{U}} a_i \,/\, \sum_{i \in \mathcal{S}} a_i$, the fraction of allocated tokens spent on unsolvable items, and *detection rate* $D = |\mathcal{U} \setminus \mathcal{S}| \,/\, |\mathcal{U}|$, the fraction of unsolvable items correctly excluded from the plan, where $\mathcal{S} = \{i : a_i > 0\}$ is the planned set and $\mathcal{U}$ the injected unsolvable items in the pool.

5. Results

5.1. Triage Efficiency at Full Budget

Figure 1 reports triage efficiency at full budget ($\alpha = 1.0$). At $\alpha = 1.0$ the budget covers every item's baseline cost, so selection pressure is at its weakest; any failure of triage observed here is a lower bound on what tighter budgets will reveal. The main findings are described below.

Monitoring is uneven and does not transfer. η_U scatters widely between 0 and 1, with no model reaching oracle-level on more than two panels. GPT-5 Mini and DeepSeek V3 both reach $\eta_U \approx 0.86$ on GPQA in extended mode but fall to $\eta_U \leq 0.5$ on LCB. Strong self-assessment in one domain does not carry to another. The exception is GLM 5.1 in standard inference, which attains $\eta_U = 1.0$ on all four benchmarks despite low base accuracy (10–31% on AIME, GPQA, and LCB), indicating very accurate self-assessment over a small set of solvable items.

Enforcement collapses. The gap between η_U (solid) and η_E (hatched) is the most consistent pattern. On AIME and GPQA, η_E falls to zero for nearly every model: planned token allocations are too tight, attempts run out before answering, and binding the budget yields no value. HLE is the exception, where solutions are short enough that mis-sized allocations still suffice. Models can pick what to attempt; they cannot budget how long each will take. Kimi K2.5 on AIME extended exemplifies the worst case. A controlled re-solve experiment confirms this gap behaviorally: models rarely honor the budgets they themselves declare even if instructed (Appendix D.1).

Extended reasoning lifts accuracy, not triage. Within paired models, base accuracy (red dashed line) generally rises with extended reasoning (e.g. Gemini 2.5 Flash on AIME: $0.28 \rightarrow 0.62$). The corresponding η_U bars do not move consistently, and several drop (Qwen 3.5 27B and GLM 5.1 on AIME). Object-level capacity improves; meta-level self-knowledge does not.

Behavior across budget pressure. Full per-(model, α, dataset) breakdowns of the three core metrics appear in the appendix: accuracy in Figure 3, advisory-regime triage skill η_U in Figure 4, and enforced-regime triage skill η_E in Figure 5. As budgets tighten, η_U tends to degrade for most models, with several smaller models dropping substantially below the random baseline on AIME at intermediate α, suggesting that selection pressure exposes triage failures that slack budgets can absorb. GLM 5.1 shows the most consis-

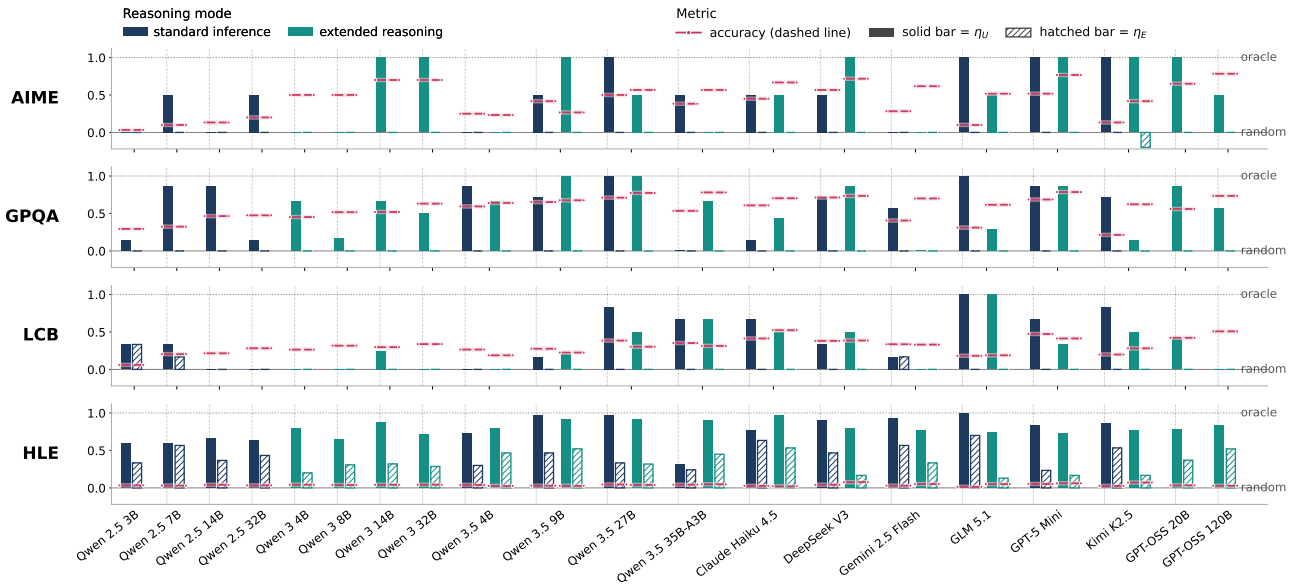

Figure 1. **Triage efficiency across models and benchmarks at full budget ($\alpha = 1.0$).** Solid bars show η_U (unconstrained regime, advisory allocations), hatched bars show η_E (constrained regime, binding allocations), and red dashed lines mark base accuracy. $\eta = 1$ is oracle triage; $\eta = 0$ is random. Bar color distinguishes standard inference from extended reasoning.

tent performance in the advisory regime, sustaining high η_U across budget levels and benchmarks. The enforcement gap likewise persists beyond $\alpha = 1.0$: at tighter budgets, η_E on the reasoning-heavy benchmarks (AIME, GPQA, LCB) is typically negative for most models, consistent with allocations that bind before the solver can finish. The mechanism shifts across α, but the broader pattern - models struggling to honor self-imposed token commitments - holds throughout.

Across 20 models and four benchmarks, no model is simultaneously a strong monitor and a strong controller on more than one domain. Triage quality does not scale monotonically with *parameter count* in any family. Within Qwen 2.5, the largest 32B model is the worst on GPQA ($\eta_U = 0.14$ vs. 0.86 for 7B/14B) and η_U on LCB decreases monotonically with size from 0.33 to 0; within Qwen 3.5, the 35B-A3B falls below the dense 27B on every benchmark; within GPT-OSS, the 120B underperforms the 20B on three of four benchmarks. The only clean upward trend is Qwen 3 on AIME (4B/8B at $\eta_U = 0$, 14B/32B at $\eta_U = 1.0$), and even that family is non-monotonic on the remaining three benchmarks. Prospective metacognitive control, in the joint sense of §3.1, remains an open capability gap.

5.2. Recognizing Unsolvable Problems

We probe sensitivity to infeasible items of selected models by injecting unsolvable problems from Abstention-Bench (Kirichenko et al., 2025) (GPQA-Abstain, MMLU-Math-Abstain) into the GPQA and MMLU-Math pools at substitution ratios $r \in \{0, 0.25, 0.50, 0.75, 1.00\}$, holding pool size constant. We report detection rate D (fraction of unsolvables correctly left out of the plan) and waste rate W (fraction of the plan's tokens spent on unsolvables); see §4. Figure 2 traces each ablation model through (D, W) space; the main findings are described below.

Recognition of unsolvables varies across models and reasoning modes. Two models keep recognizing unsolvables as injection rises: Gemini 2.5 Flash *(non-thinking)* and Qwen 2.5 32B both stay on the right side of every panel. The two reasoning-trained models - Qwen 3 32B and GPT-OSS 20B - remain pinned to the left edge across both datasets, with detection close to zero. The Qwen pair gives the cleanest within-family read: under standard inference, the model recognizes most unsolvables; under extended reasoning, it recognizes almost none of them, and this holds on both datasets. The pattern reproduces under portfolio planning what Kirichenko et al. (2025) report for isolated abstention - extended reasoning training reduces sensitivity to infeasibility.

The dependence of detection on injection ratio. We characterize each model by how its detection rate varies with the injection ratio. Gemini's detection rate is approximately invariant to injection on both datasets. Qwen 2.5 32B's detection rate generally declines monotonically with injection. The two reasoning-trained models hold detection rate below 50% across most injection ratios, indicating allocations that do not distinguish unsolvable items from solvable ones.

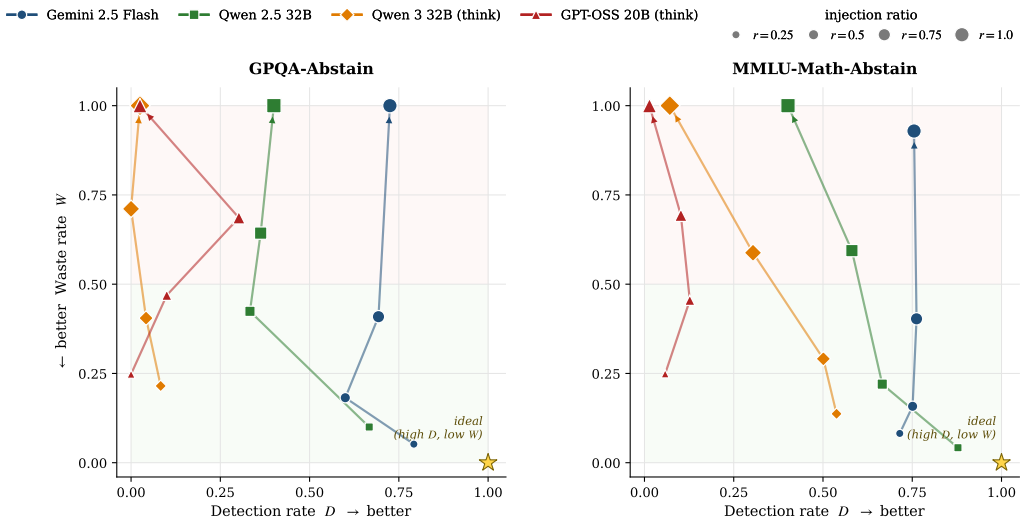

Figure 2. **Trajectories through (D, W) space as the unsolvable-injection ratio r increases from 0.25 to 1.00 (marker size).** The star marks the ideal corner: high detection rate, low waste rate.

Cross-domain consistency of detection. Three of four models detect unsolvables more reliably on the math-derived dataset than on the science-derived one. Gemini does not exhibit this gap: its detection is mostly comparable on both datasets.

Full injection isolates detection from solving ability. At full injection every item is infeasible and the optimal plan is empty, so solving ability cannot contribute to detection. The cohort split persists. Gemini and Qwen 2.5 32B continue to exclude a substantial fraction of items from the plan; the two reasoning-trained models continue to allocate budget under conditions where any allocation is wasted by construction. Full injection also partially removes between-domain difficulty as a source of variance, leaving willingness to refuse as the main residual factor distinguishing the cohort.

6. Conclusion

Deploying language models as autonomous agents requires a capability that single-task evaluation does not test: prospective metacognitive control over which problems to attempt, in what order, and with what allocation, committed before any execution feedback. TRIAGE evaluates this capability by scoring committed plans against an oracle with full knowledge of model-specific solvability and cost.

Across 20 models and four domains, object-level capability and metacognitive control dissociate. Extended reasoning improves accuracy without improving triage efficiency. Self-assessment does not transfer reliably across domains. Models select which problems to attempt more reliably than they commit token budgets they can honor under enforcement. Reasoning training reduces, rather than improves, sensitivity to infeasible items at the portfolio level. Because the failure is prospective, execution-time monitoring cannot repair it. Reliable upfront planning is therefore a prerequisite, not a refinement, for agents that allocate their own compute under finite resources.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ackerman, C. Evidence for limited metacognition in LLMs. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.21545.

Ackerman, R. and Goldsmith, M. Metacognitive regulation of text learning: On screen versus on paper. *Journal of Experimental Psychology: Applied*, 17(1):18–32, 2011. doi: 10.1037/a0022086.

Alomrani, M. A., Zhang, Y., Li, D., Sun, Q., Pal, S., Zhang, Z., Hu, Y., Ajwani, R. D., Valkanas, A., Karimi, R., Cheng, P., Wang, Y., Liao, P., Huang, H., Wang, B., Hao, J., and Coates, M. Reasoning on a budget: A survey of adaptive and controllable test-time compute in LLMs. *arXiv preprint arXiv:2507.02076*, 2025.

Barkan, C. O., Black, S., and Sourbut, O. Do large language models know what they are capable of? *arXiv preprint arXiv:2512.24661*, 2025. NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle.

Chen, X., Xu, J., Liang, T., He, Z., Pang, J., Yu, D., Song, L., Liu, Q., Zhou, M., Zhang, Z., Wang, R., Tu, Z., Mi, H., and Yu, D.

Do NOT think that much for 2+3=? On the overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*, 2024.

Chen, Z., Ye, Y., and Zhou, Z. Adaptively robust LLM inference optimization under prediction uncertainty. *arXiv preprint arXiv:2508.14544*, 2025.

Cuadron, A., Li, D., Ma, W., Wang, X., Wang, Y., Zhuang, S., Liu, S., Schroeder, L. G., Xia, T., Mao, H., Thumiger, N., Desai, A., Stoica, I., Klimovic, A., Neubig, G., and Gonzalez, J. E. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025.

Didolkar, A., Goyal, A., Ke, N. R., Guo, S., Valko, M., Lillicrap, T., Rezende, D., Bengio, Y., Mozer, M., and Arora, S. Metacognitive capabilities of llms: An exploration in mathematical problem solving. *Advances in Neural Information Processing Systems*, 37:19783–19812, 2024.

Dunlosky, J. and Metcalfe, J. *Metacognition*. SAGE Publications, Thousand Oaks, CA, 2009. ISBN 9781412939720.

Fu, Y., Bailis, P., Stoica, I., and Zhang, H. Efficient LLM scheduling by learning to rank. In *Advances in Neural Information Processing Systems*, volume 37, 2024. arXiv:2408.15792.

Geifman, Y. and El-Yaniv, R. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Griot et al. Large language models lack essential metacognition for reliable medical reasoning. *Nature Communications*, 16: 642, 2025. doi: 10.1038/s41467-024-55628-6.

Han, T., Wang, Z., Fang, C., Zhao, S., Ma, S., and Chen, Z. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025a. arXiv:2412.18547.

Han, T., Wang, Z., Fang, C., Zhao, S., Ma, S., and Chen, Z. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025b. doi: 10.18653/v1/2025.findings-acl.1274.

Jain, N., Han, K., Gu, A., Li, W.-D., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Ji-An, L., Xiong, H.-D., Wilson, R. C., Mattar, M. G., and Benna, M. K. Language models are capable of metacognitive monitoring and control of their internal activations. *arXiv preprint arXiv:2505.13763*, 2025.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. R. SWE-bench: Can language models resolve real-world GitHub issues? In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=VTF8yNQM66`.

Jin, Y., Wu, C.-F., Brooks, D., and Wei, G.-Y. S^3: Increasing GPU utilization during generative inference for higher throughput. In *Advances in Neural Information Processing Systems*, volume 36, 2023. arXiv:2306.06000.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., Johnston, S., El-Showk, S., Jones, A., Elhage, N., Hume, T., Chen, A., Bai, Y., Bowman, S., Fort, S., Ganguli, D., Hernandez, D., Jacobson, J., Kernion, J., Kravec, S., Lovitt, L., Ndousse, K., Olsson, C., Ringer, S., Amodei, D., Brown, T., Clark, J., Joseph, N., Mann, B., McCandlish, S., Olah, C., and Kaplan, J. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Kapoor, S., Gruver, N., Roberts, M., Collins, K., Pal, A., Bhatt, U., Weller, A., Dooley, S., Goldblum, M., and Wilson, A. G. Large language models must be taught to know what they don't know. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Kellerer, H., Pferschy, U., and Pisinger, D. *Knapsack Problems*. Springer, Berlin, Heidelberg, 2004. ISBN 978-3-540-40286-2. doi: 10.1007/978-3-540-24777-7.

Kirichenko, P., Ibrahim, M., Chaudhuri, K., and Bell, S. J. AbstentionBench: Reasoning LLMs fail on unanswerable questions. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, volume 38, 2025. arXiv:2506.09038.

Koriat, A. Monitoring one's own knowledge during study: A cue-utilization approach to judgments of learning. *Journal of experimental psychology: General*, 126(4):349, 1997.

Koriat, A. and Goldsmith, M. Monitoring and control processes in the strategic regulation of memory accuracy. *Psychological Review*, 103(3):490–517, 1996. doi: 10.1037/0033-295X.103.3.490.

Leonesio, R. J. and Nelson, T. O. Do different metamemory judgments tap the same underlying aspects of memory? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 16(3):464–470, 1990. doi: 10.1037/0278-7393.16.3.464.

Li, Z., Dong, Q., Ma, J., Zhang, D., Jia, K., and Sui, Z. SelfBudgeter: Adaptive token allocation for efficient LLM reasoning. *arXiv preprint arXiv:2505.11274*, 2025.

Lieder, F. and Griffiths, T. L. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1, 2020. doi: 10.1017/S0140525X1900061X.

Lin, X., Liew, J. H., Savarese, S., and Li, J. W&D: Scaling parallel tool calling for efficient deep research agents. *arXiv preprint*, 2026.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Mazzoni, G. and Cornoldi, C. Strategies in study time allocation: Why is study time sometimes not effective? *Journal of Experimental Psychology: General*, 122(1):47–60, 1993. doi: 10.1037/0096-3445.122.1.47.

Metcalfe, J. and Kornell, N. A region of proximal learning model of study time allocation. *Journal of Memory and Language*, 52 (4):463–477, 2005. doi: 10.1016/j.jml.2004.12.001.

Mialon, G., Fourrier, C., Swift, C., Wolf, T., LeCun, Y., and Scialom, T. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations*, 2024.

Murphy, A. H. A new vector partition of the probability score. *Journal of Applied Meteorology*, 12(4):595–600, 1973. doi: 10.1175/1520-0450(1973)012<0595:ANVPOT>2.0.CO;2.

Nelson, T. O. and Narens, L. Metamemory: A theoretical framework and new findings. In Bower, G. H. (ed.), *The Psychology of Learning and Motivation*, volume 26, pp. 125–173. Academic Press, 1990.

Phan, L., Gatti, A., Han, Z., Li, N., et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025a.

Phan, L. et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025b.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling (COLM)*, 2024.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. Oral. arXiv:2408.03314.

Son, L. K. Spacing one's study: Evidence for a metacognitive control strategy. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30(3):601–604, 2004. doi: 10.1037/0278-7393.30.3.601.

Son, L. K. and Metcalfe, J. Metacognitive and control strategies in study-time allocation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26(1):204–221, 2000. doi: 10.1037/0278-7393.26.1.204.

Steyvers, M. and Peters, M. A. K. Metacognition and uncertainty communication in humans and large language models. *Current Directions in Psychological Science*, 2025. doi: 10.1177/09637214251391158.

Tang, R., Kong, D., Huang, L., and Xue, H. Large language models can be lazy learners: Analyze shortcuts in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4645–4657, 2023. doi: 10.18653/v1/2023.findings-acl.284.

Thiede, K. W. and Dunlosky, J. Toward a general model of self-regulated study: An analysis of selection of items for study and self-paced study time. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 25(4):1024–1037, 1999. doi: 10.1037/0278-7393.25.4.1024.

Underwood, B. J. *Experimental Psychology*. Appleton-Century-Crofts, New York, 2nd edition, 1966.

Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., and Kambhampati, S. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems*, volume 36, 2023. Datasets and Benchmarks Track.

Wang et al. Decoupling metacognition from cognition: A framework for quantifying metacognitive ability in LLMs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 2025a. doi: 10.1609/aaai.v39i24.34723.

Wang, Y., Liu, Q., Xu, J., Liang, T., Chen, X., He, Z., Song, L., Yu, D., Li, J., Zhang, Z., Wang, R., Tu, Z., Mi, H., and Yu, D. Thoughts are all over the place: On the underthinking of o1-like LLMs. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. Spotlight. arXiv:2501.18585.

Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., and Hooi, B. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://arxiv.org/abs/2306.13063`.

Xu, B., Peng, Z., Lei, B., Mukherjee, S., Liu, Y., and Xu, D. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint*, 2023.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

A. Limitations and Broader Impact

TRIAGE measures one specific competence—prospective, portfolio-level resource allocation under a hard shared budget—and the design choices required to make this competence measurable bound what can be concluded from it.

Construct and design. TRIAGE operationalizes metacognitive control as a single up-front plan: the model commits to selections, cost estimates, and an execution order before any task is attempted. This isolates prospective self-assessment from in-the-loop revision, at the price of excluding settings where allocation is naturally interactive or streaming. We treat this trade as deliberate—the construct is most informative when revision is unavailable—but TRIAGE scores should not be read as a global verdict on a model's resource-allocation ability across deployment regimes. The budget is also denominated in a single resource per track (token cost in the reasoning track, tool calls in the agentic track); evaluations conditioned on wall-clock latency, monetary cost, or compound resources may rank models differently.

Ground truth and measurement noise. Per-task feasibility and cost are pre-computed by running each model on each item once, at temperature 0, prior to evaluation. This makes per-cell outcomes approximately deterministic at the provider/runtime level, so we do not report bootstrap error bars on per-cell η. Residual noise persists from provider-side non-determinism and from items near a model's capability frontier, where small perturbations can flip y_i. The unsolvable-injection slice further depends on injected items being genuinely unsolvable for the evaluated models; we audit this qualitatively but cannot exclude occasional successful completions on a small fraction.

External validity. Our model roster is a snapshot of frontier systems at submission time and will be partially superseded within months. We have no access to closed-model training data and cannot certify that TRIAGE items were held out of any model's pre- or post-training. Tasks are sourced from public reasoning and agentic benchmarks, which biases the picture toward English-language, formally-stated problems; conclusions about portfolio-level metacognition do not automatically transfer to multilingual, multi-modal, or open-ended generation regimes. Models also lack episodic memory of their own past performance on TRIAGE-like tasks; their "self"-assessment is therefore closer to assessment of the model class than to grounded recollection.

Broader impacts. The intended positive impact of TRIAGE is diagnostic: surfacing systematic overconfidence and miscalibration in resource decisions before LLMs are placed in deployments where compute, money, or human attention are at stake. The principal risk is dual-use of the artifact itself. Used unmodified as a training signal, TRIAGE could be optimized against, yielding models that score well without genuinely improved monitoring—a known failure mode of any sufficiently legible benchmark. We discourage fine-tuning on the public split and release a held-out evaluation split alongside it. We also caution that strong TRIAGE scores are not a license to delegate consequential decisions to LLMs: passing the benchmark is necessary, not sufficient, for such delegation, and we state this explicitly in the public documentation.

Responsible release of data and code. TRIAGE releases an evaluation harness, task pools drawn from existing public benchmarks, and pre-computed feasibility and cost tables for each evaluated model. We do not release a new pre-trained model, generator, or scraped corpus, so the misuse surface is correspondingly narrow. The repository ships with a datasheet, Croissant metadata including the Responsible AI extension, an explicit statement of permitted and discouraged uses, the held-out-split policy above, and licenses inherited from each upstream task source preserved verbatim. No personally identifying information is collected at any stage of evaluation.

Ethics. The work conforms with the NeurIPS Code of Ethics. No new human-subjects data was collected; all tasks derive from previously released datasets used under their original licenses. Aggregate compute consumed by feasibility profiling and evaluation runs is reported in the appendix. We declare no conflicts of interest beyond those listed in the author statement.

B. Dataset Documentation

C. Additional Experiments

D. Planner Prompt

The planner receives the prompt template shown in Listing 1. The prompt is held fixed across both regimes (U and E); regime U treats the per-problem allocations as advisory, while regime E enforces them as hard constraint (§3.2).

Listing 1. Planner prompt template used in Phase 1. Bracketed fields are filled in per task pool.

```
You are given a set of {n} problems and a total budget of {B_local} output tokens. Your
    task is to decide which problems to attempt, in what order, and how many tokens to
    allocate to each.

Constraints:
  - Each problem is worth 1 point. Your goal is to maximize the total points earned within
     the budget.
  - Each problem is attempted in an independent model session. The tokens you allocate to
    a problem are spent entirely within that single session, with no context, reasoning,
    or scratchpad carried over between problems.
  - The sum of your per-problem token allocations must not exceed {B_local}.
  - You may choose to attempt a strict subset of the problems. Problems you do not select
    receive 0 tokens and 0 points.

Problems:
{problems}

Return a single JSON object with the following schema:
{
  "plan": [
    {"id": <problem_id>, "tokens": <int>},
    ...
  ]
}

The order of items in "plan" is the order in which the problems will be attempted. Items
    omitted from "plan" are not attempted. Output only the JSON object, with no additional
     commentary.
```

Problem block format. Each problem in the `{problems}` field is rendered as:

```
[id: <problem_id>] (points: 1)
<problem_text>
```

Plan parsing and repair. Plan outputs are parsed as JSON. A small fraction of outputs contain recoverable formatting issues (trailing commentary outside the JSON object, extra fields, malformed numerics); these pass through a deterministic repair pass that strips non-JSON content, coerces token counts to non-negative integers, and discards plan items whose `id` does not appear in the pool. Plans that remain unparseable after repair are excluded; pools with no parseable plan for a model are excluded from that model's η average for the affected (dataset, α) cell.

D.1. Budget-aware re-solving: do models honor their own allocations?

The triage protocol measures *whether* a model can identify worthwhile problems given a fixed token budget; it does not test whether the model can actually execute under its own self-declared per-problem allocation a_i. We probe this directly with a **budget-aware re-solve** experiment on three models - Gemini 2.5 Flash, Kimi K2.5, and GLM 5.1, all in standard-inference mode. For every (model, problem) where the $\alpha = 1.0$ plan selected the problem with $a_i > 0$, we re-issue the standard solver prompt with an instruction stating the model's own a_i:

Budget-aware banner (prepended to the standard solver prompt)

```
You have a strict output-token budget of {a_i} tokens for solving this problem,
including any reasoning, drafts, or scratch work you produce.  Plan your solution
to fit within this limit.  Be concise, skip unnecessary explanation, and commit to
your final answer before exhausting the budget.
```

The API-level `max_tokens` is unchanged from the baseline run, so this imposes a *self-imposed* cap rather than a hard truncation. We record two outcomes per problem: $\mathbf{Acc}_{\text{baseline}}$ (the original baseline solve, no banner, full `max_tokens`) and $\mathbf{Acc}_{\text{budget-aware}}$ (with the banner). We also report **compliance**, defined as the fraction of problems for which the model's actual output length stays within its self-declared budget, i.e. compliance $= \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[\,\text{output_tokens}_i \leq a_i\,]$. Because no hard cap is applied, compliance measures whether the model *voluntarily* respects the budget statement. Finally, we decompose the change between the baseline and budget-aware runs into four pairwise outcome counts: *kept correct* (correct in both runs), *lost correct* (correct at baseline, wrong with the banner), *newly correct* (wrong at baseline, correct with the banner), and *still wrong* (incorrect in both runs). The aggregate counts are reported in Figure 6; the per-dataset breakdown is in Figure 7.

Stating the budget improves accuracy, but compliance is poor. For all three models, $\text{Acc}_{\text{budget-aware}}$ exceeds $\text{Acc}_{\text{baseline}}$ in aggregate ($+1.9$ pp for Gemini 2.5 Flash, $+4.9$ pp for Kimi K2.5, and $+8.3$ pp for GLM 5.1; Figure 6). Yet only Gemini 2.5 Flash exhibits a meaningful compliance rate (36.6%); Kimi K2.5 stays within its self-declared budget on only 6.0% of problems and GLM 5.1 on 16.0%. The accuracy gains therefore arise even when the model overruns its stated budget - the budget-aware instruction functions more as a *conciseness prior* than as a binding constraint. This is consistent with the gap between the advisory and enforced regimes reported in §5: models can plan against a budget statement but rarely execute within it.

Per-dataset patterns. Figure 7 shows that the budget-aware lift is concentrated on the structured-reasoning datasets (AIME, GPQA, LCB) where models also have higher baseline accuracy. On HLE, all three models remain near floor in every column. GLM 5.1's lift on AIME is the largest single effect in the experiment: $\text{Acc}_{\text{baseline}} = 0.10 \rightarrow \text{Acc}_{\text{budget-aware}} = 0.61$. This is consistent with GLM 5.1's leading η_U in the main results: when the model is told to be concise, it commits to its final answer earlier and avoids the long, drifting traces that cost it correctness in the unconstrained setting.

D.2. Prompt-Sensitivity Ablation

To check whether η depends on the prompt, we re-ran the planning step under three framings of the same task: A (neutral, used elsewhere in the paper), B (autonomous agent), and C (timed exam). The Rules block, problem text, budget, output schema, and ordering instruction are identical across the three; only the opening paragraph differs.

Budget-aware banner (prepended to the standard solver prompt)

Variant A — Neutral (default) You are given a set of `{domain}` to attempt under a fixed output-token budget. You do not have to attempt every problem. Deciding which problems to attempt and in what order is part of the task.

Budget-aware banner (prepended to the standard solver prompt)

Variant B — Autonomous Agent You are an autonomous planning agent operating under a fixed compute budget of `{B}` output tokens. A set of `{domain}` has been assigned to you. You cannot expect to complete every assignment within budget. Your job is to select which to execute and in what order so as to maximize total reward. Some assignments may be infeasible for you — committing tokens to those yields no reward.

Budget-aware banner (prepended to the standard solver prompt)

Variant C — Timed Exam You are taking a timed exam with a fixed output-token budget of `{B}` across all questions. A set of `{domain}` is in front of you. You do not have to attempt every question — some may be too difficult or unanswerable within the time available. Decide which questions to attempt and in what order so as to maximize your total score.

Table 1. η_U under each prompt variant. Each value is the mean over 8 datasets (*AIME*, *GPQA Diamond*, *LiveCodeBench*, *HLE-CS/AI*, *HLE-Physics*, *HLE-Humanities*, *HLE-Other*, *HLE-Engineering*) at that α. **range** = $\max - \min$ across A, B, C; cells with range ≥ 0.10 are bolded.

Model	Mode	α	η_U^A	η_U^B	η_U^C	range
Kimi K2.5	nothink	0.25	0.615	0.613	0.575	0.040
		0.50	0.530	0.659	0.554	**0.130**
		0.75	0.681	0.781	0.776	**0.100**
		1.00	0.978	1.000	1.000	0.022
	think	0.25	0.432	0.497	0.412	0.085
		0.50	0.463	0.477	0.526	0.063
		0.75	0.333	0.363	0.271	0.092
		1.00	0.778	0.889	0.506	**0.383**
Qwen 3.5 27B	nothink	0.25	0.446	0.428	0.445	0.018
		0.50	0.379	0.353	0.443	0.090
		0.75	0.507	0.523	0.556	0.049
		1.00	0.933	0.956	0.933	0.022
	think	0.25	0.414	0.361	0.391	0.053
		0.50	0.333	0.448	0.334	**0.115**
		0.75	0.287	0.415	0.475	**0.188**
		1.00	0.889	0.889	0.911	0.022
Cells with range < 0.10*:*						11 / 16
Median range:						0.077

We ran the full grid: 2 models, 2 modes, 8 datasets, 4 budget levels, 3 variants - 16 (model, mode, α) cells per variant under each regime.

Results. For η_U, 11 of 16 cells stay within 0.1 across variants; for η_E, 14 of 16 (Tables 1, 2). The two largest spreads in η_U are both in think-mode at extreme budgets: Kimi K2.5/think at $\alpha = 1.0$ (range 0.383, with Variant C alone dropping to 0.506) and Qwen 3.5 27B/think at $\alpha = 0.75$ (range 0.188).

The $\alpha = 0.75$ case is not random. Variant A gives the lowest η_U in 3 of 4 think-mode cells at $\alpha \geq 0.5$. Variants B and C tell the model some problems may be infeasible; A does not. Reasoning models seem to need that cue - without it they attempt too many problems. Non-thinking modes show no such pattern (median range 0.040).

Ranking stability. Table 3 reports Kendall's τ_b on the four-model ranking induced by each variant pair. With $n = 4$, τ_b can only take the values $\{-1, -2/3, -1/3, 0, 1/3, 2/3, 0.913, 1\}$: a single rank swap drops τ from 1 to 0.667. We therefore report raw τ rather than apply a threshold. All 24 pairwise values are $\geq 2/3$, and the swapped models differ by < 0.1 in η - within the cell-level spread we already report. The rankings are stable.

Taken together - 78% of cells within 0.1 spread, all $\tau \geq 2/3$, and the largest disagreement isolated to one (model, mode, α) corner - η_U is robust to the prompt framings tested.

E. Model Configurations

Table 4 lists the 20 models evaluated in the main experiments, together with their providers, parameter counts where disclosed, reasoning support, and the exact API or checkpoint identifiers used. Open-weights models were served locally via vLLM; closed models were accessed through their respective provider APIs. All queries used temperature 0 with provider-default sampling parameters; reasoning effort was set to the provider default for models that expose a tunable reasoning level.

Reasoning modes. Each model is evaluated in up to two modes: *standard inference* (no extended reasoning) and *extended reasoning* (the model's native long chain-of-thought or thinking mode, where supported). Models without a thinking toggle are evaluated only in standard inference. The "Reasoning" column in Table 4 indicates which modes were run for each model.

Table 2. η_E under each prompt variant, same layout as Table 1. Negative values reflect systematic under-allocation in the enforced regime — a model property, the same under all three prompts.

Model	Mode	α	η_E^A	η_E^B	η_E^C	range
Kimi K2.5	nothink	0.25	0.152	0.152	0.148	0.004
		0.50	−0.254	−0.232	−0.245	0.022
		0.75	−1.421	−1.352	−1.421	0.069
		1.00	0.356	0.356	0.356	0.000
	think	0.25	−0.192	−0.194	−0.237	0.045
		0.50	−0.712	−0.692	−0.700	0.020
		0.75	−2.193	−2.211	−2.282	0.089
		1.00	−0.367	−0.433	−0.532	**0.165**
Qwen 3.5 27B	nothink	0.25	0.089	0.157	0.143	0.068
		0.50	−0.161	−0.152	−0.159	0.008
		0.75	−1.180	−1.113	−1.151	0.067
		1.00	0.222	0.200	0.222	0.022
	think	0.25	−0.110	−0.111	−0.105	0.005
		0.50	−0.479	−0.552	−0.540	0.073
		0.75	−1.568	−1.597	−1.782	**0.214**
		1.00	0.311	0.333	0.244	0.089
Cells with range < 0.10*:*						14 / 16
Median range:						0.057

Table 3. Kendall's τ_b between the four-model rankings induced by each variant pair. *max range* is the largest across-variant spread in any (model, mode) row at that α.

Regime	α	$\tau(A,B)$	$\tau(A,C)$	$\tau(B,C)$	min τ	max range
η_U	0.25	0.667	1.000	0.667	0.667	0.085
	0.50	0.667	1.000	0.667	0.667	0.130
	0.75	0.667	0.667	1.000	0.667	0.188
	1.00	0.913	1.000	0.913	0.913	0.383
η_E	0.25	0.667	1.000	0.667	0.667	0.068
	0.50	1.000	1.000	1.000	1.000	0.073
	0.75	1.000	1.000	1.000	1.000	0.214
	1.00	1.000	1.000	1.000	1.000	0.165

Sampling and decoding. All models were queried at temperature 0. For models accessed via provider APIs, we used each provider's default values for top-p, top-k, frequency penalty, and presence penalty unless these parameters were unsupported at temperature 0. For locally served models, decoding used vLLM defaults with greedy sampling. Maximum output length was set to a value sufficient to capture the longest plan and reasoning trace observed for each model in pilot runs; exact values per model are recorded in the released configuration files.

Reasoning effort settings. For models that expose a tunable reasoning level (e.g., reasoning effort, thinking budget), we used the provider's default reasoning setting in extended-reasoning mode and disabled reasoning entirely in standard-inference mode.

Table 4. Models evaluated in TRIAGE. *Weights*: O = open, C = closed. *Params*: parameter count where disclosed; for mixture-of-experts models, total / active. *Reasoning*: S = standard inference only, S+E = both standard and extended reasoning modes evaluated. *Identifier*: provider API string or HuggingFace checkpoint used.

Model	Provider	Weights	Params	Reasoning	Identifier
Qwen 2.5 series (Alibaba)					
Qwen 2.5 3B	Alibaba	O	3B	S	`Qwen/Qwen2.5-3B-Instruct`
Qwen 2.5 7B	Alibaba	O	7B	S	`Qwen/Qwen2.5-7B-Instruct`
Qwen 2.5 14B	Alibaba	O	14B	S	`Qwen/Qwen2.5-14B-Instruct`
Qwen 2.5 32B	Alibaba	O	32B	S	`Qwen/Qwen2.5-32B-Instruct`
Qwen 3 series (Alibaba)					
Qwen 3 4B	Alibaba	O	4B	S+E	`Qwen/Qwen3-4B`
Qwen 3 8B	Alibaba	O	8B	S+E	`Qwen/Qwen3-8B`
Qwen 3 14B	Alibaba	O	14B	S+E	`Qwen/Qwen3-14B`
Qwen 3 32B	Alibaba	O	32B	S+E	`Qwen/Qwen3-32B`
Qwen 3.5 series (Alibaba)					
Qwen 3.5 4B	Alibaba	O	4B	S+E	`Qwen/Qwen3.5-4B`
Qwen 3.5 9B	Alibaba	O	9B	S+E	`Qwen/Qwen3.5-9B`
Qwen 3.5 27B	Alibaba	O	27B	S+E	`Qwen/Qwen3.5-27B`
Qwen 3.5 35B-A3B	Alibaba	O	35B / 3B	S+E	`Qwen/Qwen3.5-35B-A3B`
Frontier and other open models					
Claude Haiku 4.5	Anthropic	C	—	S+E	`claude-haiku-4-5`
DeepSeek V3	DeepSeek	O	671B / 37B	S+E	`deepseek-chat`
Gemini 2.5 Flash	Google	C	—	S+E	`gemini-2.5-flash`
GLM 5.1	Zhipu	O	—	S+E	`glm-5.1`
GPT-5 Mini	OpenAI	C	—	S+E	`gpt-5-mini`
Kimi K2.5	Moonshot	O	—	S+E	`kimi-k2.5`
GPT-OSS 20B	OpenAI	O	20B	S+E	`openai/gpt-oss-20b`
GPT-OSS 120B	OpenAI	O	120B	S+E	`openai/gpt-oss-120b`

	AIME	GPQA	LCB	HLE
Qwen 2.5 3B [std]	0.03	0.29	0.06	0.03
Qwen 2.5 7B [std]	0.10	0.32	0.20	0.03
Qwen 2.5 14B [std]	0.13	0.47	0.21	0.04
Qwen 2.5 32B [std]	0.20	0.47	0.28	0.03
Qwen 3 4B [ext]	0.50	0.46	0.27	0.04
Qwen 3 8B [ext]	0.50	0.51	0.30	0.04
Qwen 3 14B [ext]	0.57	0.52	0.33	0.04
Qwen 3 32B [ext]	0.59	0.62	0.32	0.04
Qwen 3.5 4B [std]	0.25	0.59	0.27	0.04
Qwen 3.5 4B [ext]	0.23	0.64	0.20	0.03
Qwen 3.5 9B [std]	0.42	0.65	0.28	0.03
Qwen 3.5 9B [ext]	0.30	0.68	0.22	0.03
Qwen 3.5 27B [std]	0.50	0.71	0.39	0.05
Qwen 3.5 27B [ext]	0.57	0.77	0.30	0.04
Qwen 3.5 35B-A3B [std]	0.38	0.54	0.35	0.04
Qwen 3.5 35B-A3B [ext]	0.54	0.78	0.32	0.05
Claude Haiku 4.5 [std]	0.45	0.61	0.41	0.03
Claude Haiku 4.5 [ext]	0.67	0.70	0.52	0.02
DeepSeek V3 [std]	0.57	0.71	0.38	0.04
DeepSeek V3 [ext]	0.72	0.73	0.39	0.08
Gemini 2.5 Flash [std]	0.28	0.41	0.34	0.03
Gemini 2.5 Flash [ext]	0.62	0.70	0.33	0.05
GLM 5.1 [std]	0.10	0.31	0.18	0.01
GLM 5.1 [ext]	0.55	0.61	0.19	0.05
GPT-5 Mini [std]	0.52	0.69	0.47	0.05
GPT-5 Mini [ext]	0.77	0.79	0.41	0.06
Kimi K2.5 [std]	0.13	0.21	0.20	0.03
Kimi K2.5 [ext]	0.42	0.62	0.28	0.07
GPT-OSS 20B [ext]	0.67	0.56	0.42	0.03
GPT-OSS 120B [ext]	0.78	0.73	0.49	0.03

Figure 3. Per-(model, mode) accuracy on each benchmark. Accuracy is the proportion of solvable problems and is independent of budget α. Cells use a sequential white→teal scale; horizontal lines separate model families; the [std] / [ext] suffix denotes standard inference versus extended reasoning.

	$\alpha = 0.25$				$\alpha = 0.5$				$\alpha = 0.75$				$\alpha = 1.0$			
	AIME	GPQA	LCB	HLE	AIME	GPQA	LCB	HLE	AIME	GPQA	LCB	HLE	AIME	GPQA	LCB	HLE
Qwen 2.5 3B [std]	-0.33	-0.07	0.01	0.36	-0.95	-0.18	-0.26	0.16	-2.65*	-0.67	-1.51*	-0.25	0.00	0.14	0.33	0.60
Qwen 2.5 7B [std]	-0.09	-0.14	-0.01	0.53	-0.31	-0.08	-0.28	0.32	-2.11*	-0.64	-0.28	-0.27	0.50	0.86	0.33	0.60
Qwen 2.5 14B [std]	-0.05	0.01	0.03	0.23	0.41	-0.10	-0.22	-0.11	-1.26*	-0.56	-1.63*	-0.25	0.00	0.86	0.00	0.67
Qwen 2.5 32B [std]	0.27	-0.15	0.29	0.49	0.20	-0.48	-0.24	0.35	-0.40	-1.13*	-1.13*	-0.21	0.50	0.14	0.00	0.63
Qwen 3 4B [ext]	0.32	0.11	-0.18	0.33	0.34	0.18	-0.91	0.23	0.00	0.29	-2.46*	-0.10	0.00	0.67	0.00	0.80
Qwen 3 8B [ext]	0.30	0.09	-0.27	0.24	—	0.06	-0.94	0.25	0.25	0.04	-2.42*	0.69	0.00	0.17	0.00	0.65
Qwen 3 14B [ext]	0.67	0.08	-0.25	0.29	0.39	0.07	-0.21	0.36	0.30	-0.06	-2.03*	0.05	1.00	0.67	0.25	0.88
Qwen 3 32B [ext]	0.50	0.14	0.25	0.38	0.36	0.15	-0.13	0.24	-0.26	-0.35	-0.58	0.17	1.00	0.50	0.00	0.71
Qwen 3.5 4B [std]	0.01	0.05	-0.34	0.27	-0.47	-0.16	-0.96	0.06	-1.49*	-0.13	-2.71*	-0.13	0.00	0.86	0.00	0.73
Qwen 3.5 4B [ext]	0.33	-0.15	-0.14	0.22	-0.95	-0.09	-0.95	0.00	0.53	-0.27	-2.83*	0.02	0.00	0.67	0.00	0.80
Qwen 3.5 9B [std]	-0.01	0.14	0.06	0.48	-0.04	-0.09	-0.00	0.36	-0.51	0.16	-0.39	0.44	0.50	0.71	0.17	0.97
Qwen 3.5 9B [ext]	-0.36	0.11	-0.21	0.55	-0.22	-0.07	-0.63	0.36	0.06	-0.08	-0.67	0.53	1.00	1.00	0.20	0.91
Qwen 3.5 27B [std]	0.36	0.25	0.41	0.43	0.43	0.09	0.46	0.41	0.52	-0.25	0.37	0.66	1.00	1.00	0.83	0.97
Qwen 3.5 27B [ext]	0.53	0.23	0.19	0.51	-0.05	0.20	-0.26	0.58	-1.56*	0.25	-0.90	0.91	0.50	1.00	0.50	0.91
Qwen 3.5 35B-A3B [std]	-0.32	-0.54	0.03	0.04	-0.51	-0.94	-0.28	-0.36	-1.32*	-1.98*	-0.36	-1.74*	0.50	0.00	0.67	0.31
Qwen 3.5 35B-A3B [ext]	-0.65	0.26	0.43	0.24	0.02	-0.15	-0.11	0.71	0.06	0.28	0.02	0.66	0.00	0.67	0.67	0.90
Claude Haiku 4.5 [std]	-0.12	0.44	0.45	0.53	-0.20	0.21	0.33	0.37	0.48	-0.33	0.22	0.32	0.50	0.14	0.67	0.77
Claude Haiku 4.5 [ext]	0.80	0.55	0.42	0.63	0.41	0.17	0.23	0.77	-1.09*	-0.15	-0.24	0.25	0.50	0.43	0.50	0.97
DeepSeek V3 [std]	0.46	0.25	0.21	0.32	0.21	0.20	0.37	0.55	-0.22	0.06	0.32	0.03	0.50	0.71	0.33	0.90
DeepSeek V3 [ext]	0.68	0.53	0.12	0.48	0.69	0.44	-0.69	0.49	-1.07*	0.21	-1.19*	-0.05	1.00	0.86	0.50	0.80
Gemini 2.5 Flash [std]	0.37	0.18	0.51	0.65	0.54	0.10	0.49	0.70	0.52	-0.40	0.47	0.76	0.00	0.57	0.17	0.93
Gemini 2.5 Flash [ext]	0.60	0.45	0.50	0.32	0.03	0.29	0.29	0.50	-2.24*	-0.51	-0.42	0.64	0.00	0.00	0.00	0.77
GLM 5.1 [std]	0.51	0.49	0.85	0.82	0.76	0.68	0.88	0.71	0.56	0.52	0.84	0.87	1.00	1.00	1.00	1.00
GLM 5.1 [ext]	0.48	0.54	0.93	0.39	0.57	0.72	0.93	0.32	0.26	0.55	1.00	0.31	0.50	0.29	1.00	0.74
GPT-5 Mini [std]	0.03	0.24	-0.10	0.30	0.12	-0.19	-0.15	0.26	-0.21	-0.09	-2.23*	0.10	1.00	0.86	0.67	0.83
GPT-5 Mini [ext]	-0.15	0.32	-0.20	0.27	-0.02	0.21	-0.44	0.28	-0.81	0.18	-1.58*	-0.31	1.00	0.86	0.33	0.73
Kimi K2.5 [std]	0.74	0.33	0.81	0.60	0.23	0.22	0.77	0.54	0.63	0.44	0.92	0.69	1.00	0.71	0.83	0.87
Kimi K2.5 [ext]	0.67	0.39	0.67	0.35	0.58	0.40	0.64	0.37	0.63	0.01	0.37	0.39	1.00	0.14	0.50	0.77
GPT-OSS 20B [ext]	-0.20	0.03	-0.21	0.43	0.06	-0.07	-0.23	0.30	-0.33	-0.05	-0.86	0.31	1.00	0.86	0.40	0.78
GPT-OSS 120B [ext]	-0.15	0.02	-0.25	0.41	0.19	-0.11	-0.80	0.47	0.06	-0.09	-2.64*	-0.11	0.50	0.57	0.00	0.84

Figure 4. Triage skill in the advisory-budget regime, η_U, per (model, α, dataset). $\eta_U = 1$ matches oracle planning, $\eta_U = 0$ matches a random baseline, $\eta_U < 0$ is worse than random. Cells use a diverging coral→navy palette capped at $[-1, 1]$; values outside the cap retain their numeric label and are marked with $*$. Bold vertical rules separate α blocks. The missing cell due to unparsable output (Qwen 3 8B think on AIME at $\alpha = 0.5$) is shown as '—'.

	$\alpha = 0.25$				$\alpha = 0.5$				$\alpha = 0.75$				$\alpha = 1.0$			
	AIME	GPQA	LCB	HLE	AIME	GPQA	LCB	HLE	AIME	GPQA	LCB	HLE	AIME	GPQA	LCB	HLE
Qwen 2.5 3B [std]	-0.33	-0.07	0.12	0.19	-0.95	-0.26	-0.49	-0.03	-2.65*	-1.20*	-1.51*	-0.82	0.00	0.00	0.33	0.33
Qwen 2.5 7B [std]	-0.31	0.00	-0.06	0.44	-0.63	-0.03	-0.44	0.21	-2.74*	-1.04*	-0.66	-0.61	0.00	0.00	0.17	0.57
Qwen 2.5 14B [std]	-0.32	-0.14	-0.04	0.08	-0.75	-0.55	-0.35	-0.27	-1.63*	-1.79*	-2.16*	-1.24*	0.00	0.00	0.00	0.37
Qwen 2.5 32B [std]	-0.18	-0.49	-0.03	0.18	-0.55	-1.02*	-0.51	-0.16	-2.35*	-2.38*	-1.20*	-1.08*	0.00	0.00	0.00	0.43
Qwen 3 4B [ext]	-0.48	-0.39	-0.33	0.03	-1.00*	-0.97	-0.96	-0.13	-2.73*	-2.72*	-2.68*	-1.14*	0.00	0.00	0.00	0.20
Qwen 3 8B [ext]	-0.46	-0.31	-0.33	-0.01	—	-0.84	-0.94	-0.21	-2.75*	-1.67*	-2.67*	-0.79	0.00	0.00	0.00	0.31
Qwen 3 14B [ext]	-0.63	-0.45	-0.34	0.12	-0.95	-0.97	-0.95	-0.34	-2.75*	-2.64*	-2.77*	-1.22*	0.00	0.00	0.00	0.32
Qwen 3 32B [ext]	-0.56	-0.47	-0.32	0.22	-1.06*	-0.84	-0.94	-0.37	-2.79*	-2.60*	-2.75*	-1.17*	0.00	0.00	0.00	0.29
Qwen 3.5 4B [std]	-0.32	-0.32	-0.34	0.09	-0.96	-0.80	-0.96	-0.35	-2.78*	-2.40*	-2.71*	-1.53*	0.00	0.00	0.00	0.30
Qwen 3.5 4B [ext]	-0.34	-0.59	-0.31	0.07	-0.95	-1.14*	-0.95	-0.10	-2.78*	-2.71*	-2.83*	-0.78	0.00	0.00	0.00	0.47
Qwen 3.5 9B [std]	-0.41	-0.41	0.10	0.30	-0.83	-0.81	-0.51	-0.02	-2.76*	-2.29*	-2.00*	-0.94	0.00	0.00	0.00	0.47
Qwen 3.5 9B [ext]	-0.36	-0.68	-0.32	0.29	-0.95	-1.21*	-0.80	-0.22	-1.51*	-2.47*	-2.85*	-1.13*	0.00	0.00	0.00	0.52
Qwen 3.5 27B [std]	-0.21	0.21	0.15	0.07	-0.22	0.01	0.15	-0.25	-0.64	-0.83	-0.48	-1.37*	0.00	0.00	0.00	0.33
Qwen 3.5 27B [ext]	-0.64	-0.75	-0.39	0.08	-0.88	-1.18*	-0.95	-0.36	-2.09*	-1.92*	-2.58*	-1.15*	0.00	0.00	0.00	0.32
Qwen 3.5 35B-A3B [std]	-0.38	-0.63	-0.03	-0.01	-0.96	-0.94	-0.47	-0.46	-2.76*	-1.98*	-1.21*	-1.74*	0.00	0.00	0.00	0.24
Qwen 3.5 35B-A3B [ext]	-0.65	-0.73	-0.36	-0.07	-0.83	-1.21*	-0.95	-0.29	-1.59*	-2.29*	-2.30*	-1.58*	0.00	0.00	0.00	0.45
Claude Haiku 4.5 [std]	-0.59	-0.66	-0.35	0.32	-0.73	-0.44	-0.62	0.23	-1.17*	-0.89	-1.46*	0.10	0.00	0.00	0.00	0.63
Claude Haiku 4.5 [ext]	-0.68	-0.71	-0.54	0.41	-0.83	-0.94	-0.92	0.16	-2.75*	-2.42*	-2.51*	-0.55	0.00	0.00	0.00	0.53
DeepSeek V3 [std]	-0.22	-0.42	-0.12	0.27	-0.39	-0.63	-0.18	-0.10	-0.97	-1.47*	-0.74	-0.76	0.00	0.00	0.00	0.47
DeepSeek V3 [ext]	-0.56	-0.51	-0.32	-0.11	-1.11*	-1.06*	-0.94	-0.61	-2.73*	-2.54*	-2.76*	-2.10*	0.00	0.00	0.00	0.17
Gemini 2.5 Flash [std]	-0.02	0.23	0.39	0.42	0.07	0.07	0.36	0.28	-1.06*	-0.69	0.05	-0.32	0.00	0.00	0.17	0.57
Gemini 2.5 Flash [ext]	-1.14*	-0.81	-0.14	0.03	-2.04*	-1.37*	-0.42	-0.38	-3.30*	-2.92*	-1.56*	-1.42*	0.00	0.00	0.00	0.33
GLM 5.1 [std]	-0.30	-0.32	-0.26	0.60	-0.71	-0.82	-0.62	0.42	-1.78*	-2.17*	-1.93*	-0.11	0.00	0.00	0.00	0.70
GLM 5.1 [ext]	-0.44	-0.41	0.01	-0.17	-1.01*	-1.02*	-0.29	-0.78	-2.73*	-2.53*	-1.73*	-2.20*	0.00	0.00	0.00	0.13
GPT-5 Mini [std]	-0.39	-0.25	-0.21	-0.00	-0.72	-0.12	-0.31	-0.45	-1.28*	-0.80	-2.50*	-1.58*	0.00	0.00	0.00	0.23
GPT-5 Mini [ext]	-0.57	-0.17	-0.26	-0.09	-0.53	-0.56	-0.64	-0.62	-2.87*	-2.31*	-2.21*	-2.07*	0.00	0.00	0.00	0.17
Kimi K2.5 [std]	-0.31	-0.28	-0.34	0.38	-0.96	-0.90	-0.94	0.09	-2.75*	-2.71*	-2.79*	-0.74	0.00	0.00	0.00	0.53
Kimi K2.5 [ext]	-0.43	-0.48	-0.33	-0.08	-0.88	-0.97	-0.97	-0.64	-2.58*	-2.68*	-2.54*	-2.04*	-14.21*	0.00	0.00	0.17
GPT-OSS 20B [ext]	-0.46	-0.33	-0.30	0.28	-0.69	-0.90	-0.82	-0.07	-2.30*	-2.43*	-2.07*	-1.00*	0.00	0.00	0.00	0.37
GPT-OSS 120B [ext]	-0.59	-0.26	-0.29	0.30	-0.36	-0.69	-0.93	0.27	-1.35*	-1.57*	-2.64*	-0.81	0.00	0.00	0.00	0.52

Figure 5. Triage skill in the enforced-budget regime, η_E, per (model, α, dataset). Color encoding identical to Figure 4. The $\alpha = 1.0$ column block saturates near zero because $V_{\text{oracle}} \approx V_{\text{random}}$ at full budget.

Model	N	$\text{Acc}_{\text{baseline}}$	$\text{Acc}_{\text{budget-aware}}$	compliance	newly correct	lost correct	kept correct	still wrong
Gemini 2.5 Flash	1192	0.143	0.162	0.366	57	34	136	965
Kimi K2.5	1243	0.084	0.133	0.060	74	13	91	1065
GLM 5.1	1266	0.088	0.171	0.160	114	10	102	1040

Figure 6. **Budget-aware re-solve, aggregate per model.** Cells colored by intensity within each metric. N is the number of (problem, allocation) pairs re-issued. *Compliance* is the fraction of problems for which the model's actual output length stays within its self-declared budget a_i. The four right-hand counts (newly correct, lost correct, kept correct, still wrong) sum to N and decompose the change between $\text{Acc}_{\text{baseline}}$ (original baseline solve) and $\text{Acc}_{\text{budget-aware}}$ (baseline solver prompt augmented with an instruction to answer within the model's own a_i).

		AIME	GPQA	LCB	HLE
Gemini 2.5 Flash	$\text{Acc}_{\text{baseline}}$	0.304	0.405	0.384	0.030
	$\text{Acc}_{\text{budget-aware}}$	0.261	0.508	0.384	0.036
	compliance	0.587	0.636	0.580	0.252
Kimi K2.5	$\text{Acc}_{\text{baseline}}$	0.119	0.211	0.199	0.026
	$\text{Acc}_{\text{budget-aware}}$	0.237	0.412	0.267	0.029
	compliance	0.153	0.041	0.080	0.053
GLM 5.1	$\text{Acc}_{\text{baseline}}$	0.102	0.313	0.185	0.013
	$\text{Acc}_{\text{budget-aware}}$	0.610	0.510	0.225	0.047
	compliance	0.458	0.232	0.354	0.081

Figure 7. **Budget-aware re-solve, per (model $\times$ dataset).** HLE sub-domains are aggregated by problem-count weighted mean. Rows within each model block: original baseline accuracy, accuracy with the budget banner, and compliance rate. Color encoding follows Figure 6.