PLAN AND BUDGET: EFFECTIVE AND EFFICIENT TEST-TIME SCALING ON LARGE LANGUAGE MODEL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) have achieved remarkable success in complex reasoning tasks, but their inference remains computationally inefficient. We observe a common failure mode in many prevalent LLMs, *overthinking*, where models generate verbose and tangential reasoning traces even for simple queries. Recent works have tried to mitigate this by enforcing fixed token budgets, however, this can lead to *underthinking*, especially on harder problems. Through empirical analysis, we identify that this inefficiency often stems from unclear problem-solving strategies. To formalize this, we develop a theoretical model, **BAM** (**B**udget **A**llocation **M**odel), which models reasoning as a sequence of sub-questions with varying uncertainty, and introduce the $\mathcal{E}^3$ metric to capture the trade-off between correctness and computation efficiency. Building on theoretical results from BAM, we propose **PLAN-AND-BUDGET**, a model-agnostic, test-time framework that decomposes complex queries into sub-questions and allocates token budgets based on estimated complexity using adaptive scheduling. PLAN-AND-BUDGET improves reasoning efficiency across a range of tasks and models, achieving up to **70%** accuracy gains, **39%** token reduction, and **193.8%** improvement in $\mathcal{E}^3$. Notably, it elevates a smaller model (DS-Qwen-32B) to match the efficiency of a larger model (DS-LLaMA-70B), demonstrating PLAN-AND-BUDGET's ability to close performance gaps without retraining. Our code is available at Pland-and-Budget.

1 INTRODUCTION

Large Language Models (LLMs) exhibit strong generalization capabilities, enabling them to perform a wide range of tasks, such as mathematical problem solving (Ahn et al., 2024; Imani et al., 2023), scientific question answering (Huang et al., 2024; Lu et al., 2022), and structured reasoning (Guo et al., 2025; Wei et al., 2022), without task-specific retraining. Recent advances in test-time computation like Chain-of-Thought (CoT) prompting (Wei et al., 2022), self-consistency (Wang et al., 2023), and tool-augmented inference (Chen et al., 2023) have significantly enhanced their performance on complex, multi-step reasoning tasks. These enhancements have paved the way for LLMs to be increasingly deployed in high-stakes domains such as education (Golshan & Academy, 2023), finance (Yang et al., 2023), law (Katz et al., 2024), and scientific research (Taylor et al., 2022), where robust reasoning at inference time is critical.

Despite these advances, deploying LLMs in real-world settings introduces new challenges, particularly in scenarios requiring deliberative reasoning under compute and time constraints. A key issue is the lack of calibrated reasoning behavior during inference. Although LLMs are proficient in multi-step reasoning, they often struggle to regulate how much reasoning effort is appropriate for a task. This miscalibration manifests in two major failure modes: *overthinking* (Sui et al., 2025; Chen et al., 2024; Turpin et al., 2023), where models generate unnecessarily long and tangential reasoning paths, even for simple queries, incurring excessive computational cost without improving accuracy; and *underthinking* (Wang et al., 2025a; Wei et al., 2022), where models terminate reasoning prematurely, sacrificing correctness to conserve resources. Recent methods (Lee et al., 2025; Xu et al., 2025; Han et al., 2024) have attempted to mitigate overthinking by introducing hard token constraints (e.g., "using fewer than B tokens" in the prompt). While these strategies may be effective on simpler tasks, they often degrade performance on complex queries by inducing underthinking, highlighting the

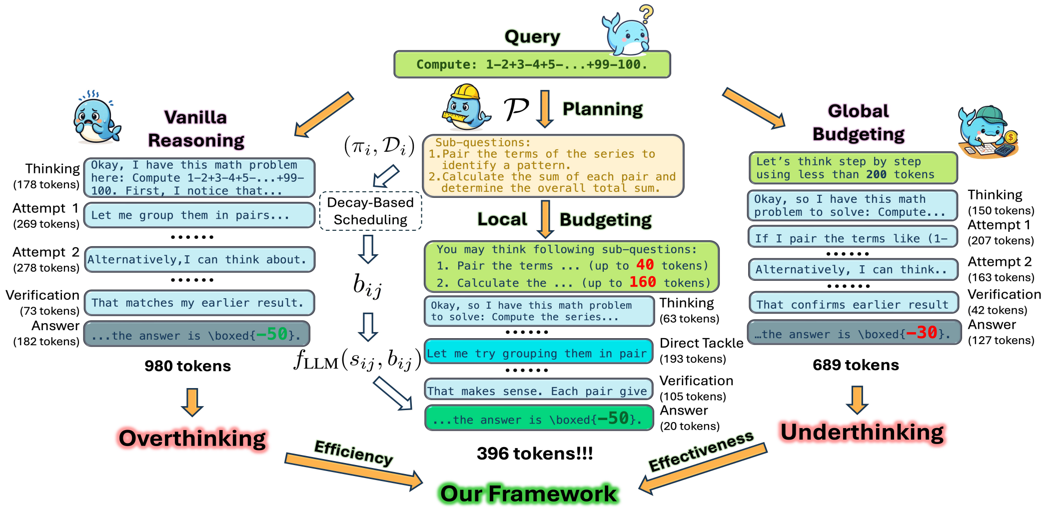

Figure 1: Illustration of REASONING MISCALIBRATION. Vanilla reasoning overthinks and wastes tokens; global budgeting underthinks and fails. Our method combines planning and local budgeting to guide structured, efficient reasoning, achieving the correct answer with fewer tokens.

limitations of fixed, non-adaptive approaches. To the best of our knowledge, there is limited work that has systematically addressed both overthinking and underthinking in a unified framework.

In this paper, we take the first step toward closing this gap. With a comprehensive empirical study of test-time reasoning behavior in state-of-the-art LLMs ranging from 32B to 70B parameters, we uncover a pervasive phenomenon we term "REASONING MISCALIBRATION", a failure mode where models exhibit unregulated inference depth during reasoning. This miscalibration manifests as either overthinking, where the model engages in unnecessary and tangential reasoning, or underthinking, where reasoning terminates prematurely. Our study reveals that reasoning miscalibration is frequently triggered by two types of queries: (1) trivial-but-ambiguous queries, which elicit diffuse token distributions and lead to speculative reasoning; and (2) hard-and-rare queries, where models engage in shallow trial-and-error without meaningful convergence. These findings raise a central research question: ***How can we characterize the internal reasoning and inference mechanisms of LLMs, and how can we guide them to allocate computation adaptively based on task complexity?***

To answer this, we analyze reasoning miscalibration through the lens of uncertainty, quantified by the entropy of the model's marginal next-token distribution at each step. This distribution reflects the model's belief over possible continuations, where higher entropy signals indecision or ambiguity. We find that high entropy often correlates with unnecessarily deep reasoning (i.e., overthinking), while low entropy observed at early steps often leads to premature truncation of reasoning (i.e., underthinking). These suggest that uncertainty can serve as a valuable signal for dynamically adjusting reasoning depth. Motivated by this, we introduce the **Budget Allocation Model (BAM)**, a theoretical resource allocation model that aligns computation with uncertainty. BAM conceptualizes reasoning as a sequence of sub-problems, each characterized by varying degrees of uncertainty, and allocates greater computational budget to sub-questions with higher uncertainty, enabling more calibrated and efficient inference. From this perspective, we derive two key principles for effective reasoning: (1) reasoning should be *structured*: decomposing complex queries into smaller, targeted sub-questions helps reduce speculative exploration; and (2) computation should be *adaptive*: early reasoning steps typically bear higher uncertainty and thus merit greater computational focus.

Building on these principles, we propose a novel compute-efficient reasoning strategy, called **PLAN-AND-BUDGET**, which consists of two stages: **Plan** and **Budget**. In the **Plan** Step, the model decomposes the original query into a sequence of sub-questions, providing a soft scaffold for structured reasoning. In the **Budget** Step, we apply simplified scheduling strategies that dynamically assign token budgets to each sub-question, guided by its uncertainty pattern, following the BAM principle. To evaluate our approach, we introduce $\mathcal{E}^3$, namely the **E**fficiency-Aware **E**ffectivenss **E**valuation Score that captures the trade-off between reasoning accuracy and computational cost. Unlike conventional efficiency metrics that overlook output quality, $\mathcal{E}^3$ offers a more robust, holistic measure of inference performance.

We evaluate our method through extensive experiments across four state-of-the-art LLMs, including DeepSeek-R1 Distill-Qwen-32B (DS-Qwen-32B) (Guo et al., 2025), QwQ-32B (Team, 2025), DeepSeek-R1 Distill-Llama-70B (DS-LLaMA-70B) (Guo et al., 2025), and OpenAI o4-mini (OpenAI, 2025) on three representative task domains: mathematical reasoning, instruction following, and agentic planning. Our method is model-agnostic: it requires no retraining or fine-tuning, relying only on prompting and lightweight planning. Despite this simplicity, PLAN-AND-BUDGET consistently improves all LLMs across all benchmarks. We observe downstream accuracy gains of up to **70%**, token usage reductions of up to **39%**, and combined efficiency improvements (as measured by $\mathcal{E}^3$) of up to **193.8%** over strong baselines. An especially notable case comes from the agentic planning task domain, where a smaller DS-Qwen-32B improves from a low $\mathcal{E}^3$ of 0.16 to 0.47 using PLAN-AND-BUDGET—**closing the gap with the larger DS-LLaMA-70B model** ($\mathcal{E}^3$ = 0.50) without planning. This demonstrates that uncertainty-guided planning and budgeting can act as inference-time equalizers, boosting the efficiency and competitiveness of smaller models without retraining. Together, these findings underscore the promise of principled compute allocation for more calibrated, efficient, and accessible LLM inference.

2 RELATED WORKS AND PRELIMINARY

2.1 RELATED WORKS

Scaling Laws. Recent work has explored how test-time computation affects LLM performance, showing that an increased inference budget can reduce failure rates but often suffers from diminishing returns (Snell et al., 2024; Wu et al., 2025; Zeng et al., 2025). Methods like MCTS-Judge (Wang et al., 2025b) and EAG (Mei et al., 2025) demonstrate the benefits of adaptive compute in tasks like code evaluation and multi-hop reasoning. Unlike prior work focusing on simply increasing compute, we investigate how to allocate it efficiently through structured planning and uncertainty-aware budgeting.

Uncertainty. Quantifying uncertainty in deep models is often framed through epistemic vs. aleatoric components (Hüllermeier & Waegeman, 2021), with techniques like MC Dropout (Gal & Ghahramani, 2016), ensembles (Lakshminarayanan et al., 2017), and evidential learning (Huang et al., 2023). Recent work extends these ideas to LLMs via consistency checks and parameter-efficient ensembles (Tonolini et al., 2024; Halbheer et al., 2024). Our work builds on this by using uncertainty decomposition to guide token allocation at inference time, offeringg a novel application of uncertainty for test-time efficiency.

2.2 PRELIMINARY

We begin by summarizing previous work and introducing the key notations used throughout this work. Table 1 lists the symbols relevant to our reasoning formulation.

Table 1: Notation Summary

Symbol	Description
m	Number of sub-questions
x_i	i-th query
s_{ij}	j-th sub-question of query x_i
b_{ij}	Tokens allocated to sub-question s_{ij}
β_{ij}	Complexity of sub-questions_{ij}
B	Total token budget per query
π_i	Decomposition plan for query x_i
w_{ij}	Normalized complexity weight for s_{ij}
γ, ϵ, p	Decay scheduler hyperparameters
c_{ij}	Parameter characterizing epistemic uncertainty reduction

Reasoning Miscalibration in LLMs. While LLMs excel at complex reasoning tasks, they often struggle to regulate how much inference effort is appropriate per query. We refer to this phenomenon as REASONING MISCALIBRATION. It describes a mismatch between task complexity and the depth of reasoning a model performs at test time.

This miscalibration presents itself in two primary modes: (1) Overthinking Sui et al. (2025); Chen et al. (2024); Turpin et al. (2023), where the model engages in excessively verbose or tangential reasoning even for simple queries, incurring unnecessary computational cost and introducing noise or contradictions; and (2) Underthinking Wang et al. (2025a); Wei et al. (2022), where the model prematurely stops reasoning to conserve budget, often yielding incomplete or incorrect answers.

Contrary to the common belief that allocating more decoding tokens leads to better performance, we observe that excessive generation can degrade quality. In our empirical analysis, we show that longer outputs can lead models to wander within the solution space, becoming verbose, redundant, or self-inconsistent. Our findings suggest that REASONING MISCALIBRATION does not stem from a lack of knowledge or model capacity, but rather from the model's inability to dynamically align reasoning effort with a query's evolving informational needs—particularly in response to uncertainty at each step. We leverage a foundational concept of predictive uncertainty Hüllermeier & Waegeman (2021) which decomposes the total uncertainty $\mathcal{U}(x)$ for a given input x into two distinct components:

$$\mathcal{U}(x) = \mathcal{U}_{\text{epistemic}}(x) + \mathcal{U}_{\text{aleatoric}}(x),$$

where $\mathcal{U}_{\text{epistemic}}(x)$ captures uncertainty due to incomplete knowledge (and is reducible through targeted computation), while $\mathcal{U}_{\text{aleatoric}}(x)$ accounts for irreducible ambiguity or noise in the input. Recent work by Falck et al. (2024b) extends this decomposition to LLMs, revealing that LLMs display dynamic uncertainty profiles throughout inference. These evolving patterns offer valuable insights into both the models' reasoning processes and the quality of their generated outputs. We further demonstrate the validity of this decomposition in the LLM setting through formal analysis in Appendix B.

Problem Definition. In multi-step reasoning, this decomposition reveals a crucial insight: REASONING MISCALIBRATION arises from unregulated computational effort across sub-questions with varying uncertainty levels. Some sub-problems demand greater inference depth to reduce epistemic uncertainty, while others—dominated by aleatoric uncertainty—benefit from early termination or concise solutions. Yet current LLMs lack a mechanism to adaptively allocate computation across these stages. This misalignment leads to inefficiency and degraded reasoning quality. The goal is to improve efficiency while mitigating REASONING MISCALIBRATION.

Efficiency-Aware Effectiveness Evaluation: $\mathcal{E}^3$ Score. We introduce the $\mathcal{E}^3$ index as an efficiency-aware metric that jointly captures reasoning quality and computational cost. Rather than treating token usage and accuracy as separate concerns, the $\mathcal{E}^3$ directly quantifies their trade-off:

$$\mathcal{E}^3 = A \cdot \frac{A}{T} = \frac{A^2}{T}.$$

Here, A denotes the average accuracy achieved across a set of queries, and T represents the average number of decoding tokens used per query. Earlier works typically measure efficiency as accuracy per token (Muennighoff et al.; Lee et al.) (i.e., A/T). By weighting this term with accuracy, the $\mathcal{E}^3$ emphasizes correctness, discouraging degenerate strategies that minimize token usage at the cost of quality. In doing so, it reflects how well a model aligns its computational effort with task complexity, rewarding those that invest more where needed and conserve resources otherwise. Thus, the $\mathcal{E}^3$ provides a principled evaluation framework for assessing whether a model mitigates REASONING MISCALIBRATION while maximizing reasoning efficiency. To address this, we now formalize our target problem as follows:

Problem 1. ***LLM Reasoning Calibration***
Given: *(1) A set of complex queries $\{x_1, \ldots, x_n\}$, where each x_i can be decomposed into a sequence of m sub-questions; and (2) A total token budget B_i for each query x_i.*
Find: *A computation strategy that maximizes the efficiency-aware score $\mathcal{E}^3 = \frac{A^2}{T}$, subject to the constraint B_i for each query. The objective is to allocate inference effort in a way that prioritizes correctness under limited computational resources.*

3 BUDGET ALLOCATION MODEL (BAM)

To address reasoning miscalibration in Problem 1, we need a principled method for allocating computation across sub-questions with varying uncertainty. As established by Falck et al. (2024b), effective reasoning requires focusing effort where epistemic uncertainty is high, and limiting it where aleatoric noise dominates. Existing methods lack a formal mechanism for this adaptive allocation. They often treat all reasoning steps uniformly, leading to inefficient budget use and exacerbating reasoning miscalibration. To bridge this gap, we introduce the **Budget Allocation Model (BAM)**, a theoretical framework that models token allocation as uncertainty reduction under a fixed budget. BAM provides a principled foundation for our adaptive reasoning framework presented in Section 4.

To distribute a finite token budget B_i across the sub-questions of x_i, we adopt a Bayesian decision-theoretic formulation that aims to maximize reasoning utility by minimizing total uncertainty. While standard LLM inference is deterministic, recent theoretical work suggests that In-Context Learning can be viewed as implicit Bayesian inference (Falck et al., 2024a). We leverage this normative view to characterize reasoning behavior, even without performing explicit posterior sampling at test time. We assume an inverse power law governs epistemic uncertainty reduction for sub-question s_{ij} with token allocation b_{ij}:

$$\mathcal{U}_{\text{epistemic}}(s_{ij} \mid b_{ij}) = \frac{c_{ij}}{b_{ij}^{\beta_{ij}}}, \tag{1}$$

where $c_{ij} > 0$ reflects the initial epistemic uncertainty and $\beta_{ij} \geq 1$ captures the complexity of reducing that uncertainty (where higher β_{ij} corresponds to being easier to reduce the uncertainty). This formulation is motivated by established Neural Scaling Laws (Kaplan et al., 2020; Hoffmann et al., 2022; Zeng et al., 2025), which demonstrate that model loss—a proxy for uncertainty—scales as a power law with compute. We effectively model test-time reasoning as a 'miniature' scaling law (Snell et al., 2024; Wu et al., 2025), where allocating more tokens reduces error at a diminishing rate.

We model total uncertainty as the sum of the epistemic and aleatoric components:

$$\mathcal{U}(s_{ij} \mid b_{ij}) = \frac{c_{ij}}{b_{ij}^{\beta_{ij}}} + \mathcal{U}_{\text{aleatoric}}(s_{ij}). \tag{2}$$

Here, we treat $\mathcal{U}_{\text{aleatoric}}$ as a constant with respect to b_{ij}, since it reflects irreducible uncertainty that cannot be mitigated through additional inference effort. A proof of the decomposition of total uncertainty in LLM is provided in Appendix B.

We define the utility of successfully resolving a sub-question s_{ij} as inversely proportional to its uncertainty:

$$r(s_{ij} \mid b_{ij}) = \alpha \cdot (1 - \mathcal{U}(s_{ij} \mid b_{ij})), \tag{3}$$

where α is a model/task-based scaling factor. The total utility for query x_i is then:

$$\mathcal{R}_{\text{total}} = \sum_{j=1}^{m} r(s_{ij} \mid b_{ij}). \tag{4}$$

The optimal budget allocation solves the following constrained optimization problem:

$$\max_{b_{i1},\ldots,b_{im}} \sum_{j=1}^{m} \alpha \cdot \left(1 - \frac{c_{ij}}{b_{ij}^{\beta_{ij}}} - \mathcal{U}_{\text{aleatoric}}(s_{ij})\right) \quad \text{s.t.} \quad \sum_{j=1}^{m} b_{ij} \leq B_i. \tag{5}$$

By introducing a Lagrange multiplier λ to handle the budget constraint and solving the resulting Lagrangian, we arrive at the **optimality principle**:

$$b_{ij} = B_i \cdot \frac{(c_{ij}\beta_{ij})^{\frac{1}{\beta_{ij}+1}}}{\sum_k (c_{ik}\beta_{ik})^{\frac{1}{\beta_{ik}+1}}}. \tag{6}$$

This allocation rule reveals a unimodal relationship between b_{ij} and β_{ij}, i.e., token budget increases with complexity up to the peak, then decreases as further effort yields diminishing returns. This relationship is key to mitigating reasoning miscalibration: *moderately difficult sub-questions receive more tokens to avoid underthinking, while overly difficult ones receive fewer to prevent overthinking.* BAM thus provides a principled, self-regulating mechanism for aligning inference effort with reasoning value. Detailed proofs are provided in Appendix C and D.

4 REASONING CALIBRATION FRAMEWORK: PLAN-AND-BUDGET

Building directly on BAM's principle in Eq. 6, the optimal allocation prescribes distributing a query-level budget B across sub-questions by maximizing expected uncertainty reduction. In practice, however, the marginal gain curves are unknown. Thus, we proposed PLAN-AND-BUDGET which operationalizes the same objective via two proxies:

- We approximate $\Delta U_i(\cdot)$ using lightweight online signals (e.g., prefix-entropy drop, self-consistency disagreement, verifier loss), yielding marginal gain proxies $\hat{g}_i$.

- Relaxing Equation (6) with a Lagrange multiplier λ and assuming diminishing returns leads to multiplicative updates $t_i^{(k+1)} = \gamma_i \, t_i^{(k)}, \gamma_i \approx \exp\big(-\lambda/\hat{g}_i\big)$, which adaptively shift budget toward sub-questions with higher estimated gain.

This schedule is budget-feasible by construction and recovers the BAM allocation when proxies are consistent.

4.1 Plan Step: Question Decomposition as Guided Scaffold

Inspired by human problem-solving strategies, we use query decomposition as a reasoning scaffold to improve efficiency and focus. Our planning process has two phases:

Phase 1: Automatic Planning. A lightweight planning function $\mathcal{P}$ decomposes x_i into an ordered sequence of sub-questions π_i and their estimated complexity scores $\mathcal{D}_i$:

$$\mathcal{P}(x_i) \to (\pi_i, \mathcal{D}_i), \quad \pi_i = \langle s_{i1}, s_{i2}, \ldots, s_{im} \rangle, \quad \mathcal{D}_i = \langle d_{i1}, d_{i2}, \ldots, d_{im} \rangle.$$

Here, π_i denotes the decomposition plan, a sequence of m sub-questions, where each s_{ij} is a natural language prompt targeting a specific sub-problem of the query x_i. The vector $\mathcal{D}_i = \langle d_{i1}, d_{i2}, \ldots, d_{im} \rangle$ contains corresponding complexity scores, with each $d_{ij} \in \mathbb{R}_{>0}$ reflecting the estimated complexity of solving s_{ij} based on LLM confidence, problem structure, or other heuristics.

The decomposition plan π_i is not unique or guaranteed to be optimal, but acts as a *soft scaffold*, a plausible high-level reasoning path as a prompt to guide the main LLM. The planning function $\mathcal{P}$ can be implemented via applying a decomposition prompt in a lightweight LLM (see Appendix H). The resulting complexity scores d_{ij} reflect epistemic uncertainty and help estimate the computational effort required for each sub-question. These scores are then normalized into a weight vector $\mathbf{w}_i$:

$$w_{ij} = \frac{d_{ij}}{\sum_{k=1}^{m} d_{ik}}.$$

This normalized weight w_{ij} represents the proportion of the total "complexity" that is attributed to the j-th sub-question. This weight vector then plays a key role in the budget allocation mechanism, determining how the total token budget B_i is distributed across the individual sub-questions.

Phase 2: Guided Reasoning. After decomposing x_i into sub-questions $\langle s_{i1}, \ldots, s_{im} \rangle$ and allocating token budgets $b_{i1}, \ldots, b_{im}$, the main reasoning LLM is guided by these sub-questions (see the prompt template in Appendix H). Each sub-question s_{ij} is answered within its allocated budget b_{ij}, yielding responses $a_{ij} = f_{\text{LLM}}(s_{ij}, b_{ij})$, where f_{LLM} denotes the budget-constrained generation process. This constraint mitigates reasoning miscalibration by preventing excessive token use on individual steps. After all sub-questions are answered, a synthesis function $\mathcal{S}$ aggregates the responses, which answers the original query x_i: $y_i = S(a_{i1}, \ldots, a_{im})$.

4.2 Budget Step: Decay-Based Budget Allocation

While our Bayesian formulation offers an optimal allocation strategy based on sub-question-specific uncertainty parameters (c_{ij} and β_{ij}), estimating these values reliably in practice is often infeasible. To bridge this gap, we introduce a family of *decay-based scheduling functions* that approximate uncertainty-aware budget allocation in a lightweight and practical manner.

These functions allocate more tokens to early sub-questions, based on the observation that epistemic uncertainty is typically highest at the start of reasoning—when foundational understanding and strategy formation occur. Early token investment yields greater uncertainty reduction, consistent with the power law behavior of epistemic uncertainty in Equation 1. In contrast, later steps are generally narrower in scope or more deterministic, and over-allocating tokens at these stages risks wasting inference effort, as additional computation cannot reduce

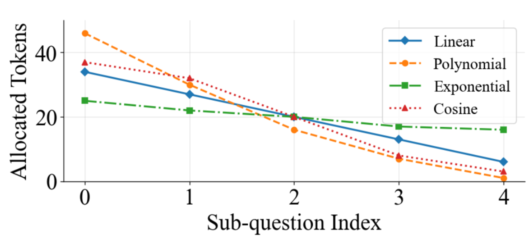

Figure 2: Visualization of decay functions. We take $B = 100$, $p = 2$, $\gamma = 0.9$, and 5 sub-questions with the same complexity as an example.

Table 2: Decay-based scheduling strategies for token budget allocation.

Strategy	Formula of d_{ij}	Description
Non-decay	1	Equal priority for all sub-questions; budget follows w_{ij}.
Linear decay	$m - j$	Decreases priority linearly with j; emphasizes early steps.
Polynomial decay	$(m - j)^p$	Stronger emphasis on early steps; steeper with higher $p > 1$.
Exponential decay	γ^j	Exponentially favors earlier sub-questions; controlled by $\gamma \in (0, 1)$.
Cosine annealing	$0.5\left(1 + \cos\left(\frac{\pi j}{m-1}\right)\right) + \epsilon$	Smooth decay with mid-sequence flexibility; ϵ adds stability.

the irreducible aleatoric uncertainty and yields diminishing returns in epistemic gain. Thus, decay functions offer a principled heuristic for prioritizing the budget where it is most valuable.

Given the normalized complexity weight vector $\mathbf{w}_i = \{w_{i1}, \ldots, w_{im}\}$ for a query x_i and the total token budget B_i, we allocate tokens using

$$b_{ij} = \left\lfloor \frac{w_{ij} \cdot d_{ij}}{\sum_{k=1}^{m} w_{ik} \cdot d_{ik}} \cdot B_i \right\rfloor, \tag{7}$$

where $d_{ij} = \texttt{schedule}(j, m)$ assigns positional priority to sub-question j in a length m sequence, reflecting the belief that earlier steps often carry higher epistemic uncertainty and merit more budget.

Experimental Scheduling Strategy. We explore several decay strategies (Table 2), each encoding a distinct prioritization schema over sub-question positions. Each strategy offers a flexible way to encode task-specific preferences. For instance, polynomial decay aggressively front-loads the budget, which may be beneficial in highly ambiguous tasks. Exponential decay offers a more balanced approach for problems with both early and mid-sequence challenges. Ultimately, these decay functions serve as practical surrogates to our Bayesian-optimal allocation by heuristically targeting the most epistemically impactful stages of reasoning.

Figure 2 shows that different decay strategies yield distinct allocation patterns even under uniform complexity, with polynomial decay and cosine annealing favoring early steps, linear offering gradual decline, and exponential decay providing balanced distribution—demonstrating that decay-based scheduling flexibly adapts token emphasis to match the structure of reasoning tasks.

5 EXPERIMENTS

We conduct extensive experiments across three types of reasoning-intensive downstream tasks to evaluate the effectiveness and efficiency of PLAN-AND-BUDGET. We assess performance in terms of raw accuracy and compute-aware reasoning efficiency using our proposed $\mathcal{E}^3$ metric. In particular, we aim to answer the following questions: **Q1**: *Does Plan-and-Budget improve reasoning efficiency without sacrificing accuracy*, compared to the baseline of using no planning (Vanilla) or applying a fixed budget (Global Budget)? **Q2**: *How does local, uncertainty-aware budgeting perform across models, datasets, and task types*, relative to uniform or global strategies? **Q3**: *Which scheduling strategies yield the best efficiency–accuracy tradeoff?*

5.1 EXPERIMENT SETUP

Datasets. We evaluate PLAN-AND-BUDGET on three representative benchmarks (see Table 6 in Appendix): (1) **MATH-500** (Lightman et al., 2024), a 500 math problem dataset requiring multi-step symbolic reasoning, evaluated by accuracy; (2) **NaturalInstructions** (Wang et al., 2022), a diverse instruction-following benchmark, evaluated using ROUGE score; and (3) **TravelPlanner** (Xie et al., 2024), a challenging agentic planning task evaluated by a hard constraint pass rate in a tool-free setting. This benchmark reflects the challenge of long-horizon, constraint-satisfying reasoning, with GPT-4-Turbo achieving 22.2% at best.

Models. We test our methods on four state-of-the-art, publicly available reasoning-tuned LLMs: DeepSeek-R1-Distill-Qwen-32B (**DS-Qwen-32B**) (Guo et al., 2025), **QwQ-32B** (Team, 2025), DeepSeek-R1-Distill-LLaMA-70B (**DS-LLaMA-70B**) (Guo et al., 2025), and OpenAI **o4-mini** (OpenAI, 2025). These models balance performance and accessibility and are specifically optimized for complex reasoning. For planning and budgeting, we use a lightweight non-reasoning LLM,

Table 3: Experiment results across different reasoning models on MATH-500. Acc denotes accuracy.

	Models →	DeepSeek-R1-Distill-Qwen-32B			QwQ-32B			DeepSeek-R1-Distill-Llama-70B			o4-mini		
	Methods↓	Acc (%) ↑	Avg. Tok.↓	$\mathcal{E}^3$ ↑	Acc (%) ↑	Avg. Tok.↓	$\mathcal{E}^3$ ↑	Acc (%) ↑	Avg. Tok.↓	$\mathcal{E}^3$ ↑	Acc (%) ↑	Avg. Tok.↓	$\mathcal{E}^3$ ↑
Direct	Vanilla	89.76$_{\pm 0.26}$	2105.12$_{\pm 31.94}$	3.83	84.88$_{\pm 1.18}$	3523.72$_{\pm 97.42}$	2.04	90.44$_{\pm 0.61}$	2286.63$_{\pm 26.42}$	3.58	**93.16$_{\pm 0.89}$**	711.20$_{\pm 8.31}$	12.20
	Global Budget	89.60$_{\pm 0.88}$	1526.15$_{\pm 10.09}$	5.26	**90.56$_{\pm 0.33}$**	2565.18$_{\pm 37.10}$	3.20	90.80$_{\pm 0.62}$	1810.83$_{\pm 51.64}$	4.55	91.84$_{\pm 0.48}$	636.41$_{\pm 8.14}$	13.25
Planned	Vanilla	**91.04$_{\pm 0.62}$**	1883.73$_{\pm 63.82}$	4.40	85.30$_{\pm 1.56}$	3309.69$_{\pm 18.06}$	2.20	92.12$_{\pm 1.16}$	2022.38$_{\pm 28.74}$	4.20	91.88$_{\pm 1.36}$	539.36$_{\pm 18.94}$	15.65
	Global Budget	91.24$_{\pm 1.34}$	1552.62$_{\pm 29.93}$	5.36	88.20$_{\pm 1.17}$	2671.60$_{\pm 15.02}$	2.91	92.56$_{\pm 0.71}$	1661.24$_{\pm 34.43}$	5.16	91.84$_{\pm 0.75}$	586.18$_{\pm 6.50}$	14.39
PLAN-AND-BUDGET	+ Uniform	90.16$_{\pm 0.74}$	1440.70$_{\pm 47.55}$	5.64	88.68$_{\pm 0.58}$	2397.16$_{\pm 23.01}$	3.28	92.28$_{\pm 0.41}$	1575.04$_{\pm 29.68}$	5.41	91.36$_{\pm 0.85}$	525.53$_{\pm 18.88}$	15.88
	+ Weighted	90.48$_{\pm 0.46}$	1485.99$_{\pm 45.63}$	5.51	87.45$_{\pm 0.66}$	2479.46$_{\pm 39.21}$	3.08	92.64$_{\pm 0.68}$	1557.64$_{\pm 47.71}$	5.51	91.64$_{\pm 1.21}$	538.22$_{\pm 5.30}$	15.60
	+ Linear	90.04$_{\pm 0.46}$	**1336.27$_{\pm 31.18}$**	**6.07**	88.13$_{\pm 0.90}$	2346.35$_{\pm 25.33}$	3.31	92.32$_{\pm 0.88}$	1529.98$_{\pm 45.35}$	5.57	90.56$_{\pm 0.73}$	534.45$_{\pm 7.64}$	15.34
	+ Exponential	90.80$_{\pm 0.68}$	1389.75$_{\pm 61.06}$	5.93	87.90$_{\pm 1.27}$	2320.04$_{\pm 72.33}$	3.33	93.04$_{\pm 0.22}$	**1469.29$_{\pm 73.77}$**	**5.89**	90.88$_{\pm 0.36}$	525.51$_{\pm 11.70}$	15.72
	+ Polynomial	90.04$_{\pm 0.26}$	1371.59$_{\pm 21.75}$	5.91	88.27$_{\pm 0.99}$	2346.94$_{\pm 17.73}$	3.32	91.92$_{\pm 1.15}$	1514.43$_{\pm 47.94}$	5.58	90.36$_{\pm 0.83}$	525.00$_{\pm 9.15}$	15.55
	+ Cosine	89.88$_{\pm 1.72}$	1365.51$_{\pm 44.92}$	5.92	88.60$_{\pm 0.28}$	**2306.83$_{\pm 24.11}$**	**3.40**	**92.88$_{\pm 0.46}$**	1487.83$_{\pm 61.78}$	5.80	91.32$_{\pm 0.94}$	**522.89$_{\pm 10.01}$**	**15.95**

Table 4: Experiment results across different reasoning models on NaturalInstructions.

	Models →	DeepSeek-R1-Distill-Qwen-32B			QwQ-32B			DeepSeek-R1-Distill-Llama-70B			o4-mini		
	Methods↓	ROUGE (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	ROUGE (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	ROUGE (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	ROUGE (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑
Direct	Vanilla	**43.47$_{\pm 0.52}$**	968.17$_{\pm 44.78}$	1.95	43.16$_{\pm 1.12}$	1818.34$_{\pm 24.99}$	1.02	43.13$_{\pm 0.76}$	894.46$_{\pm 50.69}$	2.08	**47.24$_{\pm 0.31}$**	460.99$_{\pm 11.31}$	4.84
	Global Budget	42.81$_{\pm 0.39}$	787.25$_{\pm 58.17}$	2.33	**44.77$_{\pm 0.73}$**	1360.49$_{\pm 101.64}$	1.47	**43.80$_{\pm 1.28}$**	772.98$_{\pm 47.44}$	2.48	45.39$_{\pm 1.27}$	422.20$_{\pm 56.78}$	4.88
Planned	Vanilla	42.48$_{\pm 0.67}$	860.85$_{\pm 49.58}$	2.10	44.24$_{\pm 0.67}$	1426.74$_{\pm 52.92}$	1.37	43.40$_{\pm 0.18}$	821.27$_{\pm 21.85}$	2.29	43.78$_{\pm 1.47}$	**344.99$_{\pm 14.44}$**	5.56
	Global Budget	42.50$_{\pm 0.36}$	717.98$_{\pm 36.28}$	2.52	45.13$_{\pm 0.56}$	1265.78$_{\pm 33.23}$	1.61	42.48$_{\pm 0.33}$	691.79$_{\pm 12.18}$	2.61	43.78$_{\pm 0.96}$	358.84$_{\pm 14.44}$	5.34
PLAN-AND-BUDGET	+ Uniform	41.03$_{\pm 0.55}$	644.87$_{\pm 46.34}$	2.61	44.47$_{\pm 0.35}$	996.91$_{\pm 31.31}$	1.98	43.06$_{\pm 0.33}$	665.94$_{\pm 47.22}$	2.78	44.08$_{\pm 0.81}$	348.74$_{\pm 8.13}$	**5.57**
	+ Weighted	41.29$_{\pm 0.50}$	663.9427.29	2.57	44.40$_{\pm 0.61}$	1025.02$_{\pm 24.91}$	1.92	43.05$_{\pm 0.39}$	626.37$_{\pm 19.46}$	**2.96**	43.72$_{\pm 1.00}$	371.85$_{\pm 9.53}$	5.14
	+ Linear	41.56$_{\pm 0.50}$	633.79$_{\pm 34.17}$	2.73	44.22$_{\pm 0.66}$	1003.24$_{\pm 26.23}$	1.95	42.05$_{\pm 0.99}$	**613.05$_{\pm 33.68}$**	2.88	44.21$_{\pm 0.44}$	363.65$_{\pm 13.70}$	5.37
	+ Exponential	41.44$_{\pm 0.50}$	650.19$_{\pm 31.35}$	2.64	43.99$_{\pm 0.22}$	1026.89$_{\pm 8.51}$	1.88	42.73$_{\pm 0.24}$	622.72$_{\pm 33.58}$	2.93	43.68$_{\pm 1.06}$	364.86$_{\pm 10.81}$	5.23
	+ Polynomial	41.44$_{\pm 0.78}$	**600.04$_{\pm 40.52}$**	**2.86**	44.66$_{\pm 0.68}$	**995.95$_{\pm 14.43}$**	**2.00**	43.19$_{\pm 0.44}$	641.62$_{\pm 32.22}$	2.91	44.63$_{\pm 1.04}$	363.16$_{\pm 11.71}$	5.48
	+ Cosine	41.43$_{\pm 1.01}$	628.20$_{\pm 36.63}$	2.73	44.53$_{\pm 0.54}$	1000.64$_{\pm 17.85}$	1.98	42.83$_{\pm 0.63}$	657.93$_{\pm 59.06}$	2.79	44.36$_{\pm 1.06}$	363.05$_{\pm 16.72}$	5.42

LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024). To ensure that it does not inadvertently contribute to final answer quality, we evaluate its standalone performance on the three benchmarks and find that it underperforms specialized models: $48.76_{\pm 0.74}$ on MATH-500, $21.72_{\pm 0.98}$ on NaturalInstructions, and $2.91_{\pm 0.28}$ on TravelPlanner. This confirms its role as a neutral planner.

Evaluation Metrics. We report the following metrics: (1) **Score (%)**, the original evaluation metric used in each dataset; (2) **Avg. Tokens**, the average number of all billed completion tokens per query, including planning, reasoning and output tokens (for open-source models, tokens before </think> and final outputs; for o4-mini, the sum of reasoning and output tokens as reported in OpenAI documentation (OpenAI, 2025)); and (3) $\mathcal{E}^3$ **Metric**, which captures the balance between correctness and computational cost.

Baselines. We compare our proposed framework against several baselines: (1) **Vanilla.** The query is given to the LLM without planning or token constraint; (2) **Global Budget.** Same as Vanilla but with a token limit prompt (e.g., "use less than B_i tokens"); (3) **Planned Vanilla / Global Budget.** Same as above, but with the original query and its decomposed sub-questions provided; and (4) **PLAN-AND-BUDGET.** Our methods—the query, sub-questions, and local budget prompts are given. We explore several scheduling strategies for local allocation: (a) **Uniform**, equal tokens per sub-question; (b) **Weighted**, proportional to the estimated difficulty; and (c) **Linear, Polynomial, Exponential, Cosine**, weighted by difficulty with additional decay (we use $p = 2$ and $\gamma = 0.9$). A hard cutoff of 8192 tokens is applied to prevent runaway generations. We report the average and standard deviation over 5 runs for all models and baselines.

5.2 COMPARATIVE RESULTS

We now address the questions introduced earlier by analyzing results across datasets and models.

Tables 3–5 summarize our main findings. Across all datasets and model scales, PLAN-AND-BUDGET consistently outperforms both the Vanilla and Global Budget baselines, achieving up to **193.8% improvement in** $\mathcal{E}^3$, while maintaining comparable or even higher accuracy. To further illustrate this, Figure 3 shows answer pass rates of QwQ-32B on TravelPlan-

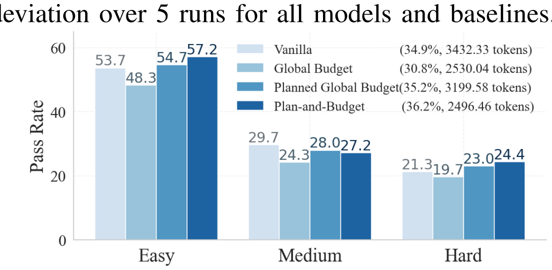

Figure 3: Answer pass rate grouped by the question difficulty level in TravelPlanner. The global budget limit hurts the pass rate on all levels, while our method not only achieves a higher pass rate but also enjoys lower token usage. Overall pass rate and average token usage are shown in the legend.

Table 5: Experiment results on TravelPlanner. Rate denotes the hard constraint pass rate.

	Models →	DeepSeek-R1-Distill-Qwen-32B			QwQ-32B			DeepSeek-R1-Distill-Llama-70B			o4-mini		
	Methods↓	Rate (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	Rate (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	Rate (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	Rate (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑
Direct	Vanilla	$14.33_{\pm 2.17}$	$1430.14_{\pm 43.73}$	0.14	$34.89_{\pm 3.20}$	$3432.33_{\pm 78.66}$	0.35	$26.22_{\pm 1.82}$	$1361.37_{\pm 47.93}$	0.50	$11.58_{\pm 2.15}$	$1559.65_{\pm 8.84}$	0.086
	Global Budget	$13.78_{\pm 1.20}$	$1158.81_{\pm 20.23}$	0.16	$30.78_{\pm 2.06}$	$2530.04_{\pm 40.87}$	0.37	$24.33_{\pm 2.30}$	$1215.29_{\pm 35.05}$	0.49	$8.33_{\pm 1.71}$	$\mathbf{1248.53_{\pm 26.97}}$	0.056
Planned	Vanilla	$20.22_{\pm 1.01}$	$1343.67_{\pm 62.44}$	0.30	$\mathbf{37.22_{\pm 1.80}}$	$3669.88_{\pm 42.09}$	0.38	$30.67_{\pm 2.17}$	$1464.50_{\pm 65.40}$	0.64	$\mathbf{12.20_{\pm 2.47}}$	$1640.46_{\pm 95.33}$	0.091
	Global Budget	$22.56_{\pm 2.41}$	$1241.19_{\pm 54.66}$	0.41	$35.22_{\pm 4.85}$	$3199.58_{\pm 63.14}$	0.39	$30.67_{\pm 1.73}$	$1220.41_{\pm 32.22}$	0.77	$7.19_{\pm 2.43}$	$1392.11_{\pm 31.05}$	0.037
Plan-and-Budget	+ Uniform	$20.67_{\pm 1.20}$	$1227.99_{\pm 68.55}$	0.35	$36.00_{\pm 2.79}$	$2854.24_{\pm 44.87}$	0.45	$31.56_{\pm 2.20}$	$1232.98_{\pm 34.16}$	0.81	$11.00_{\pm 1.62}$	$1345.32_{\pm 58.88}$	0.090
	+ Weighted	$\mathbf{23.33_{\pm 1.11}}$	$1222.09_{\pm 40.69}$	0.45	$33.89_{\pm 2.22}$	$2842.74_{\pm 77.68}$	0.40	$29.67_{\pm 3.01}$	$1197.32_{\pm 10.78}$	0.74	$10.91_{\pm 3.01}$	$1353.67_{\pm 37.64}$	0.088
	+ Linear	$19.56_{\pm 2.47}$	$\mathbf{1136.18_{\pm 54.92}}$	0.34	$34.55_{\pm 2.65}$	$2671.70_{\pm 67.97}$	0.45	$31.67_{\pm 2.32}$	$1162.24_{\pm 43.31}$	0.86	$11.66_{\pm 1.96}$	$1306.54_{\pm 55.05}$	0.103
	+ Exponential	$21.44_{\pm 2.98}$	$1156.64_{\pm 30.52}$	0.40	$35.44_{\pm 2.06}$	$2724.23_{\pm 41.87}$	0.46	$32.00_{\pm 2.14}$	$1187.85_{\pm 36.57}$	0.86	$9.91_{\pm 1.96}$	$1307.87_{\pm 40.83}$	0.075
	+ Polynomial	$23.11_{\pm 2.14}$	$1148.53_{\pm 37.33}$	**0.47**	$35.00_{\pm 3.35}$	$2511.35_{\pm 84.18}$	0.49	$\mathbf{32.67_{\pm 2.06}}$	$\mathbf{1148.14_{\pm 59.00}}$	**0.93**	$11.49_{\pm 1.31}$	$1266.11_{\pm 28.48}$	**0.104**
	+ Cosine	$20.22_{\pm 2.34}$	$1140.79_{\pm 6.68}$	0.36	$36.18_{\pm 3.00}$	$\mathbf{2496.46_{\pm 40.10}}$	**0.52**	$31.67_{\pm 2.22}$	$1173.96_{\pm 44.22}$	0.85	$9.79_{\pm 1.57}$	$1252.06_{\pm 80.85}$	0.077

ner, grouped by difficulty level. While global budget constraints reduce token usage, they also degrade pass rates across all levels. In contrast, PLAN-AND-BUDGET achieves both higher pass rates and lower token usage, especially on harder queries, highlighting its ability to scale reasoning adaptively with problem complexity.

On MATH-500, our method improves $\mathcal{E}^3$ consistently by over 20%—for instance, from 4.55 → 5.89 (+29.4%) on DS-LLaMA-70B and from 13.25 → 15.95 (+20.3%) on o4-mini. Importantly, this is achieved without compromising the accuracy. While the Global Budget baseline reduces token usage, its gains are limited due to a lack of uncertainty-awareness. Notably, we find that **planning alone** (Planned Global Budget) already mostly boosts efficiency by 2–13%, validating our first key principle: *reasoning should be structured*. This scaffolding greatly reduces speculative exploration. Moreover, $\mathcal{E}^3$ enables easy comparison across models – e.g., o4-mini consistently achieves the highest $\mathcal{E}^3$, despite having similar accuracy to other models, because it uses the fewest tokens. This underscores the importance of $\mathcal{E}^3$ as a practical efficiency metric.

A1: We achieve substantial efficiency gains with comparable accuracy. On NaturalInstructions, PLAN-AND-BUDGET improves $\mathcal{E}^3$ by 19.3–36.0%. For example, on QwQ-32B, it improves from 1.47 → 2.00 (+36%), and on o4-mini, from 4.88 → 5.57 (+14%). Although these tasks are more instruction-oriented, PLAN-AND-BUDGET remains beneficial. On TravelPlanner, the most open-ended and challenging benchmark, we observe the most dramatic gains: $\mathcal{E}^3$ improves from 0.16 → 0.47 (+193.8%) on DS-Qwen-32B, from 0.49 → 0.93 (+89.8%) on DS-LLaMA-70B, and 0.056 → 0.104 (+85.7%) on o4-mini. These results highlight that **the more complex the task, the greater the benefit of structure and adaptivity.**

A2: Local budgeting consistently improves efficiency. While structured planning alone improves efficiency, adding local budgeting yields significant additional gains. We can observe that on MATH-500, DS-LLaMA-70B improves $\mathcal{E}^3$ from 5.16 → 5.89 (+14.1%); on NaturalInstructions, QwQ-32B improves from 1.61 → 2.00 (+24.2%); and on TravelPlanner, from 0.39 → 0.52 (+33.3%). These results confirm the importance of adapting the budget to the sub-question, rather than applying a global allocation.

A3: Front-loaded scheduling performs best on complex tasks. Among local budget schedulers, polynomial decay and cosine annealing consistently deliver the highest $\mathcal{E}^3$ on mathematical and long-form planning tasks. These strategies front-load computation, allocating more budget to early, uncertain steps where reasoning direction is established. This pattern is particularly effective on MATH-500 and TravelPlanner, where clarity at the beginning of the reasoning is crucial. In contrast, on NaturalInstructions, weighted or uniform schedules usually perform well, suggesting that smooth, evenly paced reasoning suffices for tasks with clearer structure and less ambiguity.

A4: Bridging the gap between small and large Models. Our method is model-agnostic: it requires no retraining or fine-tuning, relying only on prompting and lightweight planning. We observe consistent improvements across model sizes, from small models like QwQ-32B to large models like DeepSeek-R1-70B and o4-mini. An especially notable result comes from TravelPlanner, where a compact model (DS-Qwen-32B) originally achieved only $\mathcal{E}^3$ = 0.16, but reached $\mathcal{E}^3$ = 0.47 after applying PLAN-AND-BUDGET, on par with a larger model with no planning (DS-LLaMA-70B, $\mathcal{E}^3$ = 0.50). This demonstrates that planning and budgeting can serve as powerful inference-time equalizers, closing the gap between small and large models through better compute utilization.

6 Conclusion

We propose Plan-and-Budget, a lightweight test-time framework that improves LLM reasoning efficiency by combining structured planning with uncertainty-aware token budgeting. Built on our BAM, Plan-and-Budget models reasoning as a sequence of sub-questions and adaptively allocates computation based on estimated difficulty. Experiments on three different reasoning tasks show that Plan-and-Budget achieves significant improvements in compute efficiency over strong baselines, without compromising accuracy. Although effective, our method currently requires an additional LLM call to generate the decomposition plan. In future work, we aim to fine-tune and develop a dedicated planner LLM to internalize the plan-and-budget strategy, enabling end-to-end, efficient reasoning within a single model.

Reproducibility Statement

We have taken multiple steps to ensure reproducibility. Theoretical assumptions and derivations of the Budget Allocation Model (BAM) are detailed in Section 3, with complete proofs provided in Appendix B–D. Experimental setups, evaluation metrics, and dataset statistics are described in Section 5 and Appendix E, including licenses for all datasets and models. We provide an anonymized code repository (linked in the abstract) containing implementations of all baselines, our Plan-and-Budget framework, and scripts to reproduce every table and figure. Additional details, such as prompt templates and ablation studies of scheduling strategies, are included in Appendix G and F. Together, these resources allow independent verification and extension of both our theoretical and empirical findings.

Ethics Statement

This work relies exclusively on publicly available datasets (MATH-500, NaturalInstructions, and TravelPlanner) and open-source or API-accessible large language models (e.g., DeepSeek, QwQ, o4-mini). No human subjects, private, or sensitive data were used. The proposed method improves inference efficiency by reducing unnecessary computation, which can lower environmental and financial costs of deploying LLMs. However, as with any efficiency-focused technique, there is a risk of misuse in high-stakes applications (e.g., medical or legal decision-making) if efficiency is prioritized over accuracy. We mitigate this risk by explicitly emphasizing correctness in our E3 metric and by recommending careful, task-specific evaluation before real-world deployment. Our framework does not modify underlying model internals and thus inherits any limitations or biases present in the base models.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. In *The 18th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 225, 2024.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Zhipeng Chen, Kun Zhou, Beichen Zhang, Zheng Gong, Xin Zhao, and Ji-Rong Wen. ChatCoT: Tool-augmented chain-of-thought reasoning on chat-based large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14777–14790, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.985. URL https://aclanthology.org/2023.findings-emnlp.985/.

Fabian Falck, Ziyu Wang, and Chris Holmes. Is in-context learning in large language models bayesian? a martingale perspective, 2024a. URL https://arxiv.org/abs/2406.00793.

Fabian Falck, Ziyu Wang, and Chris Holmes. Is in-context learning in large language models bayesian? a martingale perspective, 2024b. URL https://arxiv.org/abs/2406.00793.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, pp. 1050–1059. PMLR, 2016.

Behzad Golshan and Khan Academy. Khanmigo: An ai-powered assistant for education, 2023. URL https://www.khanmigo.ai/. Accessed: 2025-05-14.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Michelle Halbheer, Dominik J. Mühlematter, Alexander Becker, Dominik Narnhofer, Helge Aasen, Konrad Schindler, and Mehmet Ozgur Turkoglu. Lora-ensemble: Efficient uncertainty modelling for self-attention networks, 2024. URL https://arxiv.org/abs/2405.14438.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Zhen Huang, Zengzhi Wang, Shijie Xia, Xuefeng Li, Haoyang Zou, Ruijie Xu, Run-Ze Fan, Lyumanshan Ye, Ethan Chern, Yixin Ye, et al. Olympicarena: Benchmarking multi-discipline cognitive reasoning for superintelligent ai. *Advances in Neural Information Processing Systems*, 37:19209–19253, 2024.

Ziyi Huang, Henry Lam, and Haofeng Zhang. Quantifying epistemic uncertainty in deep learning, 2023. URL https://arxiv.org/abs/2110.12122.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110(3):457–506, 2021.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3):457–506, March 2021. ISSN 1573-0565. doi: 10.1007/s10994-021-05946-3. URL http://dx.doi.org/10.1007/s10994-021-05946-3.

Shima Imani, Liang Du, and Harsh Shrivastava. Mathprompter: Mathematical reasoning using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pp. 37–42, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Daniel Martin Katz, Michael J. Bommarito, Shang Gao, and Pablo Arredondo. GPT-4 passes the bar exam. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 382 (2270):20230254, Apr 2024. doi: 10.1098/rsta.2023.0254. Epub 2024 Feb 26.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Ayeong Lee, Ethan Che, and Tianyi Peng. How well do llms compress their own chain-of-thought? a token complexity approach, 2025. *URL https://arxiv. org/abs/2503.01141*.

Ayeong Lee, Ethan Che, and Tianyi Peng. How well do llms compress their own chain-of-thought? a token complexity approach. *arXiv preprint arXiv:2503.01141*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

Lingrui Mei, Shenghua Liu, Yiwei Wang, Baolong Bi, Yuyao Ge, Jun Wan, Yurong Wu, and Xueqi Cheng. a1: Steep test-time scaling law via environment augmented generation. *arXiv preprint arXiv:2504.14597*, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Workshop on Reasoning and Planning for Large Language Models*.

OpenAI. Introducing o3 and o4-mini, April 2025. URL https://openai.com/index/introducing-o3-and-o4-mini/. Accessed: 2025-05-12.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science, 2022. URL https://arxiv.org/abs/2211.09085.

Qwen Team. Qwq-32b: Embracing the power of reinforcement learning, March 2025. URL https://qwenlm.github.io/blog/qwq-32b/.

Francesco Tonolini, Nikolaos Aletras, Jordan Massiah, and Gabriella Kazai. Bayesian prompt ensembles: Model uncertainty estimation for black-box large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12229–12272, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.728. URL https://aclanthology.org/2024.findings-acl.728/.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting, 2023. URL https://arxiv.org/abs/2305.04388.

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

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, 2022.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*, 2025a.

Yutong Wang, Pengliang Ji, Chaoqun Yang, Kaixin Li, Ming Hu, Jiaoyang Li, and Guillaume Sartoretti. Mcts-judge: Test-time scaling in llm-as-a-judge for code correctness evaluation. *arXiv preprint arXiv:2502.12468*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems (NeurIPS)*, 35:24824–24837, 2022.

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

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: a benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 54590–54613, 2024.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models, 2023. URL https://arxiv.org/abs/2306.06031.

Xinyue Zeng, Haohui Wang, Junhong Lin, Jun Wu, Tyler Cody, and Dawei Zhou. Lensllm: Unveiling fine-tuning dynamics for llm selection. *arXiv preprint arXiv:2505.03793*, 2025.

A LLM USAGE

Large language models (LLMs) were employed in a limited and transparent manner during the preparation of this manuscript. Specifically, LLMs were used to assist with linguistic refinement, style adjustments, and minor text editing to improve clarity and readability. They were not involved in formulating the research questions, designing the theoretical framework, conducting experiments, or interpreting results. All scientific contributions—including conceptual development, methodology, analyses, and conclusions—are the sole responsibility of the authors.

B PROOF OF UNCERTAINTY DECOMPOSITION FOR LLMS

Let θ denote the parameters of an LLM (e.g., transformer weights), and let x^* be the test-time input with corresponding output y^*. Under a Bayesian treatment, the predictive distribution is given by:

$$p(y^*|x^*, D) = \int p(y^*|x^*, \theta)p(\theta|D)\, d\theta,$$

and is often approximated via Monte Carlo sampling:

$$p(y^*|x^*, D) \approx \frac{1}{M}\sum_{m=1}^{M} p(y^*|x^*, \theta_m), \quad \theta_m \sim p(\theta|D).$$

We define the **total predictive uncertainty** as the Shannon entropy of this marginal predictive distribution:

$$\mathcal{U}(x^*) = \mathcal{H}\left[p(y^*|x^*, D)\right] = \mathcal{H}\left[\int p(y^*|x^*, \theta)p(\theta|D)d\theta\right].$$

To derive the decomposition, we apply the *law of total entropy*, which relates the entropy of the marginal to the expected entropy of the conditionals and the mutual information:

$$\mathcal{H}[y^*|x^*, D] = \mathbb{E}_{p(\theta|D)}[\mathcal{H}[y^*|x^*, \theta]] + \mathcal{I}(y^*; \theta|x^*, D).$$

Step-by-step Derivation:

Let: - $p(y^*|x^*, \theta)$ — the conditional predictive distribution. - $p(y^*|x^*, D)$ — the marginal (Bayesian averaged) predictive distribution.

The total predictive uncertainty is:

$$\mathcal{U}(x^*) = \mathcal{H}\left[p(y^*|x^*, D)\right] = -\sum_{y^*} p(y^*|x^*, D)\log p(y^*|x^*, D).$$

Define **aleatoric uncertainty** as the expected conditional entropy:

$$\mathcal{U}_{\text{aleatoric}}(x^*) = \mathbb{E}_{p(\theta|D)}[\mathcal{H}[p(y^*|x^*, \theta)]] = \int p(\theta|D)\left(-\sum_{y^*} p(y^*|x^*, \theta)\log p(y^*|x^*, \theta)\right) d\theta.$$

Then define **epistemic uncertainty** as the mutual information:

$$\mathcal{U}_{\text{epistemic}}(x^*) = \mathcal{I}(y^*; \theta|x^*, D) = \mathcal{H}[y^*|x^*, D] - \mathbb{E}_{p(\theta|D)}[\mathcal{H}[y^*|x^*, \theta]].$$

Combining the above, we obtain:

$$\mathcal{U}(x^*) = \mathcal{U}_{\text{aleatoric}}(x^*) + \mathcal{U}_{\text{epistemic}}(x^*).$$

Interpretation:

- $\mathcal{U}_{\text{aleatoric}}(x^*)$: Irreducible uncertainty present in each individual model prediction, even if θ were known.

- $\mathcal{U}_{\text{epistemic}}(x^*)$: Captures model uncertainty due to limited data, reflected in disagreement across posterior samples.

In practice, following Hüllermeier & Waegeman (2021), we approximate this decomposition using Monte Carlo estimation. Drawing M samples $\theta_1, \ldots, \theta_M$ from $p(\theta|D)$, we compute:

$$\mathcal{U}(x^*) \approx \mathcal{H}\left[\frac{1}{M}\sum_{m=1}^{M} p(y^*|x^*, \theta_m)\right],$$

$$\mathcal{U}_{\text{aleatoric}}(x^*) \approx \frac{1}{M}\sum_{m=1}^{M} \mathcal{H}\left[p(y^*|x^*, \theta_m)\right],$$

$$\mathcal{U}_{\text{epistemic}}(x^*) \approx \mathcal{U}(x^*) - \mathcal{U}_{\text{aleatoric}}(x^*).$$

Thus, the uncertainty decomposition holds in both exact Bayesian inference and its Monte Carlo approximation, validating its use in practical LLM reasoning pipelines. In the context of our Plan-and-Budget framework, we utilize this decomposition as a theoretical lens to explain why structured budgeting works. We do not perform the computationally expensive Monte Carlo sampling described above during inference; rather, we rely on the deterministic approximation that the model's single generation path is dominated by the properties of its underlying uncertainty distribution.

C Proof of Lagrange Optimality

Proof. We aim to maximize the total utility:

$$\mathcal{R}_{\text{total}} = \sum_{j=1}^{m} r(s_{ij} \mid b_{ij}) = \sum_{j=1}^{m} \alpha \left(1 - \frac{c_{ij}}{b_{ij}^{\beta_{ij}}} - \mathcal{U}_{\text{aleatoric}}(s_{ij})\right). \tag{8}$$

Since α and $\mathcal{U}_{\text{aleatoric}}(s_{ij})$ are constants with respect to b_{ij}, maximizing the total utility is equivalent to minimizing the following:

$$\sum_{j=1}^{m} \frac{c_{ij}}{b_{ij}^{\beta_{ij}}} \quad \text{subject to} \quad \sum_{j=1}^{m} b_{ij} = B_i. \tag{9}$$

Step 1: Form the Lagrangian.
We define the Lagrangian:

$$\mathcal{L}(\{b_{ij}\}, \lambda) = \sum_{j=1}^{m} \frac{c_{ij}}{b_{ij}^{\beta_{ij}}} + \lambda\left(\sum_{j=1}^{m} b_{ij} - B_i\right). \tag{10}$$

Taking the partial derivative with respect to b_{ij} and setting it to zero:

$$\frac{\partial \mathcal{L}}{\partial b_{ij}} = -c_{ij}\beta_{ij} b_{ij}^{-(\beta_{ij}+1)} + \lambda = 0 \quad \Rightarrow \quad \lambda = c_{ij}\beta_{ij} b_{ij}^{-(\beta_{ij}+1)}. \tag{11}$$

Solving for b_{ij} gives:

$$b_{ij}^{\beta_{ij}+1} = \frac{c_{ij}\beta_{ij}}{\lambda} \quad \Rightarrow \quad b_{ij} = \left(\frac{c_{ij}\beta_{ij}}{\lambda}\right)^{\frac{1}{\beta_{ij}+1}}. \tag{12}$$

Step 2: Apply the budget constraint.
Substitute into the constraint $\sum_j b_{ij} = B_i$:

$$\sum_{j=1}^{m} \left(\frac{c_{ij}\beta_{ij}}{\lambda}\right)^{\frac{1}{\beta_{ij}+1}} = B_i. \tag{13}$$

Let

$$A_j := (c_{ij}\beta_{ij})^{\frac{1}{\beta_{ij}+1}}, \quad \text{so that} \quad b_{ij} = \lambda^{-1/(\beta_{ij}+1)} A_j.$$

Then the constraint becomes:

$$\sum_{j=1}^{m} \lambda^{-1/(\beta_{ij}+1)} A_j = B_i. \tag{14}$$

This expression has a closed-form solution for λ only when all $\beta_{ij} = \beta$ (i.e., homogeneous difficulty). In that case:

$$b_{ij} = \left(\frac{c_{ij}\beta}{\lambda}\right)^{\frac{1}{\beta+1}} \quad \Rightarrow \quad \sum_j \left(\frac{c_{ij}\beta}{\lambda}\right)^{\frac{1}{\beta+1}} = B_i \quad \Rightarrow \quad \lambda = \left(\frac{\sum_j (c_{ij}\beta)^{\frac{1}{\beta+1}}}{B}\right)^{\beta+1}. \tag{15}$$

Substituting back yields:

$$b_{ij}^* = B_i \cdot \frac{(c_{ij}\beta)^{\frac{1}{\beta+1}}}{\sum_k (c_{ik}\beta)^{\frac{1}{\beta+1}}}. \tag{16}$$

In the general case of heterogeneous β_{ij}, the normalized form can still be written as:

$$b_{ij}^* = B_i \cdot \frac{(c_{ij}\beta_{ij})^{\frac{1}{\beta_{ij}+1}}}{\sum_k (c_{ik}\beta_{ik})^{\frac{1}{\beta_{ik}+1}}}, \tag{17}$$

which satisfies the budget constraint $\sum_j b_{ij} = B_i$, thus completing the proof. □

D ANALYSIS OF THE RELATIONSHIP BETWEEN b_{ij} AND β_{ij}

We examine the behavior of the allocation function in Equation 6:

$$b_{ij} = B_i \cdot \frac{(c_{ij}\beta_{ij})^{\frac{1}{\beta_{ij}+1}}}{\sum_k (c_{ik}\beta_{ik})^{\frac{1}{\beta_{ik}+1}}}. \tag{18}$$

To analyze the relationship between b_{ij} and β_{ij}, we focus on the numerator:

$$f(\beta) := (\beta c)^{\frac{1}{\beta+1}} = \exp\left(\frac{\log(\beta c)}{\beta+1}\right). \tag{19}$$

Let us define:

$$g(\beta) := \frac{\log(\beta c)}{\beta+1}, \quad \text{so that} \quad f(\beta) = e^{g(\beta)}. \tag{20}$$

We now study the behavior of $f(\beta)$ through the derivative of $g(\beta)$:

$$g'(\beta) = \frac{1}{\beta+1} \cdot \frac{1}{\beta} - \frac{\log(\beta c)}{(\beta+1)^2} \tag{21}$$

$$= \frac{1}{\beta(\beta+1)} - \frac{\log(\beta c)}{(\beta+1)^2}. \tag{22}$$

The sign of $g'(\beta)$ depends on β, and it is not monotonic. The function $g(\beta)$ increases initially, reaches a maximum, and then decreases. Consequently, $g(\beta)$ is **unimodel**, and since $f(\beta) = e^{g(\beta)}$, $f(\beta)$ is also unimodal.

E BROADER IMPACTS

Our work proposes a lightweight test-time framework that improves the efficiency of LLM reasoning through structured planning and uncertainty-aware computation. This has potential positive societal impacts by reducing computational costs, improving energy efficiency, and making advanced LLM capabilities more accessible—particularly in resource-constrained settings. By narrowing the performance gap between small and large models, our method may also promote more equitable access to language technologies.

However, as with any LLM inference technique, risks remain if deployed without careful oversight. More efficient reasoning pipelines could accelerate LLM integration into high-stakes applications (e.g., legal or medical decision-making) where accuracy, fairness, and robustness are critical. Our method does not modify model internals and inherits any limitations or biases present in the base models. Mitigation strategies include model-level auditing, task-specific evaluation, and responsible deployment practices.

F ADDITIONAL EXPERIMENTAL DETAILS

F.1 DATASET DESCRIPTIONS AND EVALUATION METRICS

To evaluate the general applicability of our framework, we select three reasoning-heavy benchmarks spanning symbolic math, instruction following, and long-horizon planning.

Table 6: Dataset Statistics. LLaMA 3.1-8B sole performance is also provided.

	MATH-500	Natural Instructions	Travel Planner
Task	Math Reasoning	Instruction Following	Agentic Planning
QA Pairs	500	500	180
Metrics	Accuracy	ROUGE	Pass rate
LLaMA 3.1-8B Performance	$48.76_{\pm 0.74}$	$21.72_{\pm 0.98}$	$2.91_{\pm 0.28}$

MATH-500. A curated 500-problem subset from the full MATH dataset, designed to test symbolic, multi-step math reasoning. Each problem requires the model to interpret, manipulate, and solve high-school level mathematical expressions. Performance is measured using exact-match accuracy against gold answers.

NaturalInstructions. A broad instruction-following benchmark consisting of over 1600 tasks covering question answering, classification, transformation, and reasoning. We randomly sample 500 test queries from the public split for evaluation. Since answers are open-ended and linguistic, we use ROUGE score to measure semantic overlap with the reference answers.

TravelPlanner. A challenging planning benchmark that simulates real-world itinerary construction under hard constraints (e.g., timing, location compatibility) and soft commonsense preferences. We focus on the sole-planning setting where all relevant knowledge is embedded in the prompt, and no tool use is required. We evaluate on the validation set using the *hard constraint pass rate*, measuring whether the generated plan satisfies the minimal feasibility constraints (e.g., no overlaps or missing connections). We omit the stricter full success rate (which includes commonsense and preference matching) to isolate planning competence. Notably, even GPT-4-Turbo only achieves 22.2% under this setting, highlighting the dataset's difficulty.

F.2 LICENSES FOR EXISTING ASSETS

All models and datasets used in this work are publicly available and used in accordance with their respective licenses:

- **DeepSeek-R1-Distill-Qwen-32B and DeepSeek-R1-Distill-LLaMA-70B** (Guo et al., 2025) are released under the DeepSeek open-source model license available at `https://github.com/deepseek-ai/DeepSeek-LLM/blob/main/LICENSE-MODEL`.
- **QwQ-32B** (Team, 2025) is licensed under the Apache License 2.0.
- **OpenAI o4-mini** (OpenAI, 2025) is accessed via the OpenAI API under the terms of service and usage policies listed at `https://openai.com/policies/terms-of-use`. No model weights are released or modified.

- **LLaMA-3.1-8B-Instruct** is used via OpenRouter and follows Meta's LLaMA 3 license available at `https://ai.meta.com/llama/license/`.
- All datasets (MATH-500, NaturalInstructions, TravelPlanner) are publicly available and properly cited. They are used for evaluation purposes under academic and research-friendly terms of use.

F.3 COMPUTE RESOURCES

All experiments were conducted using API-accessible large language models, including OpenAI's o4-mini and models hosted via OpenRouter (e.g., DeepSeek-R1-Distill-Qwen-32B). Since our method operates entirely at inference time through prompting, the computational cost is directly proportional to the number of tokens generated. We report token usage for each setting in the main paper, which can be used to estimate wall-clock runtime given model-specific generation rates (typically 50–80 tokens/sec depending on the provider) and the parallelism used.

All data preprocessing, prompt generation, and evaluation were performed on a cloud-based virtual machine equipped with an Intel Xeon E5-2698 CPU and 500GB of main memory. No model training or fine-tuning was involved, and the overall compute requirements are modest and accessible.

G ADDITIONAL RESULTS

In addition to the answer pass rates discussed in the main results, we also examine the token usage distribution across queries of varying difficulty. Figure 4 presents the average token usage and corresponding pass rates on TravelPlanner, grouped by difficulty level.

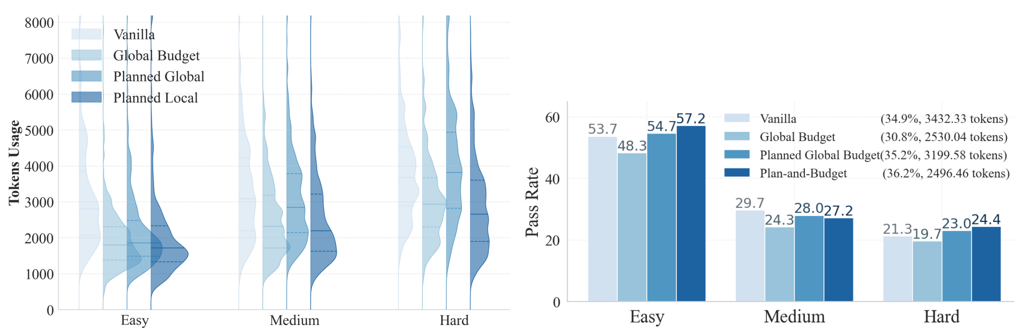

Figure 4: Token usage and pass rate analysis across difficulty levels on TravelPlanner. (Left) Token usage distributions. (Right) Answer pass rate by difficulty level.

As expected, we observe that token usage increases with query difficulty across all models—more complex tasks naturally require deeper reasoning and longer responses. However, methods using global budget constraints exhibit a consistently higher token usage across all difficulty levels compared to our approach (Planned Local). This suggests that global budgeting fails to adapt to query complexity, resulting in inefficient allocation of compute.

Moreover, global methods not only over-consume tokens but also suffer reduced answer pass rates at every difficulty level. This inefficiency leads to lower overall $\mathcal{E}^3$, further reinforcing the advantage of our uncertainty-aware local budgeting strategy. In contrast, PLAN-AND-BUDGET adapts compute based on sub-question difficulty, achieving better calibration of reasoning effort across simple and complex queries alike.

H PROMPT TEMPLATES

Prompt Templates for Question Decomposition

-Goal-
You are an experienced expert in domain and exam question designer. Your role is to help students break down challenging math problems into a series of simpler, high-level sub-questions.
We don't want too many detailed sub-questions, which are not beneficial for testing students' ability in an exam. Each sub-question should build on the previous one so that, once all have been answered, the complete solution is clear.
Your output should be a list of sub-questions with brief hints explaining the purpose of each step, but you should not reveal your internal chain-of-thought either the final solution.

Instructions for Decomposition:
First, analyze the problem and identify the key ideas needed to solve it. Then, generate a series of 2 to 5 sub-questions that lead the student step by step to the complete solution. The difficulty level of the problem is presented out of 5, wher 1 is easy, and 5 is hard. Please adjust the number of sub-questions based on the level. Ideally, we want fewer sub-questions for easy problems and more sub-questions for challenging problems.
DO NOT perform reasoning, directly output those sub-questions based on your gut feelings; only output the list of sub-questions with brief hints for each.
Your answer should be a list of numbered sub-questions. Each sub-question should have a brief accompanying hint that explains what the student will achieve by answering that part.

Example Decomposition:
Problem: Find the remainder when $(9 \times 99 \times 999 \times \cdots \times \underbrace{99\cdots9}_{\text{999 9's}})$ is divided by 1000.

Level: 3 out of 5

Decomposed Sub-questions:

1. Compute the product modulo 8.
Hint: Simplify each term using $(10 \equiv 2 \mod 8)$, noting that $(10^k \equiv 0 \mod 8)$ for $k \geq 3$, leading to terms of $(-1 \mod 8)$.

2. Compute the product modulo 125.
Hint: Recognize $(10^3 \equiv 0 \mod 125)$, so terms for $(k \geq 3)$ become $(-1 \mod 125)$. Calculate the product of the first two terms and combine with the remaining terms.

3. Solve the system of congruences using the Chinese Remainder Theorem.
Hint: Combine the results from modulo 8 and modulo 125 to find a common solution modulo 1000.

A student has presented you with the following math problem:
Problem: <problem>
Level: <level> out of 5
REMEMBER, you are not allowed to think about it, please directly generate the answer in the following:
Decomposed Sub-questions:

Prompt Templates for Question Decomposition

You are an experienced expert in <domain> and exam question designer. Your task is to evaluate the difficulty level of a given exam problem and its sub-questions by comparing it

against a set of benchmark questions of known levels.
Based on their levels, you will need to assign each subquestion a portion of the credits (assuming the total credit points is 100 for the whole problem).

Each level reflects increasing complexity from 1 (easiest) to 5 (most challenging). Evaluate based on the conceptual depth, steps involved in solving, required knowledge, and potential for misdirection.

Use the following benchmark examples as references:

<benchmarks>

1. You will be provided a question and its subquestions. You will evaluate the difficulty level of the problem and its sub-questions.
Assuming the whole problem is worth 100 points, you assign each sub-question a portion of the score points.
- Adhere to the given subquestions, and DO NOT make new subquestions.
- Sum of each subquestion's credits MUST EQUAL to 100.

2. You must return the result in a structured JSON format:
```
{
"problem": {"reason": "...", "evaluated_level": level_q}
"1": {"reason": "...", "evaluated_level": level_1, "credit": credit_1},
"2": {"reason": "...", "evaluated_level": level_2, "credit": credit_2},
...}
```
where
- "reason": a short explanation (up to 50 words) of your level assessment.
- "evaluated_level": an integer from 1 to 5 indicating your judgment.
- "credit": an integer between 1 to 100 indicating when the question is solved correctly, how many credit can be given.

Evaluate the level of the following question:
Problem: <problem>
Sub-questions: <steps>
Output:

Prompt Templates for Vanilla Model

< dataset-specific instruction>

Please reason step by step, and conclude your answer in the following format:

<dataset specific output format>

Question: <query>
Reference: <reference>(only applicable to TravelPlanner)
Output: <think>

Prompt Templates for Global Budget Model

< dataset-specific instruction>

Please reason step by step, and conclude your answer in the following format:

<dataset specific output format>

Question: <query>
Reference: <reference>(only applicable to TravelPlanner)
Let's think step by step and use less than <budget> tokens. Output: <think>

Prompt Templates for Planned Vanilla Model

<dataset-specific instruction>

The problem is given by an overall description, difficulty level out of 5, followed by a series of sub-questions as a hint.
All the credit is given when you provide a correct final answer for the overall problem.
Please solve the question efficiently and clearly to achieve as much credit as possible.

Let's start the exam. You are being given this math problem:
Problem (100pt): <query>
Reference: <reference>(only applicable to TravelPlanner)
Level: <level> out of 5

You may think following these sub-questions or feel free to use other methods that works the best towards getting the final answer:
<decomposed>

Please provide your final answer in the following format:
<dataset specific output format>

Output: <think>

Prompt Templates for Planned Global Budget Model

<dataset-specific instruction>

The problem is given by an overall description, difficulty level out of 5, followed by a series of sub-questions as a hint.
All the credit is given when you provide a correct final answer for the overall problem.
Please solve the question efficiently and clearly to achieve as much credit as possible.

Let's start the exam. You are being given this math problem:
Problem (100pt): <query>
Reference: <reference>(only applicable to TravelPlanner)
Level: <level> out of 5

You may think following these sub-questions or feel free to use other methods that works the best towards getting the final answer:
<decomposed>

Please provide your final answer in the following format:
<dataset specific output format>

Let's think step by step and use less than <budget> tokens.
Output: <think>

Prompt Templates for Planned Local Budget Model (Ours)

<dataset-specific instruction>

The problem is given by an overall description, difficulty level out of 5, followed by a series of sub-questions as a hint.
All the credit is given when you provide a correct final answer for the overall problem.
Please solve the question efficiently and clearly to achieve as much credit as possible.

Let's start the exam. You are being given this math problem:
Problem (100pt): <query>
Reference: <reference>(only applicable to TravelPlanner)
Level: <level> out of 5

You may think following these sub-questions or feel free to use other methods that works the best towards getting the final answer:
<decomposed> (For each decomposed subquestion:) Please only think a little, and directly solve it using up to <budget> words.

Please provide your final answer strictly following the format:
<dataset specific output format>

Output: <think>