

Optimizing Test-Time Compute via Meta Reinforcement Finetuning

Yuxiao Qu*, Matthew Y. R. Yang*, Amrith Setlur, Lewis Tunstall,
Edward Emanuel Beeching, Russ Salakhutdinov, Aviral Kumar

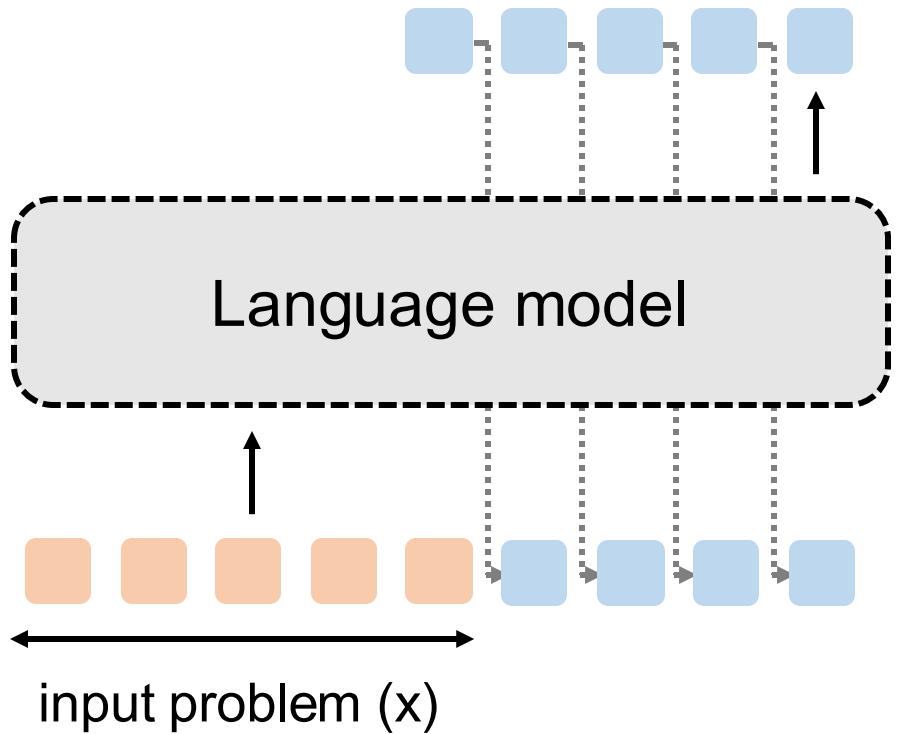


Hugging Face

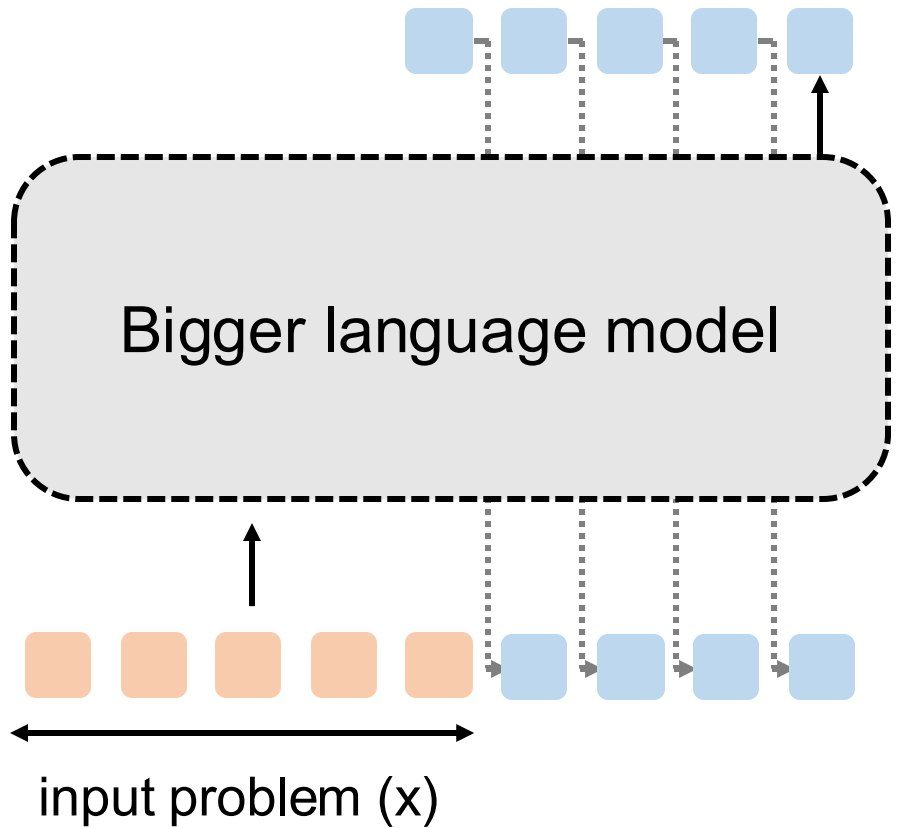


**Carnegie
Mellon
University**

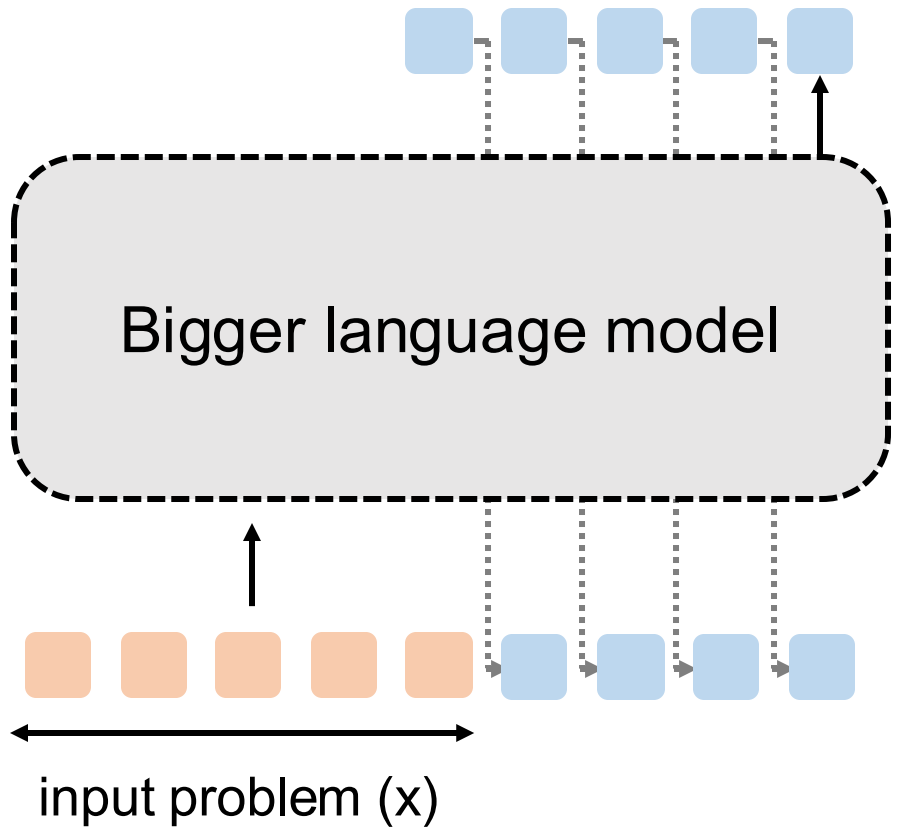
Test-Time Scaling for Large Language Models



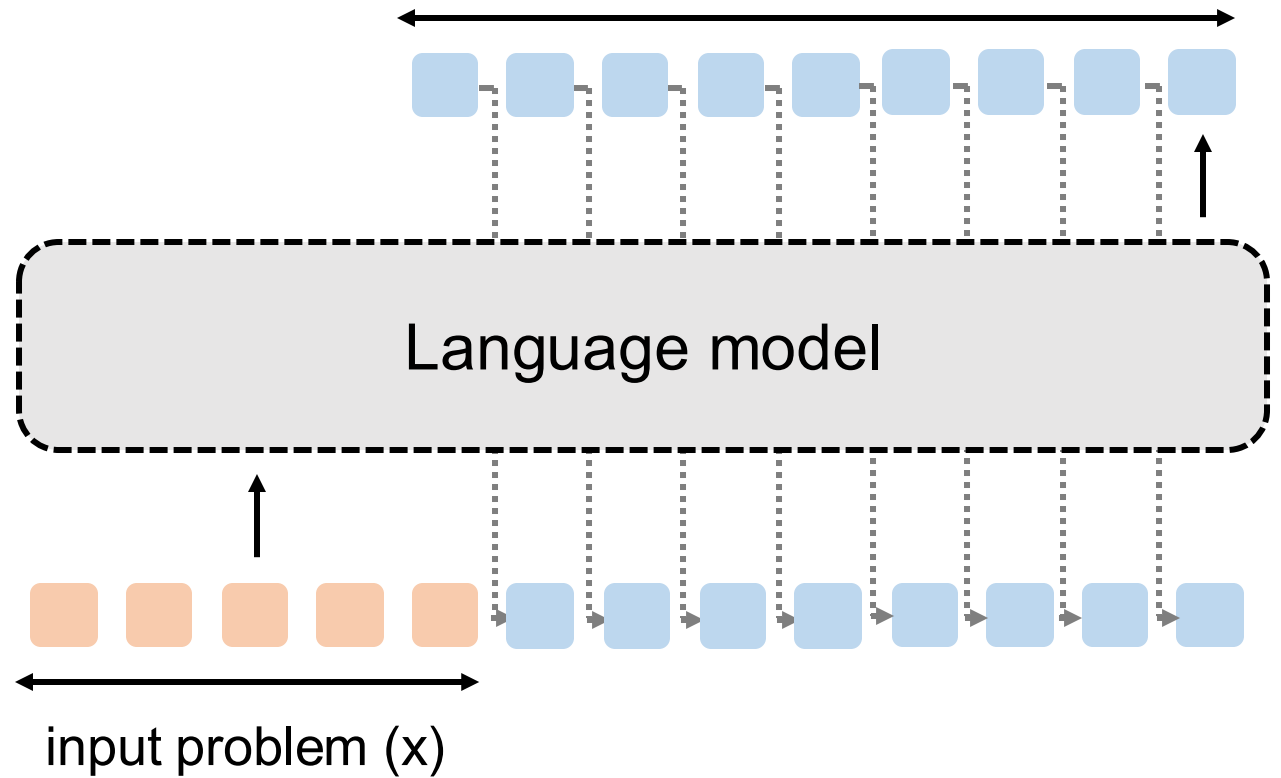
Test-Time Scaling for Large Language Models



Test-Time Scaling for Large Language Models



longer responses, "more thinking", self-correction, etc.



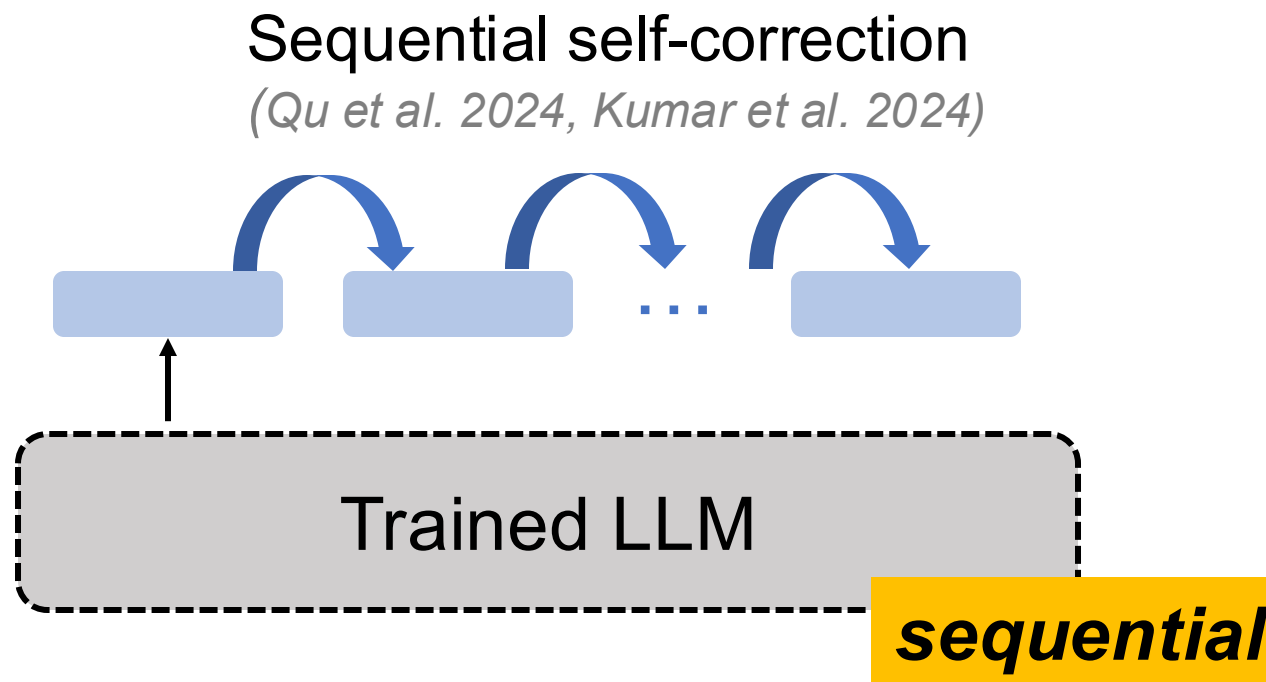
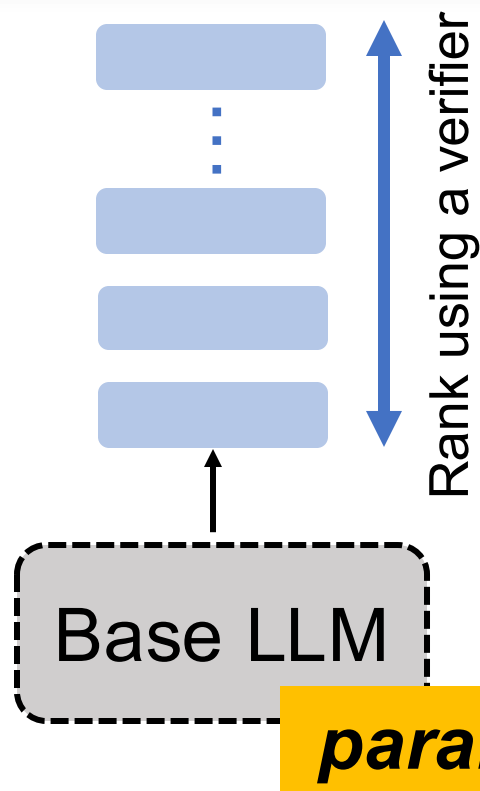
Finetuning LLMs is Critical for Test-Time Scaling

Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters

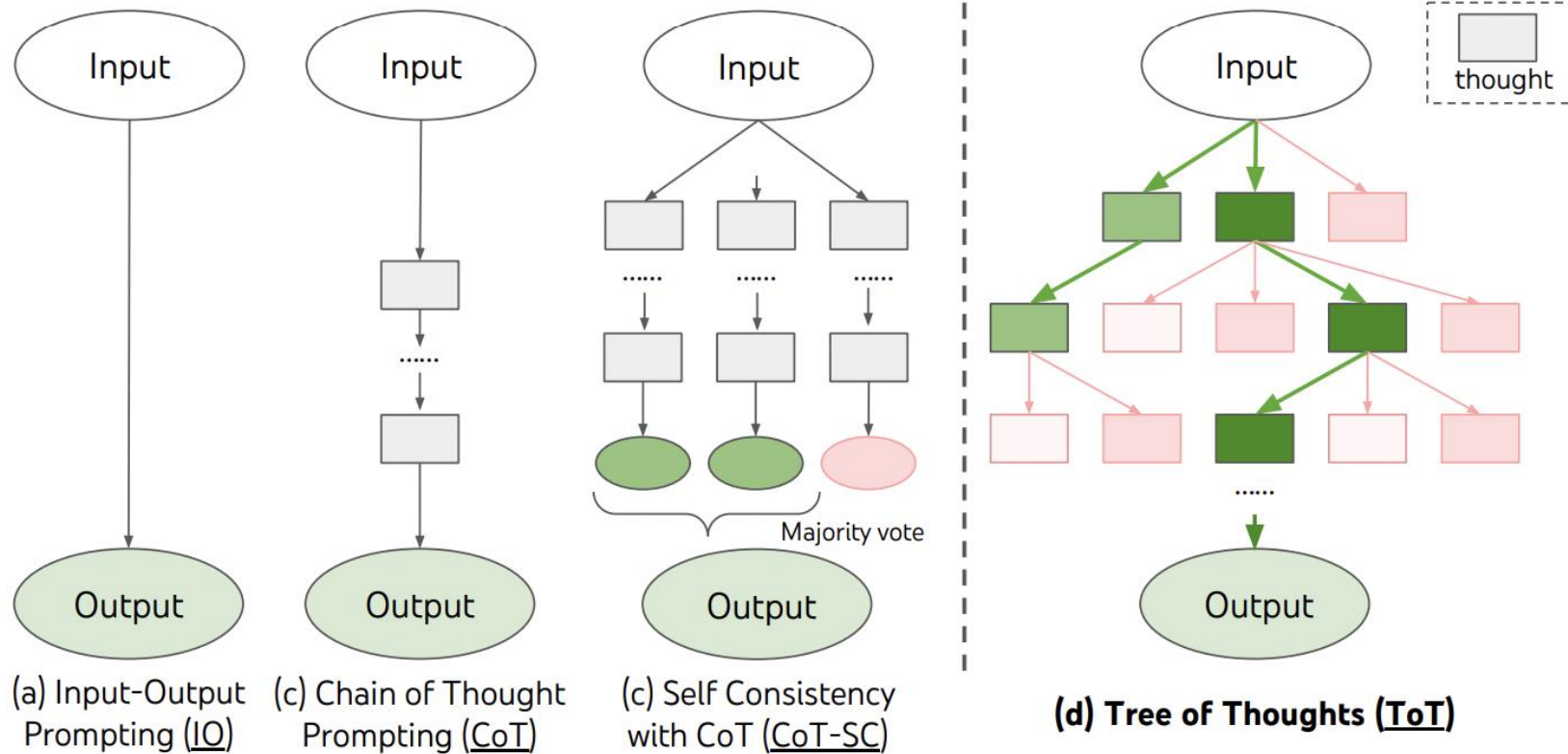
Charlie Snell^{✦, 1}, Jaehoon Lee², Kelvin Xu^{✦, 2} and Aviral Kumar^{✦, 2}

[✦]Equal advising, ¹UC Berkeley, ²Google DeepMind, [✦]Work done during an internship at Google DeepMind

Key idea: Finetune LLMs to enable this behavior!



A Family of Test-Time Scaling Algorithms



This training has been done via:

- RL or SFT
- Dense vs sparse supervision
- Different “types” of training data
-

Training for test-time scaling: Train LLMs to implement these procedures

Desiderata: What Do We Want at Test Time

Question: $2+2 = ?$

<think>

Okay, so I need to figure out what 2 plus 2 is. Let me start by recalling basic addition. When you add two numbers, you're combining their values. So if I have 2 apples and someone gives me 2 more apples, how many apples do I have in total?

Let me count them out. First, I have 1, 2 apples. Then I add another 2 apples: 3, 4. So that makes 4 apples altogether. *Wait, is that right? Let me check again. 2 plus 2... If I take the number 2 and then add 2 more, it should be one more than 2 plus 1, which is 3, so adding another 1 would make it 4. Yeah, that seems correct.....*

Desideratum 1: Don't spend too many tokens on easy questions!

Desiderata: What Do We Want at Test Time

Show that the inequality

$$\sum_{i=1}^n \sum_{j=1}^n \sqrt{|x_i - x_j|} \leq \sum_{i=1}^n \sum_{j=1}^n \sqrt{|x_i + x_j|}$$

holds for all real numbers x_1, x_2, \dots, x_n .

Wait, but if we can bound $\sum \sqrt{|x_i - x_j|} \leq \sum \sqrt{|x_i + x_j|}$, and we know that $\sum \sqrt{|x_i + x_j|} + \sum \sqrt{|x_i - x_j|} \leq 2 \sum \sqrt{|x_i| + |x_j|}$, then combining these would give $2 \sum \sqrt{|x_i - x_j|} \leq 2 \sum \sqrt{|x_i| + |x_j|}$, which would imply $\sum \sqrt{|x_i - x_j|} \leq \sum \sqrt{|x_i| + |x_j|}$. But this is different from our original inequality. Hmm.

Alternatively, maybe consider that for each pair (i, j) , $\sqrt{|x_i + x_j|} + \sqrt{|x_i - x_j|} \geq 2 \sqrt{|x_i|}$, as a form of the arithmetic mean - geometric mean inequality? Let's test with $x_i = 1, x_j = 1$: $\sqrt{2} + 0 \approx 1.414 \geq 21 = 2$. No. $x_i = 2, x_j = 0$: $\sqrt{2} + \sqrt{2} \approx 2.828 \geq 2\sqrt{2} \approx 2.828$. Equality. $x_i = 3, x_j = 1$: $\sqrt{4} + \sqrt{2} \approx 2 + 1.414 \approx 3.414 \geq 2\sqrt{3} \approx 3.464$. No, still less. So this inequality doesn't hold.

This is getting frustrating. Maybe I need to look for a different approach. Let's think about the original inequality again. It's symmetric in all x_i . Maybe consider some symmetry such as replacing x_i with $-x_i$. If we replace each x_i with $-x_i$, the left side becomes $\sum \sqrt{|-x_i + x_j|} = \sum \sqrt{|x_j - x_i|} =$ same as original left side. The right side becomes $\sum \sqrt{|-x_i - x_j|} = \sum \sqrt{|x_i + x_j|} =$ same as original right

Desideratum 2: Keep making *progress* on hard questions, even beyond compute budget

Problem Setup: LLM Math Reasoning Problems

Initial state

Problem: Suppose a and b are positive real numbers with $a > b$ and $ab = 8$. Find the minimum value of $\frac{a^2+b^2}{a-b}$.

Ground truth solution: We can write $\frac{a^2+b^2}{a-b} = \frac{a^2+b^2-2ab+16}{a-b} = \frac{(a-b)^2+16}{a-b} = a - b + \frac{16}{a-b}$. By AM-GM, $a - b + \frac{16}{a-b} \geq 2\sqrt{(a-b) \cdot \frac{16}{a-b}} = 8$. Equality occurs when $a - b = 4$ and $ab = 8$. We can solve these equations to find $a = 2\sqrt{3} + 2$ and $b = 2\sqrt{3} - 2$. Thus, the minimum value is $\boxed{8}$.

Steps = actions

reward = 1 if
answer is correct

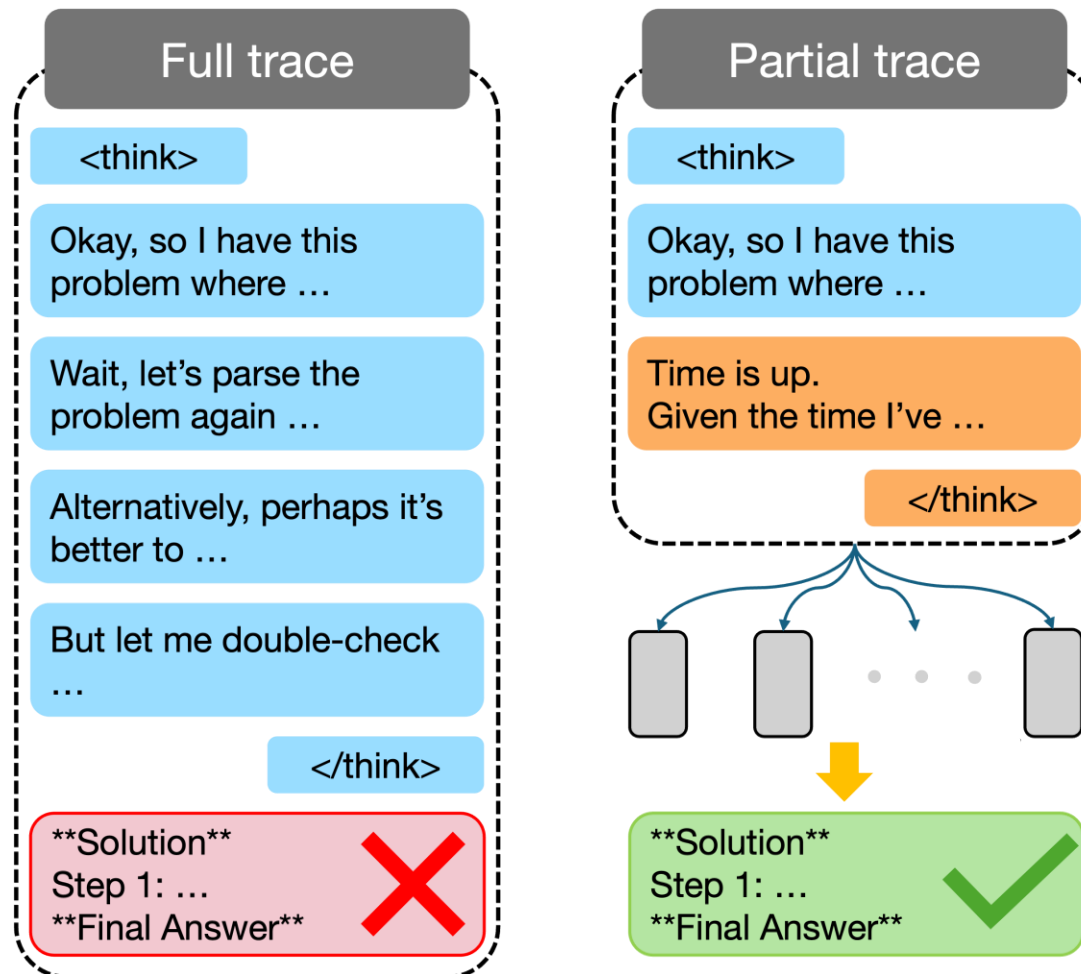
Do Current Models Enjoy these Desiderata?

Short answer: *not really!*

Experiment setup

Chop the thinking block in DeepSeek-R1 and ask it to produce best answer

- **Easy problems:** Make sure to be efficient
- **Hard problems:** Make sure to make constant *progress*



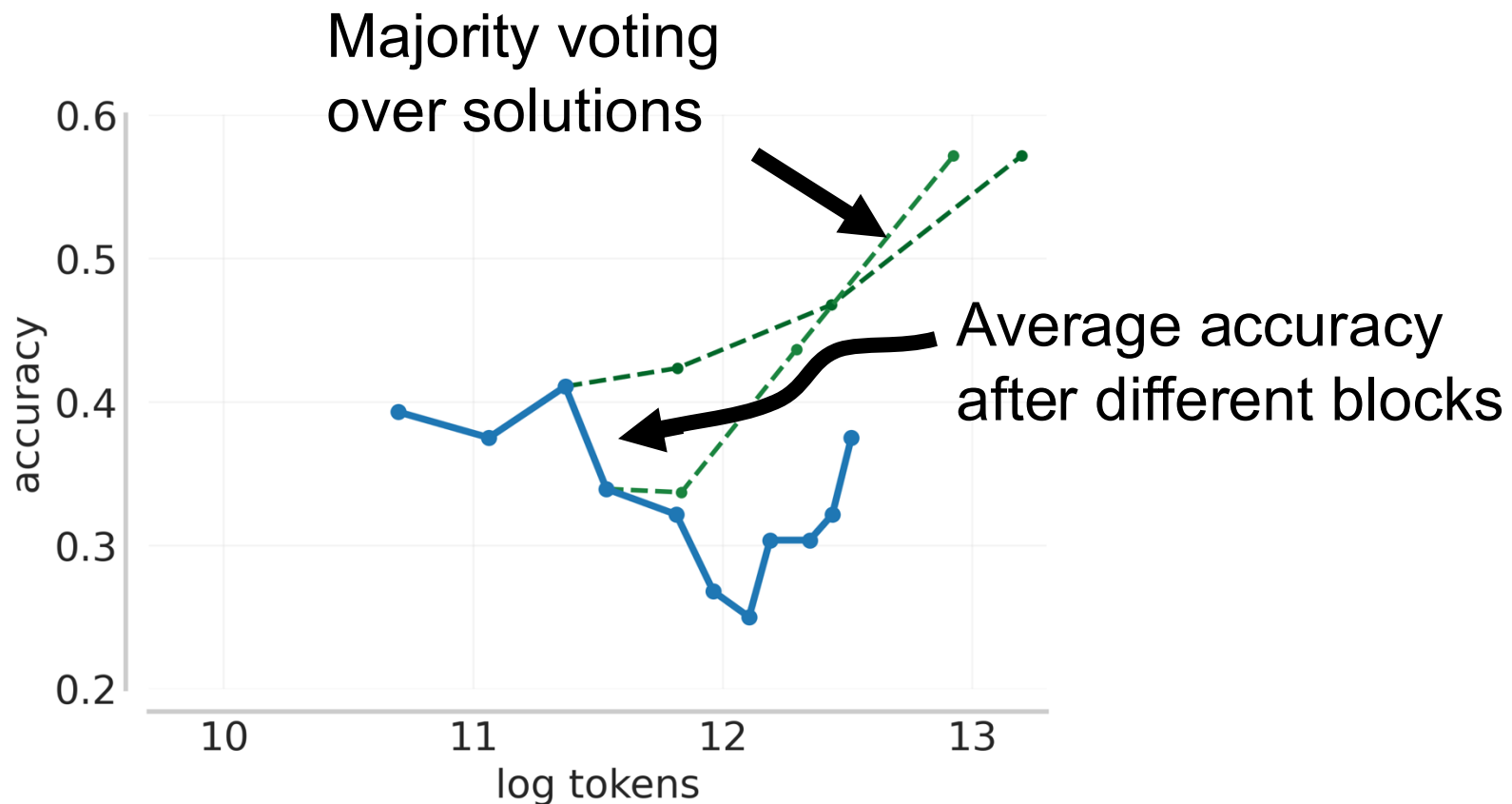
Do Current Models Enjoy these Desiderata?

Short answer: *not really!*

- **Easy problems:** Make sure to be *token-efficient*!
- **Hard problems:** Make sure to make constant *progress*

Experiment setup

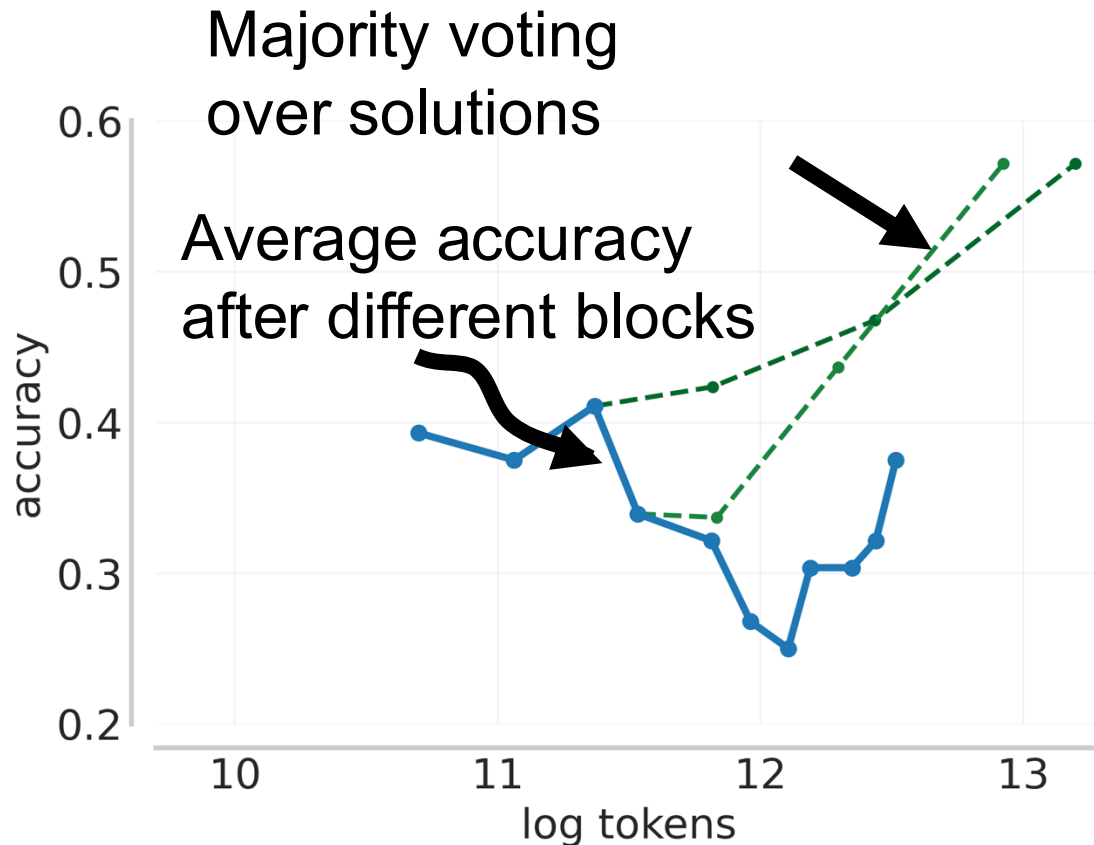
Chop the thinking block in DeepSeek-R1 and ask it to produce best answer



Do Current Models Enjoy these Desiderata?

Short answer: *not really!*

- **Easy problems:** Make sure to be *token*-efficient!
- **Hard problems:** Make sure to make constant *progress*



Takeaway: Can make progress by implementing the “algorithm” of running a simple majority vote, *but it does not.*

Formulation: How to Satisfy These Desiderata

Let's start from the final goal

$$\max_{\pi} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{test}}} \left[\mathbb{E}_{\mathbf{z} \sim \pi(\cdot | \mathbf{x})} [r(\mathbf{x}, \mathbf{z})] \right]$$

on test problems

response sampled
from model (longer
than typical solution)

Total compute constraint
per problem

Formulation: How to Satisfy These Desiderata

$$\max_{\pi} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{test}}} [\mathbb{E}_{\mathbf{z} \sim \pi(\cdot|\mathbf{x})} [r(\mathbf{x}, \mathbf{z})]] \quad \text{s.t.} \quad \forall \mathbf{x}, \mathbb{E}_{\pi(\cdot|\mathbf{x})} |\mathbf{z}| \leq C_0$$



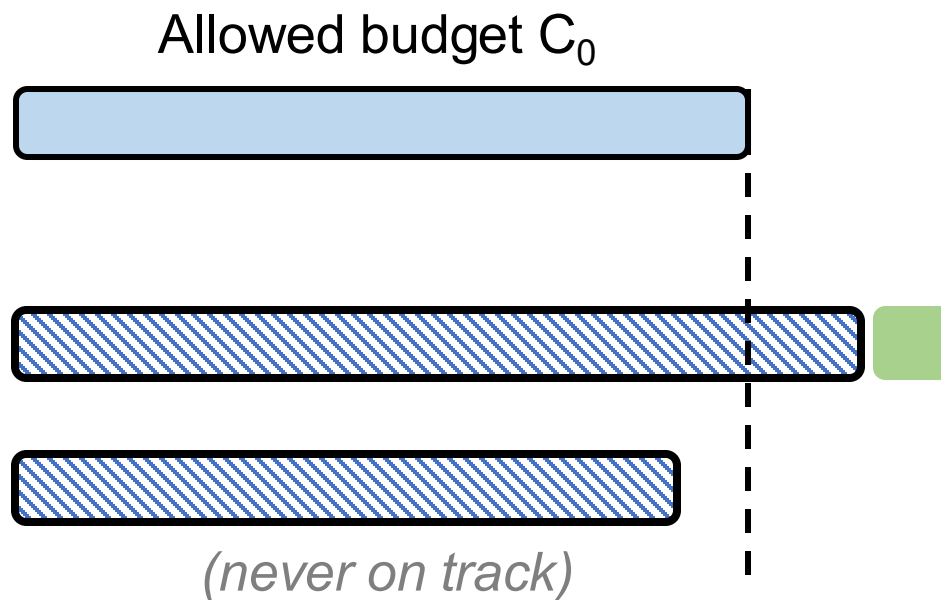
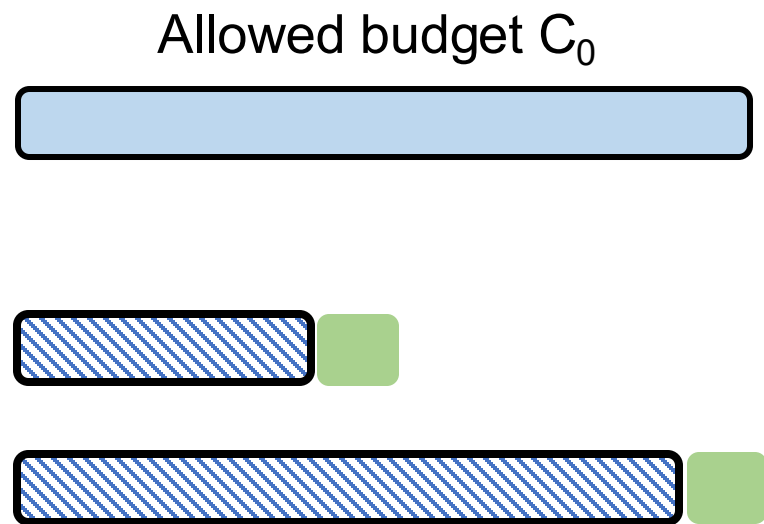
$$\max_{\pi} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{train}}} [\mathbb{E}_{\mathbf{z} \sim \pi(\cdot|\mathbf{x})} [r(\mathbf{x}, \mathbf{z})]] \quad \text{s.t.} \quad \dots$$

But this compute budget is fixed!

Can optimize this via:

- RL (like DeepSeek-R1): outcome-reward RL
- SFT / STaR: collect data, filter by correctness, maximize likelihood

Why is Outcome Reward + Fixed Budget Bad?





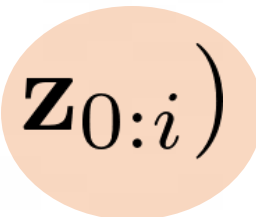
Easy problems: Both get rewarded the same way

Hard problems: Neither trace gets rewarded

Formulation: “Budget-Agnostic” LLMs

Key idea: *Incentivize the LLM to make progress regardless of the compute budget*

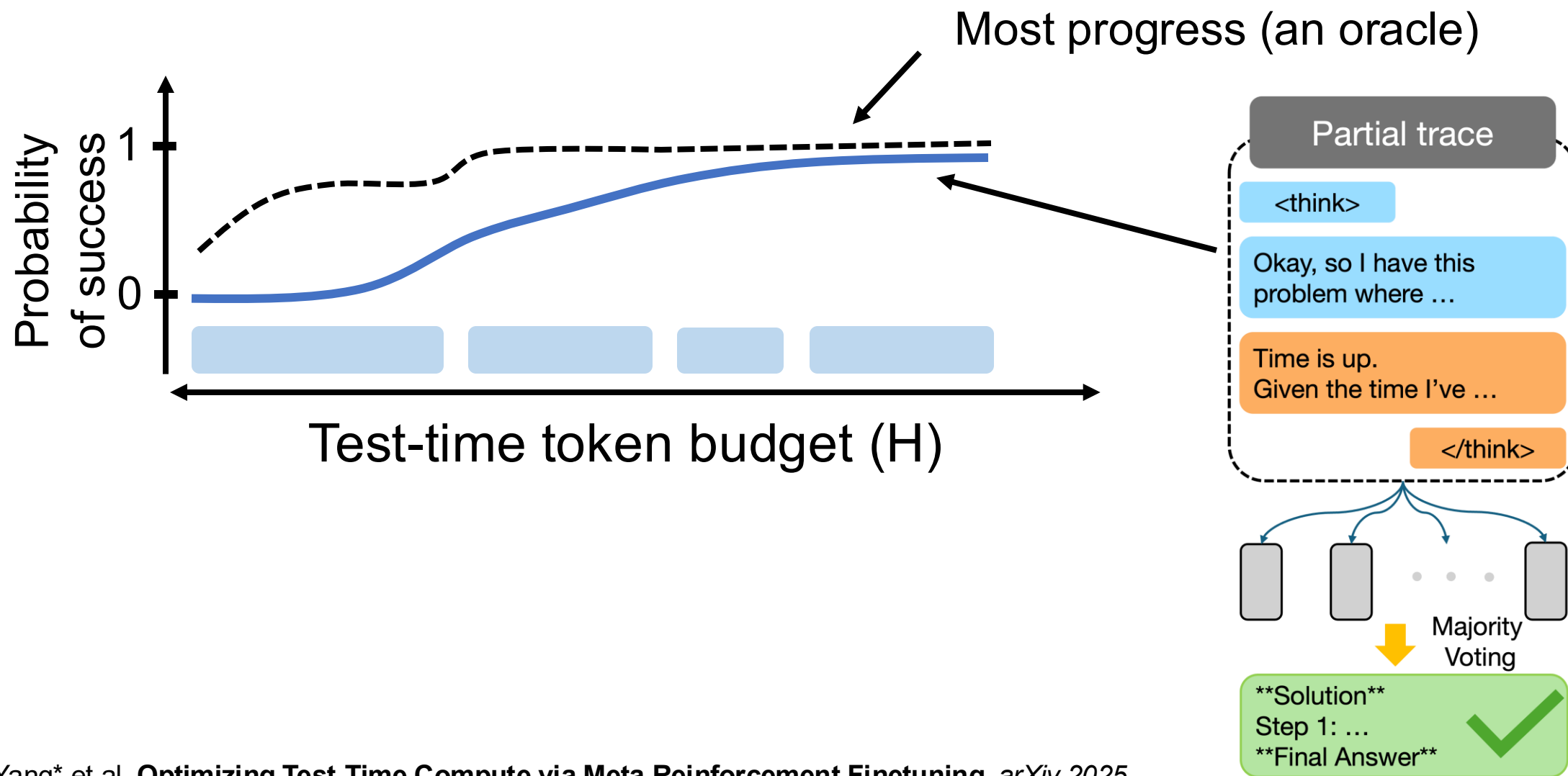
$$\max_{\pi} \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{train}}} \left[\mathbb{E}_{\mathbf{z} \sim \pi(\cdot | \mathbf{x})} [r(\mathbf{x}, \mathbf{z})] \right] \text{ s.t. } \dots$$


$$r(\mathbf{x}, \mathbf{z}) + \sum_{i=1}^H r_{\text{prg}}(\mathbf{x}, \mathbf{z}_{0:i})$$


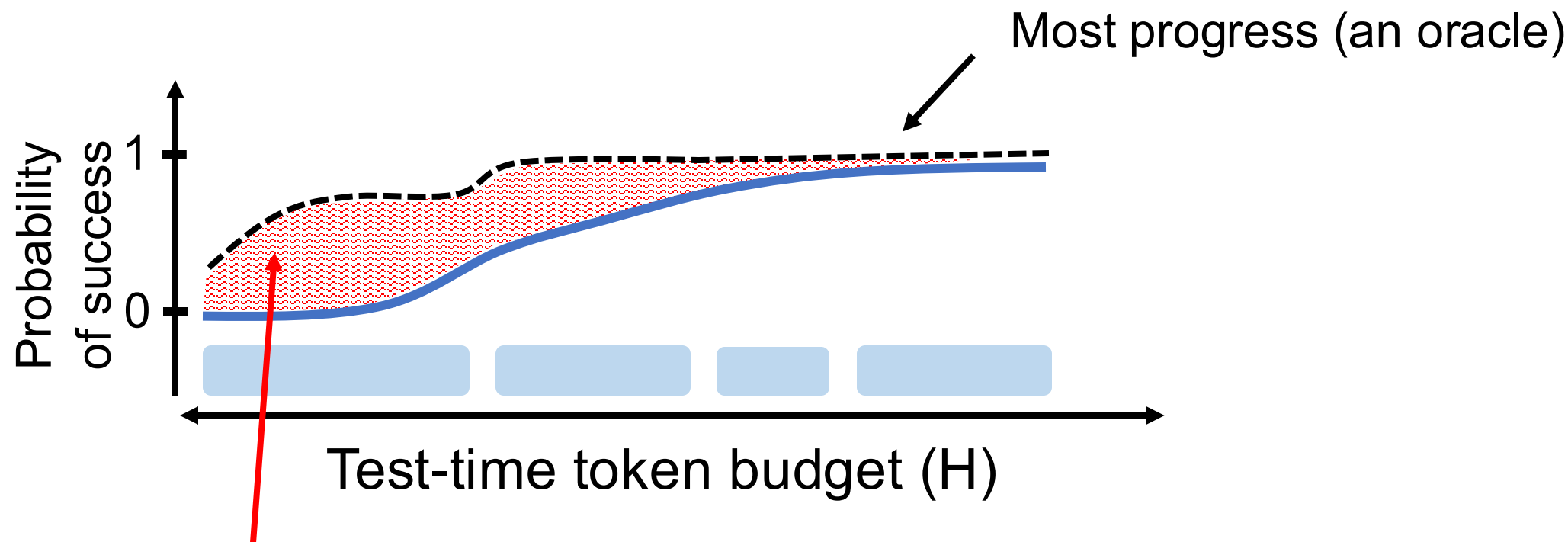
Some segment of
the entire trace

Some dense reward to incentivize progress

Idea: What is Good Progress on New Problems?

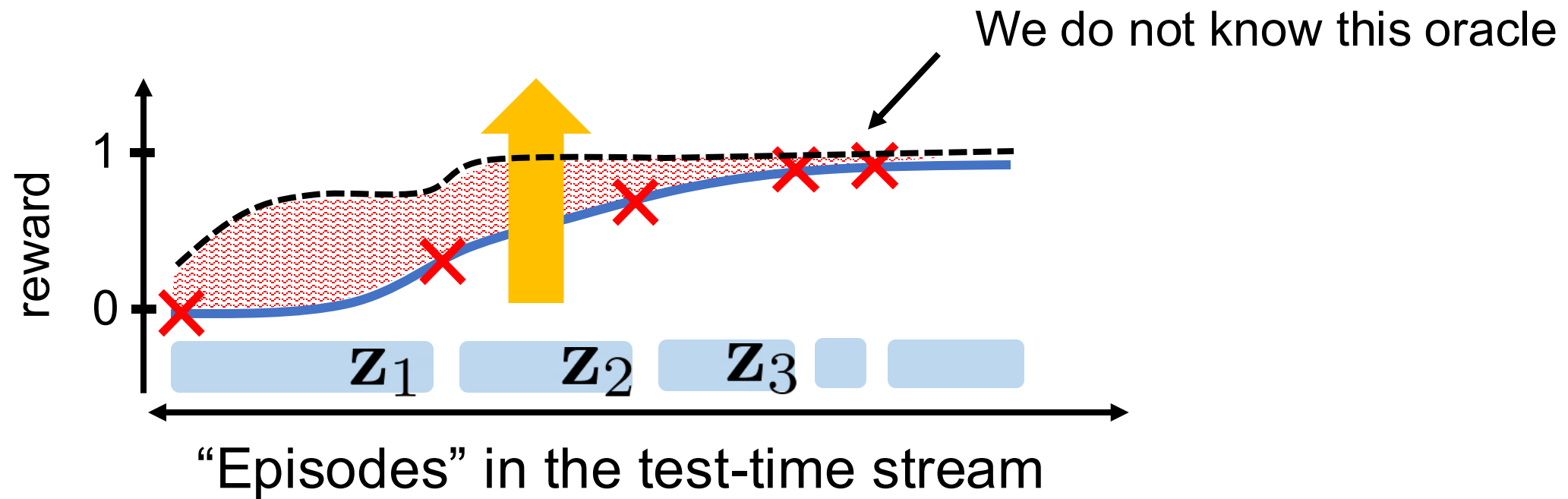


Idea: What is Good Progress on New Problems?



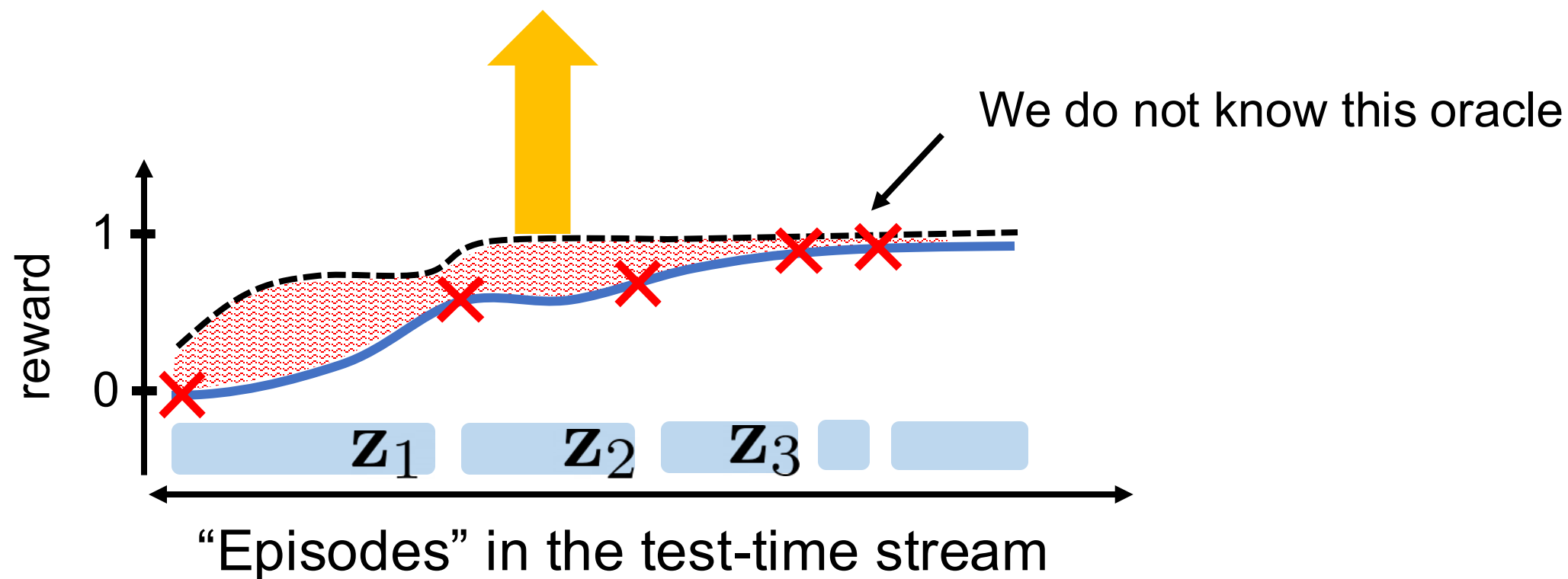
We want this area to be as low as possible!
(i.e., a notion of cumulative regret)

Inducing a Good Curve That Makes Progress



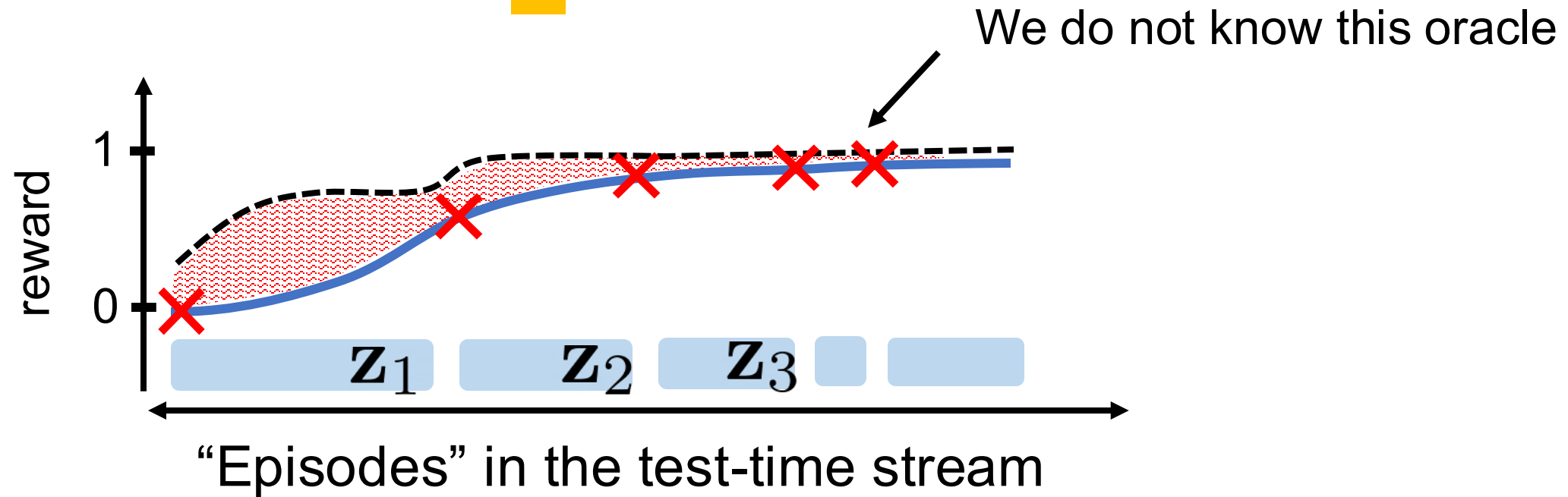
Key idea: Can still push up the performance after every episode!

Inducing a Good Curve That Makes Progress



Key idea: Can still push up the performance after every episode!

Inducing a Good Curve That Makes Progress



Key idea: Can still push up the performance after every episode!

Concrete Idea: Progress Reward Design

$$\mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\text{train}}, \mathbf{z} \sim \pi(\cdot | \mathbf{x})} \left[\sum_j \nabla_{\pi} \log \pi(\mathbf{z}_j | \mathbf{x}, \mathbf{z}_{0:j-1}) \cdot (r(\mathbf{x}, \mathbf{z}) + \alpha \cdot r_{\text{prg}}(\mathbf{x}, \mathbf{z}_{0:j})) \right]$$

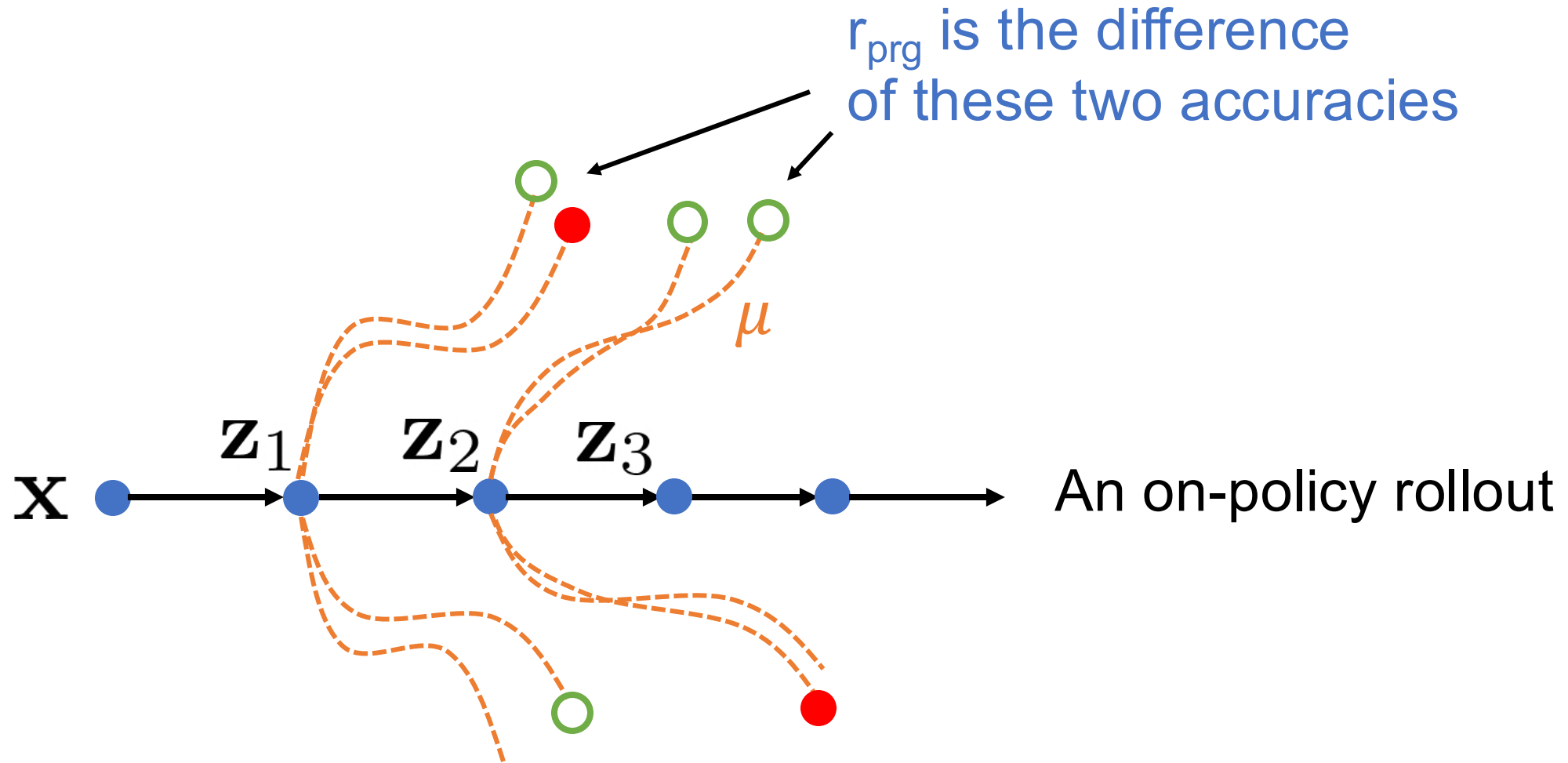
So far, we were using accuracy of the model
that aims to write out the solution given the episodes so far

Progress reward

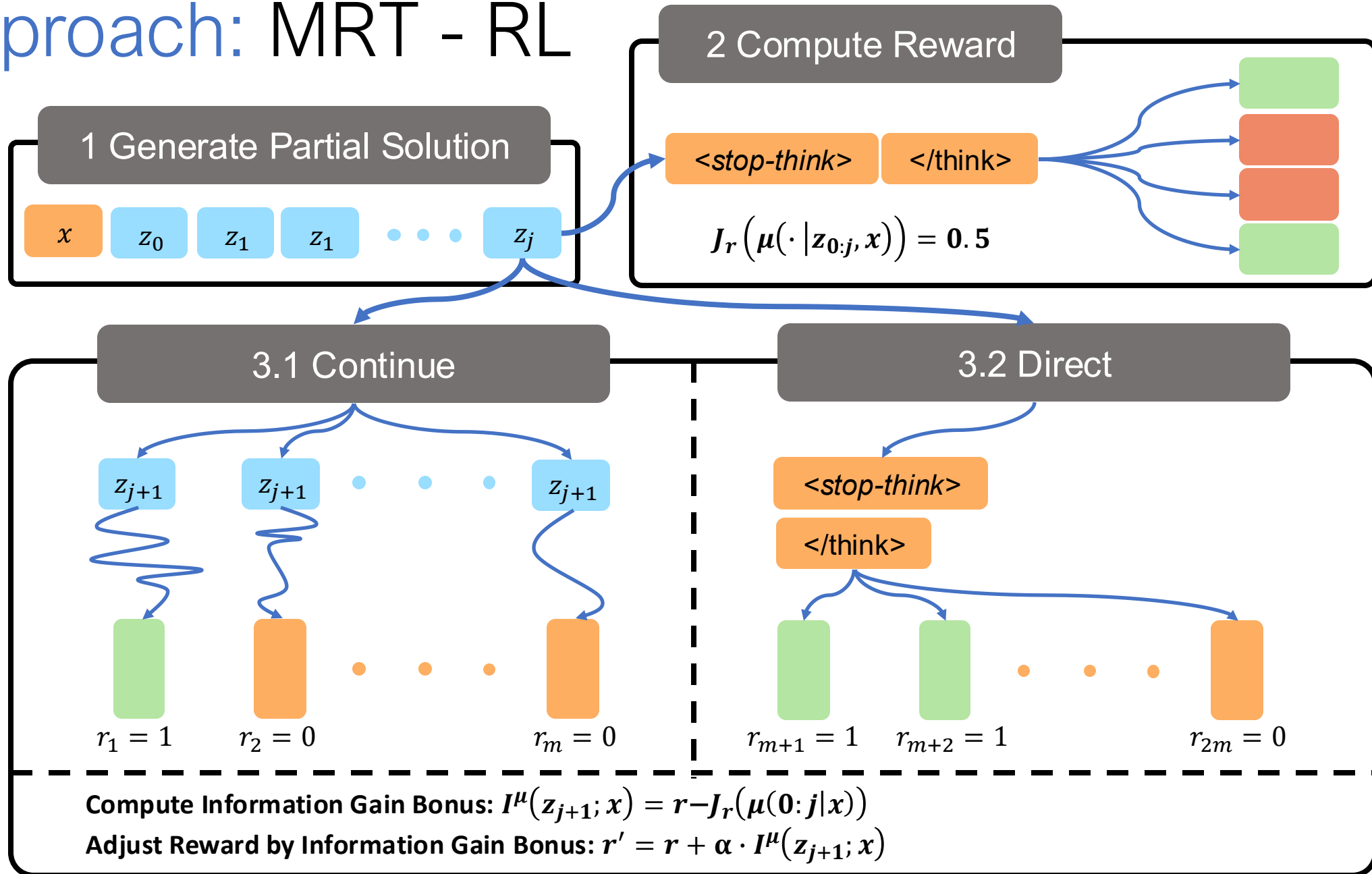
A “prover” policy that guesses the best answer

$$r_{\text{prg}}(\mathbf{x}, \mathbf{z}_{0:j}) = J_r(\mu(\cdot | \mathbf{x}, \mathbf{z}_{0:j})) - J_r(\mu(\cdot | \mathbf{x}, \mathbf{z}_{0:j-1}))$$

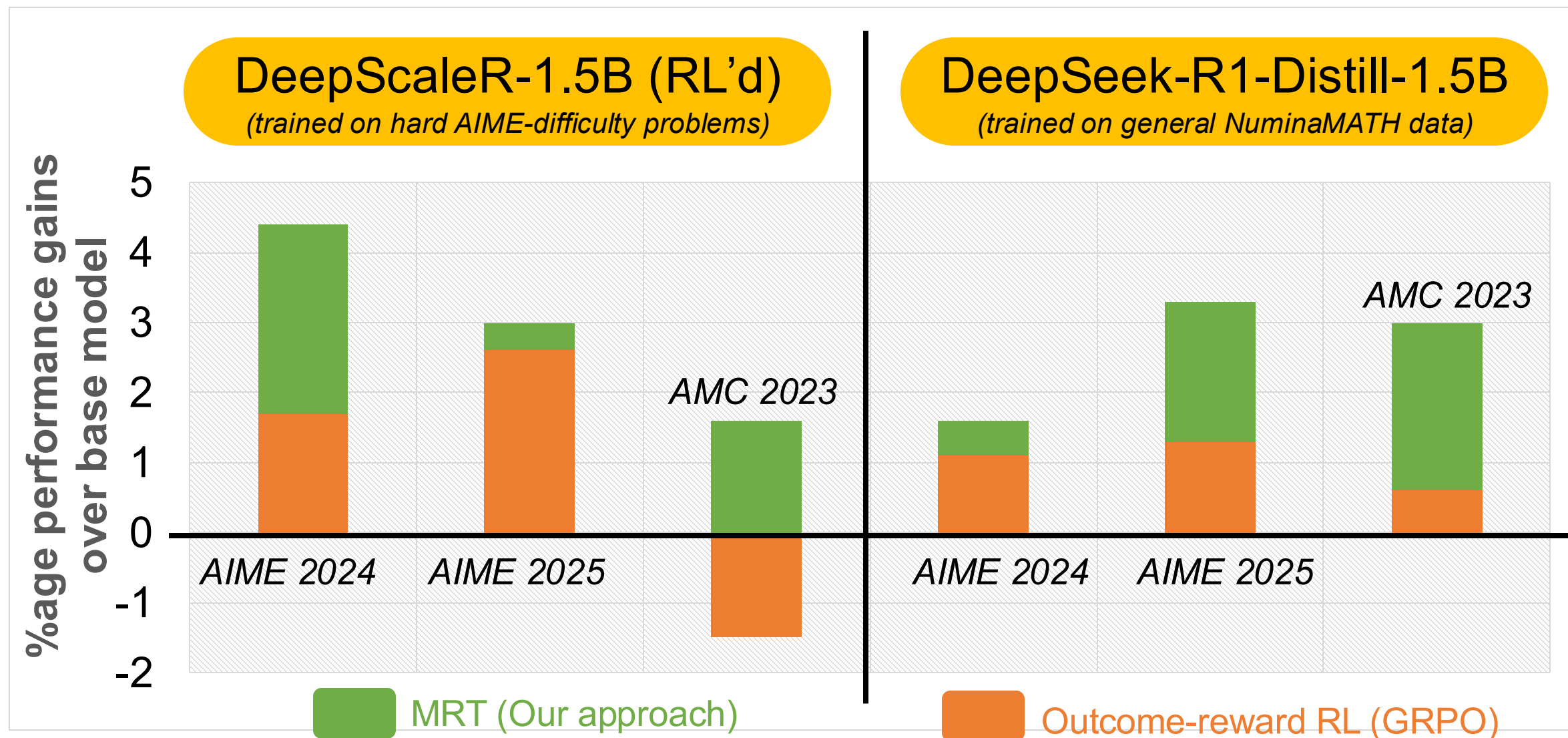
Concrete Idea: Progress Reward Design



Approach: MRT - RL



(Subset of) Results: Our Approach (MRT)



Open Questions: Dense Rewards

Computational cost

- Estimating dense rewards requires rollouts, which are costly.
 - *Can we get more juice out of the same total FLOPs??*

See: RL on Incorrect Synthetic Data Scales the Efficiency of LLM Math Reasoning 8x.
NeurIPS 2024.

The choice of the prover policy

- The policy μ determines the progress reward.
 - *How should you choose this policy??*

See: Rewarding Progress: Scaling Automated Process Verifiers. **ICLR 2025.**

Other ways of implementing the same principle

- Length curriculum and iterative training could be one other way
 - *Many open-source implementations kinda do this!*