Plan and Budget: Effective and Efficient Test-Time Scaling on Large Language Model Reasoning

Junhong Lin[1], Xinyue Zeng[2], Jie Zhu[2], Song Wang[3]
Julian Shun[1], Jun Wu[4], Dawei Zhou[2]

[1]Massachusetts Institute of Technology, [2]Virginia Tech
[3]University of Virginia, [4]Michigan State University

April 7, 2026

Reasoning Miscalibration

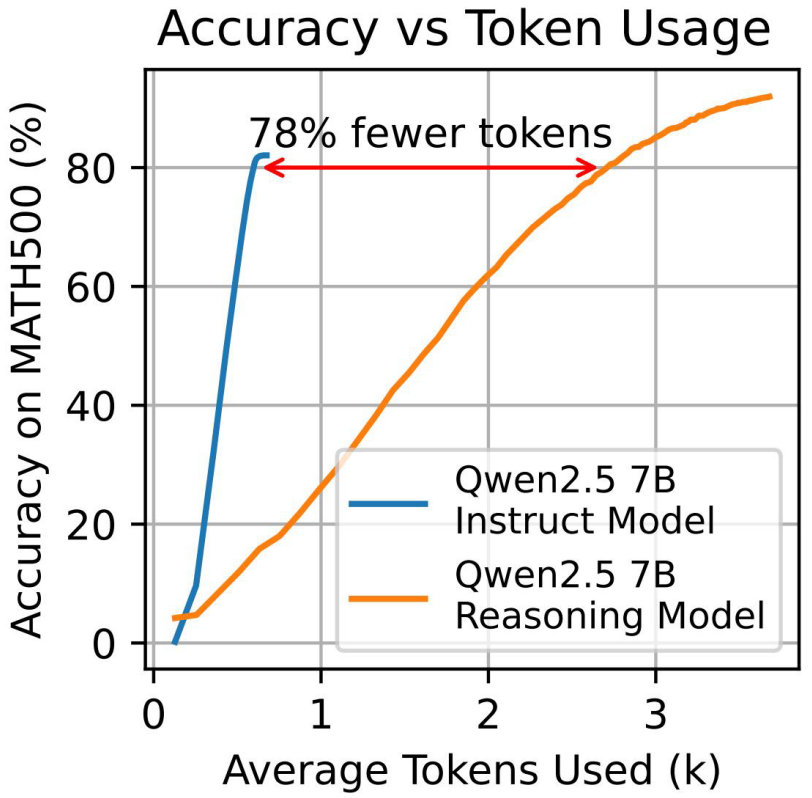
Reasoning Improves Accuracy - But at What Cost?

- **Recent Advances in Test-Time Scaling**
  - Chain-of-Thought (CoT)
  - Self-Consistency
  - Tool-augmented reasoning

→ Higher asymptotic accuracy

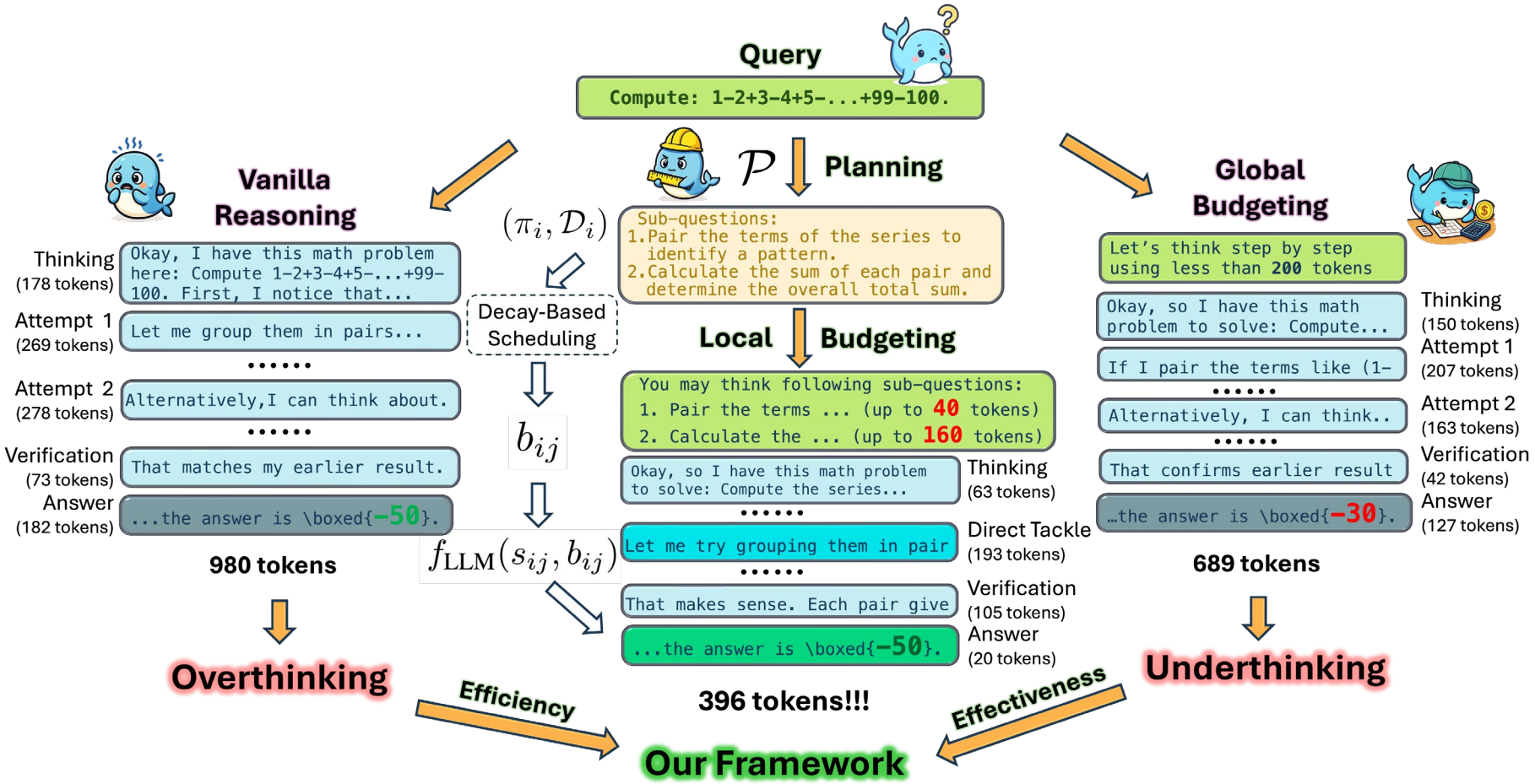
- **But…**
  - Accuracy improves slowly with more tokens
  - Token usage grows rapidly
  - Diminishing marginal returns
- **Key Question**

Can we achieve deep reasoning without paying the full token cost?



Reasoning improves accuracy, but in a highly inefficient way .

Reasoning Miscalibration



🔵 **Overthinking**

→ verbose, tangential reasoning

🔴 **Underthinking**

→ premature termination

When Does Reasoning Go Wrong?

🟦 Case 1 — Trivial but Ambiguous → *Overthinking*

Example: “Is 2 + 3 even?”

Model may busy:

- Define parity
- Consider modular arithmetic
- Provide multiple formulations
- Re-verify (let me count my fingers)
- ....

Model keeps reducing something that cannot be reduced further.

🟥 Case 2 — Hard and Rare → *Underthinking*

Example: “How many 4-digit numbers can be formed using digits {1,2,3,4,5}, if no digit is repeated and the number is even? Answer in **5 seconds**”

Model may say:

“There are 5 digits, choose 4 → 5P4 = 120, I think half of them are even → 60”

- No verification
- No intermediate check
- Stops early

The model fails to invest where uncertainty is reducible.

It’s not about too many tokens or too few tokens.

It’s about allocating compute incorrectly.

When Does Reasoning Go Wrong?

Case 1 — Trivial but Ambiguous → *Overthinking*

Example: "Is 2 + 3 e

Model may busy:

- Define parity
- Consider mod
- Provide multip
- Re-verify (let m
- ....

Model keeps reducing something that cannot be reduced further.

Case 2 — Hard and Rare → *Underthinking*

ers can be formed

epeated and the

ls"

120, I think half

- Stops early

The model fails to invest where uncertainty is reducible.

How can we guide LLMs to allocate computation adaptively based on task complexity?

We reinterpret reasoning as a **resource allocation problem under uncertainty**.

It's not about too many tokens or too few tokens.

It's about allocating compute incorrectly.

Two Types of Uncertainty in Reasoning

- When a model is unsure, why is it unsure?

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

🟩 **Epistemic Uncertainty (Reducible)**

The model doesn't know enough —
but more reasoning could help.

Query: "Compute 37 × 48."

If the model hesitates, it's because:

- It hasn't completed the multiplication
- It hasn't derived the intermediate steps

More reasoning → uncertainty decreases.

🟧 **Aleatoric Uncertainty (Irreducible)**

The ambiguity comes from the input itself.
More thinking doesn't help.

Query: "Is the bank open?"

- Which bank?
- Which city?
- Which time?

Even infinite reasoning can't resolve missing info.

Reasoning as a Sequence of Sub-Problems

- LLM reasoning is **multi-step** — implicitly solving a chain of sub-questions.
- Different sub-questions have **different uncertainty profiles**:
  - Some steps are structural / strategy (high epistemic → benefit from more compute)
  - Some steps are routine execution (low epistemic → extra tokens are wasteful)

Compute: $1 - 2 + 3 - 4 + \cdots + 99 - 100$

Sub-questions:

- Pair terms to identify a pattern
- Compute the sum of each pair
- Aggregate to get the final total

- **Miscalibration happens when we spend too much compute on easy steps, and too little on hard steps.**

From Theory to Practice:
Plan-And-Budget

Budget **A**llocation **M**odel (BAM)

- We start with a simple assumption inspired by neural scaling laws
  - Allocating more tokens reduces epistemic uncertainty, but with diminishing returns.
  - Uncertainty decreases roughly as a power law with compute:

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

s: sub-questions
b: tokens allocated
c: initial uncertainty
β: difficulty parameter

- Given a total token budget B, the optimal allocation of budget across subquestion is:

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

Allocate more tokens to sub-questions that:
- Start with high uncertainty (large c)
- And where additional compute is still effective (moderate β)

- Extremely easy steps get few tokens.
- Extremely hard steps also get few tokens.
- Moderately difficult steps get the most.

Efficiency-aware **E**ffectiveness **E**valuation Score ($\mathcal{E}^3$)

- Many existing works measure efficiency as accuracy per token:

$$A/T$$

- We propose to weights the A/T metric by accuracy to favor methods that maintain high correctness:

$$\mathcal{E}^3 = A^2/T$$

Example

For **Model A** (Acc = **90%**, Tokens = **2000**), and **Model B** (Acc = **45%**, Tokens = **1000**), they achieve the **same A/T score**:

$$A/T = 0.00045$$

Example

For Model A :

$$\mathcal{E}^3 = \frac{0.9^2}{2000} = 0.000405$$

For Model B:

$$\mathcal{E}^3 = \frac{0.45^2}{1000} = 0.000203$$

Now the scores are different: the high-accuracy model is strongly preferred.

Plan-and-Budget:

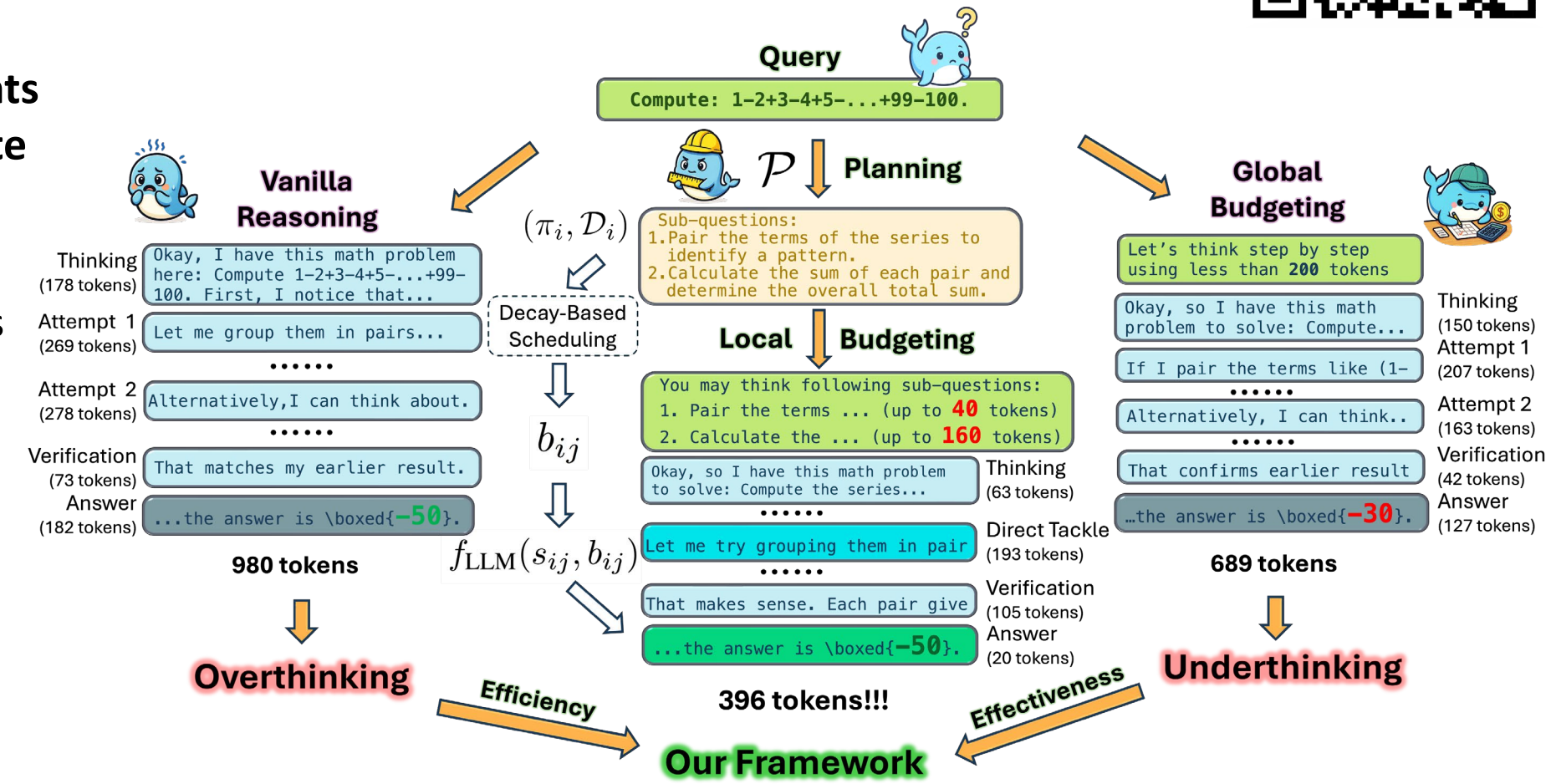
Structured & Adaptive Reasoning

arXiv 2505.16122

License MIT

Plan-and-Budget implements uncertainty-guided compute allocation:

- **Planning:** Decompose queries into sub-questions
- **Local Budgeting:** Allocate tokens adaptively across steps
- **Uncertainty-Aware:** Reduces epistemic uncertainty efficiently
- **Training-Free:** Works with any LLM at test time



Plan-and-Budget:
General Workflow

arXiv 2505.16122

Query

Step 1: Plan
(Structure the Reasoning)

- Decompose query into sub-questions
- Estimate relative difficulty
- Provide a soft reasoning scaffold

Purpose: Reduces speculative exploration.

Step 2: Budget
(Allocate Compute Locally)

- Assign token budgets per sub-question strategically
- Use uncertainty-aware scheduling
- Adapt allocation within fixed total budget

Purpose: Aligns compute with epistemic uncertainty.

Reasoning LLM

Final Answer

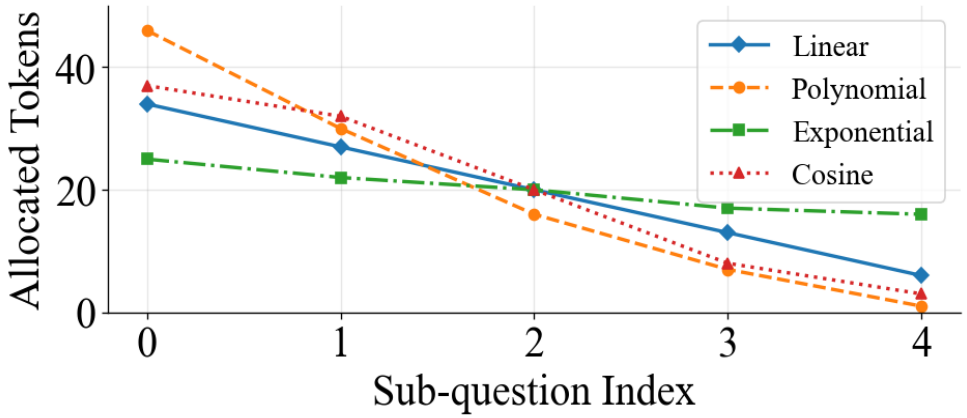
Plan-and-Budget:

Practical Budget Allocation via Decay Scheduling

arXiv 2505.16122

We approximate optimal allocation using simple decay schedules:

- Allocate more tokens to early (high-uncertainty) steps.
- Gradually reduce budget across sub-questions.
- Different decay shapes reflect different assumptions.
- All schedules are lightweight and training-free.



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.

Results

Experiment Setup

🟦 Tasks

3 key tasks covering math, general reasoning, and long-horizon planning:

- **MATH-500**
  - Multi-step symbolic reasoning
  - Exact match accuracy
- **NaturalInstructions**
  - Instruction following
  - ROUGE score
- **TravelPlanner**
  - Constraint-heavy planning
  - Hard constraint pass rate

🟥 Models

4 main-stream reasoning LLM without fine-tuning:

- **DS-Qwen-32B**
- **QwQ-32B**
- **DS-LLaMA-70B**
- **o4-mini**
- **Vanilla reasoning**
- **Global budget constraints**
- **Planned global budgeting**
- **Plan-and-Budget (ours)**

🟩 Metrics

- Accuracy / ROUGE / Pass rate
- Average tokens (completion tokens)
- $E^3 = \frac{A^2}{T}$

Extensive Results

- We conduct extensive experiment covering 3 kinds of tasks X 4 LLMs
- And you are probably too busy to read through them…

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.

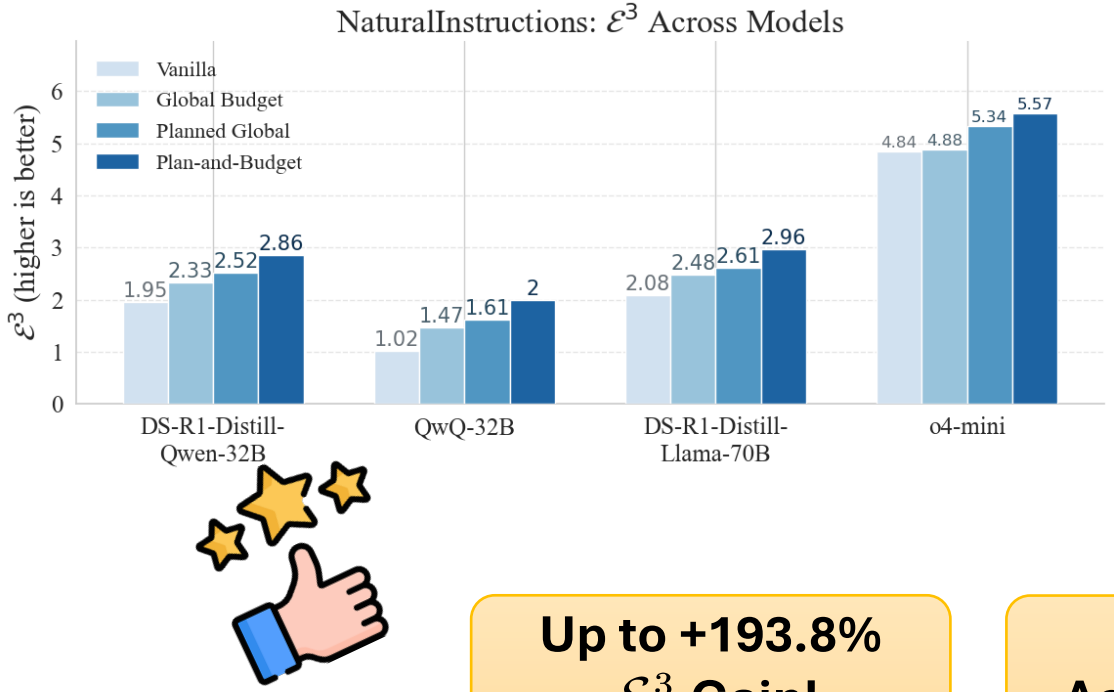
	Models →	DeepSeek-R1-Distill-Qwen-32B			QwQ-32B			DeepSeek-R1-Distill-Llama-70B			o4-mini		
	Methods↓	ROUGE (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	ROUGE (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	ROUGE (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	ROUGE (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑
Direct	Vanilla	**$43.47_{\pm 0.52}$**	$968.17_{\pm 44.78}$	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

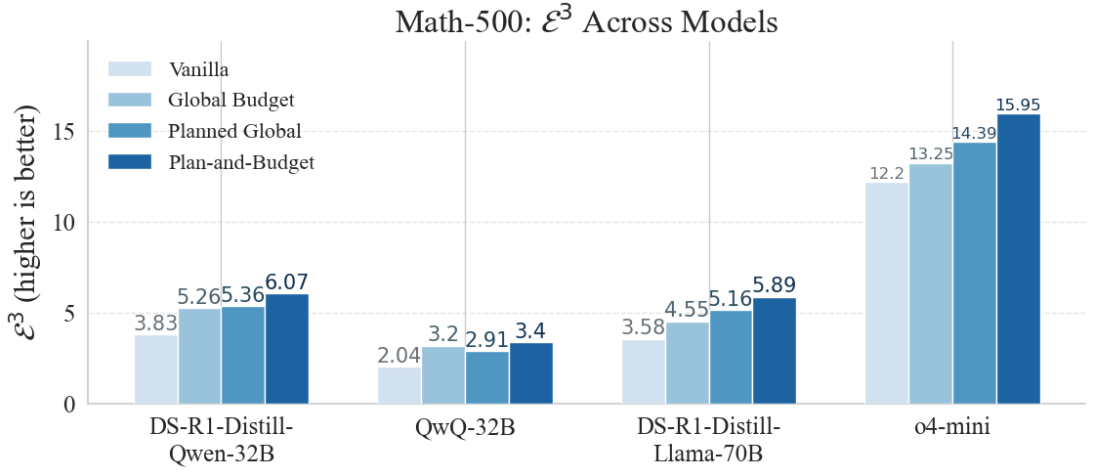
Table 5: Experiment results on TravelPlanner. Rate denotes the hard constraint pass rate.

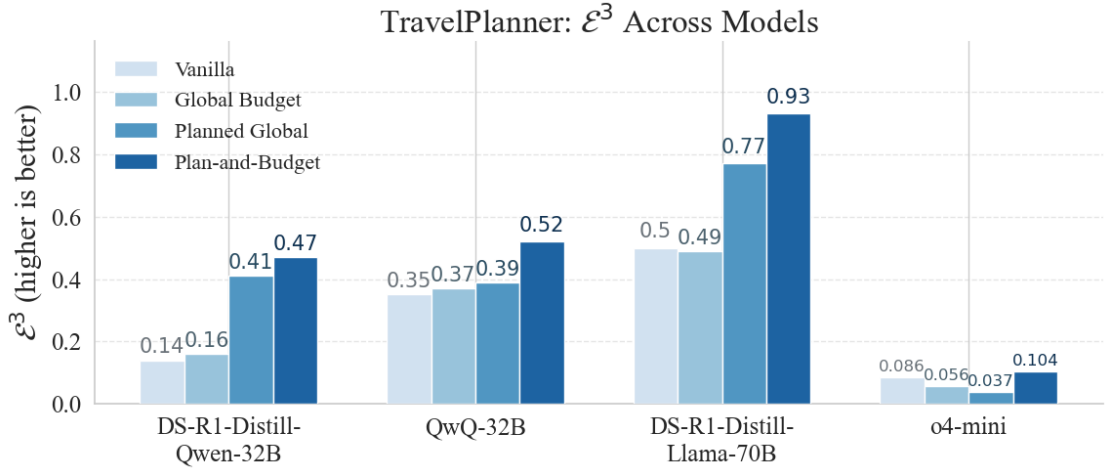
	Models →	DeepSeek-R1-Distill-Qwen-32B			QwQ-32B			DeepSeek-R1-Distill-Llama-70B			o4-mini		
	Methods↓	Rate (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	Rate (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	Rate (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑	Rate (%) ↑	Avg. Tokens ↓	$\mathcal{E}^3$ ↑
Direct	Vanilla	$14.33_{\pm 2.17}$	$1430.14_{\pm 43.73}$	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	$8.33_{\pm 1.71}$	**$1248.53_{\pm 26.97}$**	0.056
Planned	Vanilla	$20.22_{\pm 1.01}$	$1343.67_{\pm 62.44}$	0.30	**$37.22_{\pm 1.80}$**	$3669.88_{\pm 42.09}$	0.38	$30.67_{\pm 2.17}$	$1464.50_{\pm 65.40}$	0.64	**$12.20_{\pm 2.47}$**	$1640.46_{\pm 95.33}$	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	**$23.33_{\pm 1.11}$**	$1222.09_{\pm 40.69}$	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	$19.56_{\pm 2.47}$	**$1136.18_{\pm 54.92}$**	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	**$32.67_{\pm 2.06}$**	**$1148.14_{\pm 59.00}$**	**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	$36.18_{\pm 3.00}$	**$2496.46_{\pm 40.10}$**	**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

Adaptive Allocation Improves Efficiency

- Consistent gains across datasets
- Largest improvement: **+193.8%**
- Improvement holds across model scales







Up to +193.8% $\mathcal{E}^3$ Gain!

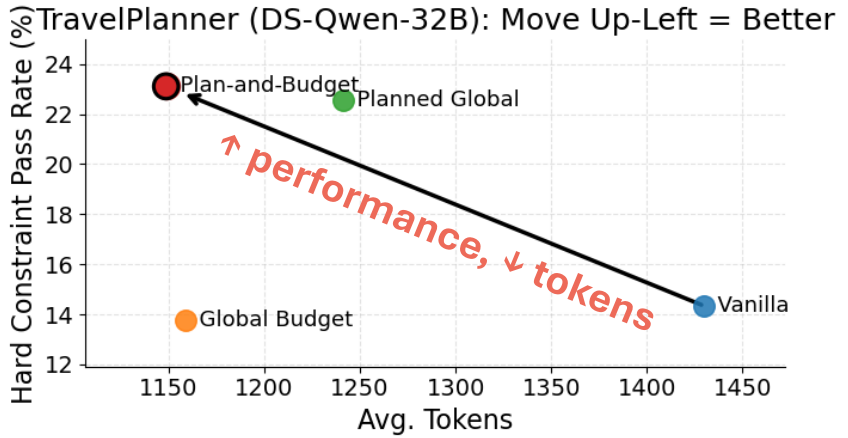
Up to +70% Accuracy Gain!

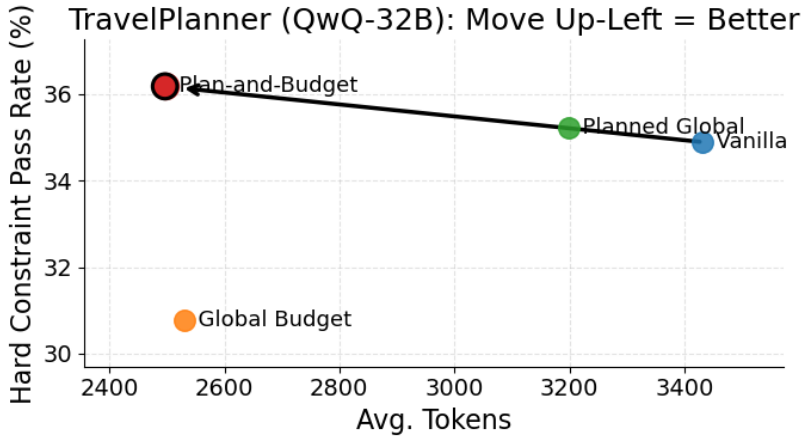
Up to -39% Token Reduction!

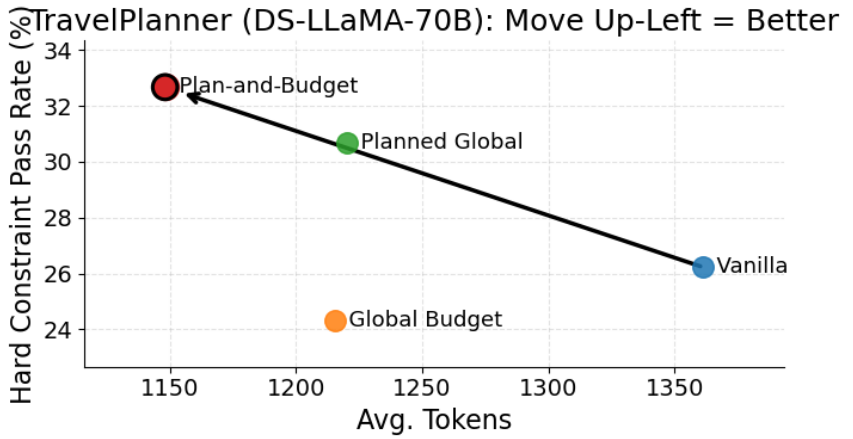
Adaptive Allocation Improves Performance at Lower Cost

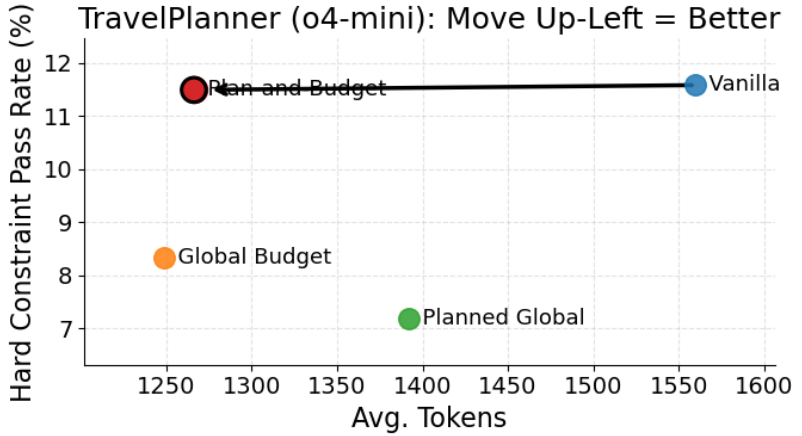
- Left = **fewer↓** tokens
- Up = **higher↑** correctness
- Global budget
  → left but down (underthink)
- Plan-and-Budget
  → **left and up** (calibrated reasoning)

Efficient reasoning requires calibrated compute allocation, not simply reducing tokens.



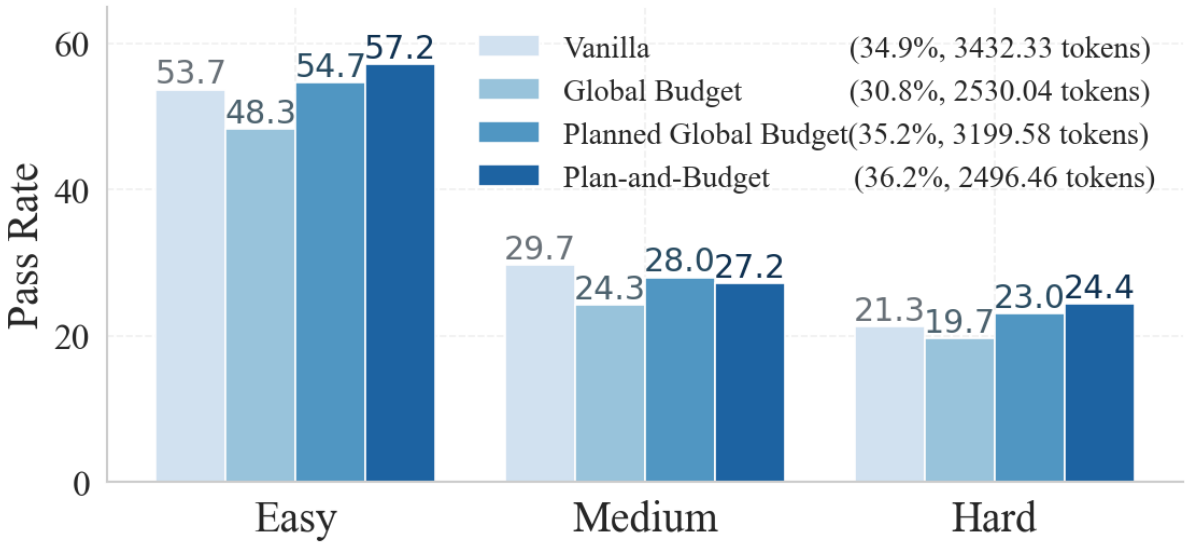






Global Budgets Misallocate Compute Across Difficulty

- Global budget uniformly reduces tokens
  → performance drops across difficulty levels
- Plan-and-Budget reallocates compute
  → improves Easy and Hard queries
- Effects vary across models and datasets
  → consistent trend is adaptive redistribution



The problem is not difficulty alone, it is uniform compute allocation across heterogeneous queries.

Key Takeaways

1. **Reasoning Miscalibration is a Core Bottleneck:** LLMs often misallocate compute at test time:
   - overthinking on simple queries,
   - underthinking on complex ones.
2. **Adaptive Allocation Improves Efficiency and Accuracy:** Plan-and-Budget:
   - decomposes reasoning into structured sub-problems,
   - allocates tokens based on uncertainty,
   - achieves better performance with fewer tokens.
3. **Structured Control Enables Efficient Reasoning**

Compute should be allocated based on uncertainty,
not applied uniformly across reasoning steps.