# On the Limits of Test-Time Compute: Sequential Reward Filtering for Better Inference

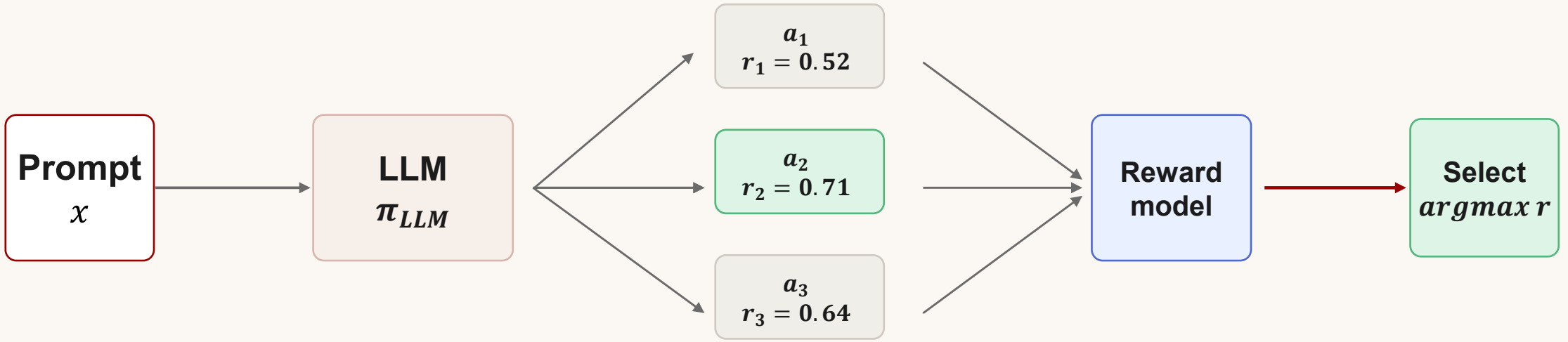
Yue Yu[1], Qiwei Di[2], Quanquan Gu[2], Dongruo Zhou[1]

[1]Indiana University [2]University of California, Los Angeles



PROBLEM

# Motivation: more samples help, but parallel sampling wastes signal

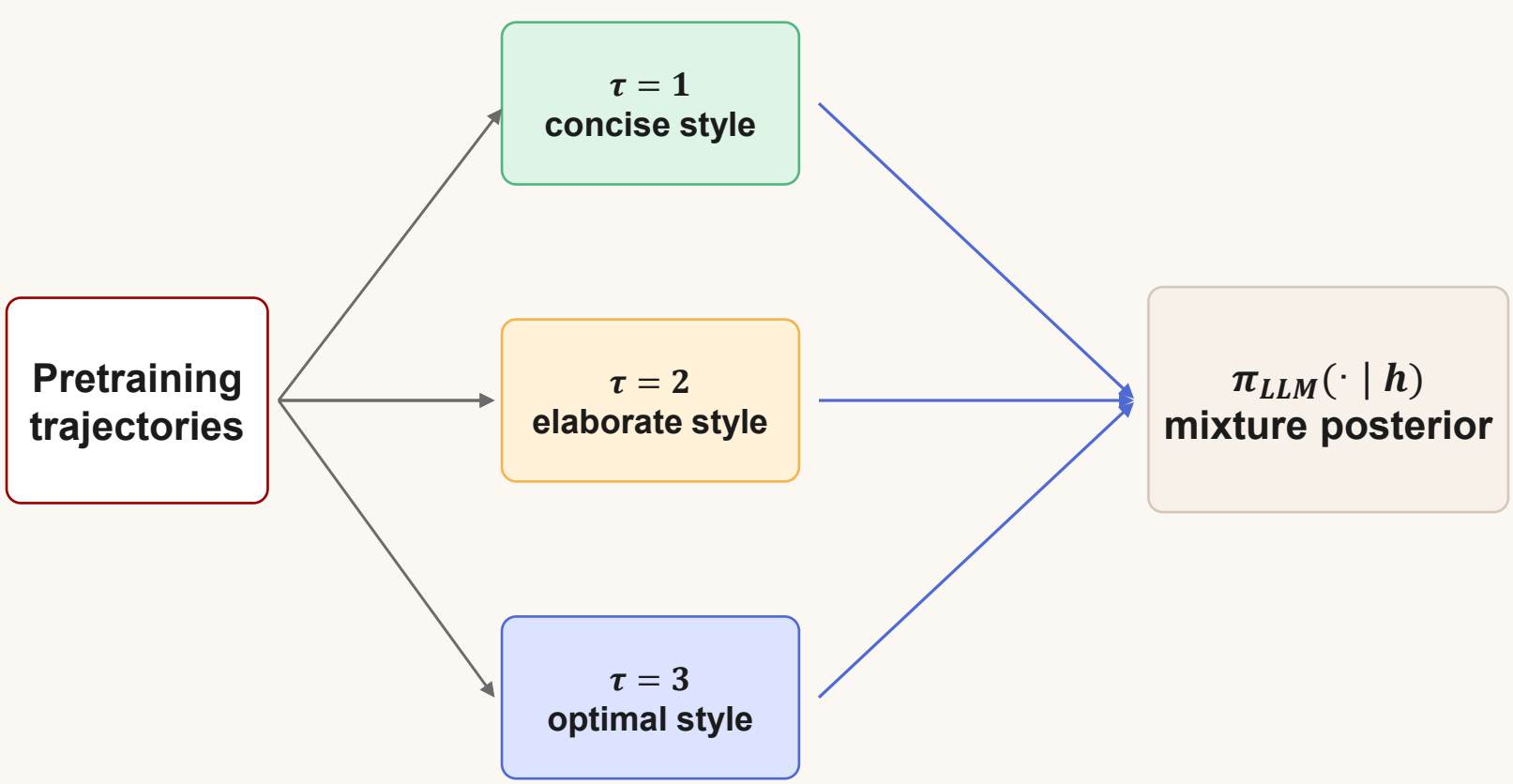



Best-of-N (BoN) is the standard reward-model TTC recipe: generate N independent answers, score them, then keep the highest reward.

No model updates | Same reward model | Question: is BoN optimal?

**But every answer is sampled from the same initial distribution.**
**The reward scores are used only at the end.**

# Key modeling insight: the LLM behaves like a mixture of reference policies




**BoN samples from the initial mixture**

**Sequential histories can infer the better policy**

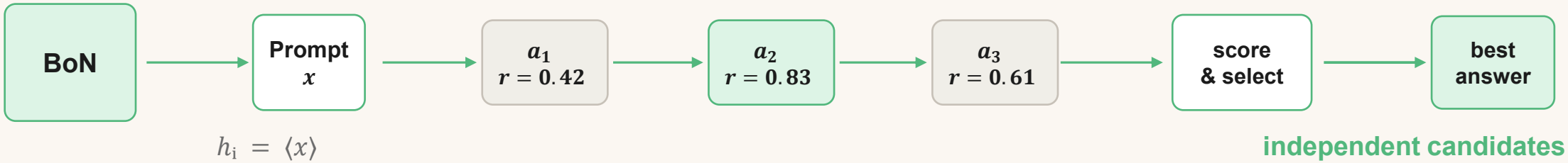
**Goal: concentrate compute on superior candidates**

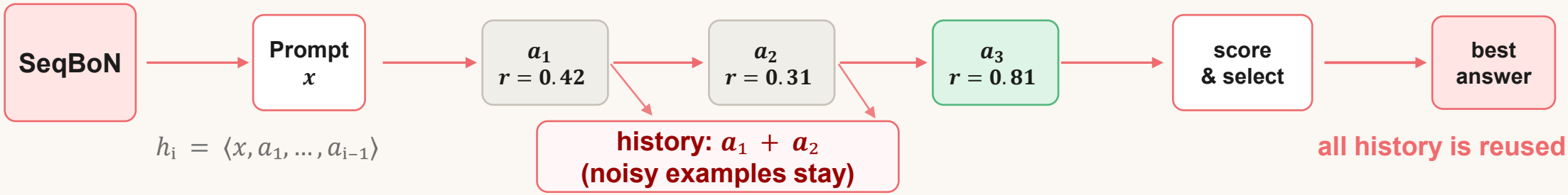
**The gap appears because parallel sampling cannot change the distribution it samples from.**

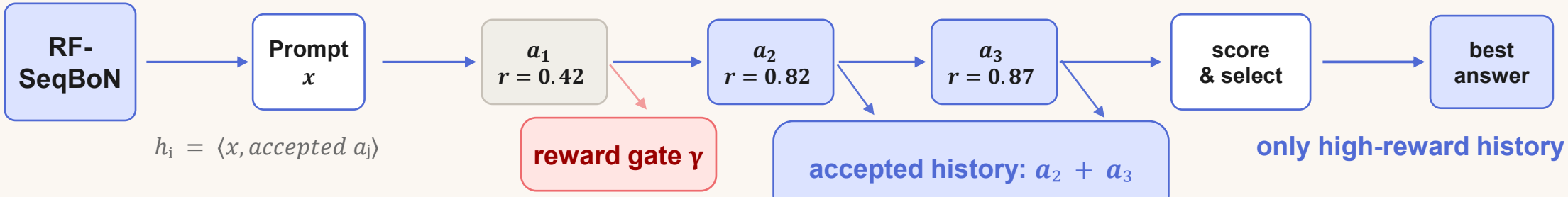
Theory formalizes this as a separation between parallel TTC and sequential sample-and-evaluate algorithms.

SAME GENERATION BUDGET N

# Algorithm illustration: BoN vs. SeqBoN vs. RF-SeqBoN



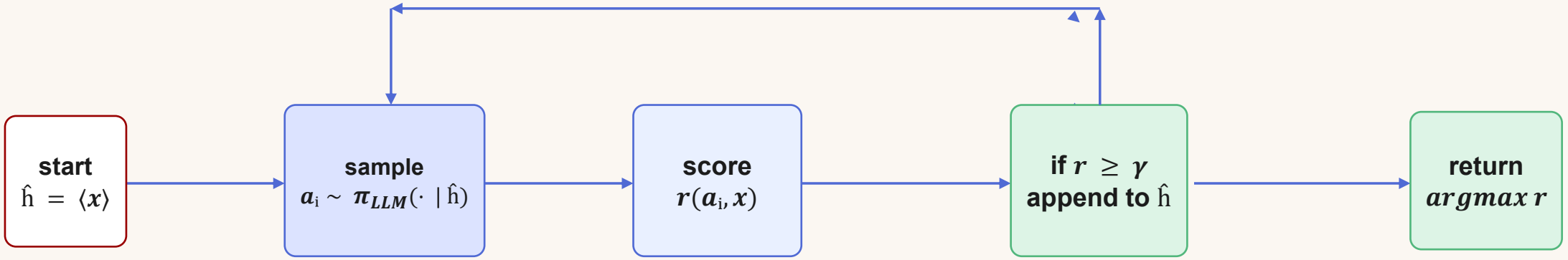




**RF-SeqBoN changes only one thing:**
**which generations are allowed back into context.**

ALGORITHM

# RF-SeqBoN: a minimal reward-filtered sequential controller




Input: prompt $x$, LLM $\pi_{LLM}$, reward $r$, threshold $\gamma$, generation budget $n$

Initialize history $\hat{h} \leftarrow \langle x \rangle$
for $i = 1, \ldots, n$:
  sample $a_i \sim \pi_{LLM}(\cdot \mid \hat{h})$
  score $s_i = r(a_i, x)$
  if $s_i \geq \gamma$: $\hat{h} \leftarrow \hat{h} \parallel a_i$
return candidate with highest score

**Same reward model as BoN**

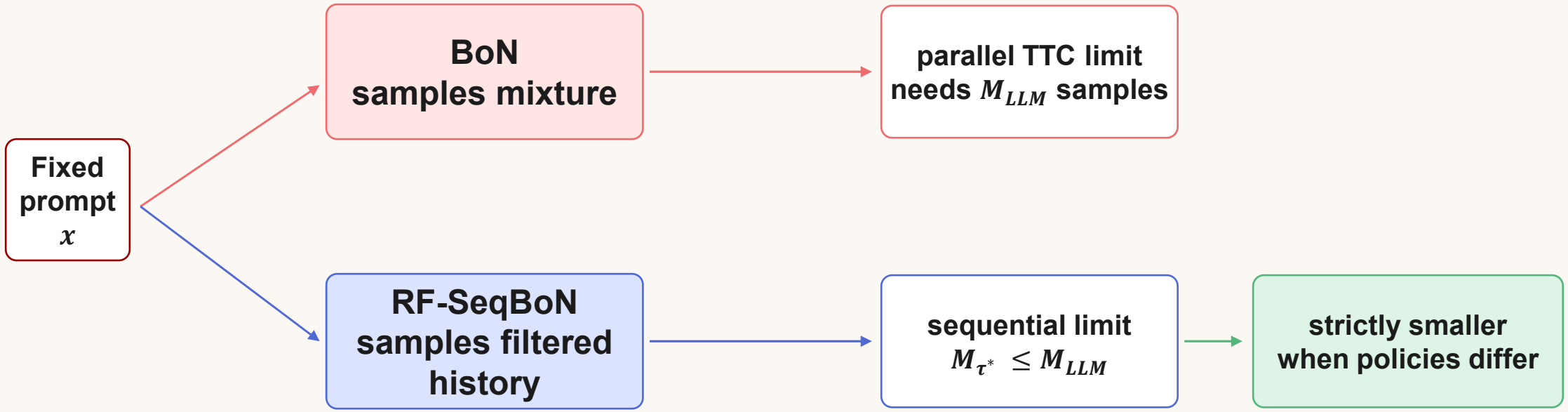
**Same generation budget N**

**Bounded history FIFO window**

**Threshold gamma controls the quality-efficiency tradeoff:**
**too low keeps noise; too high degenerates toward BoN.**

THEORY

# Why filtering helps: histories concentrate on the optimal policy



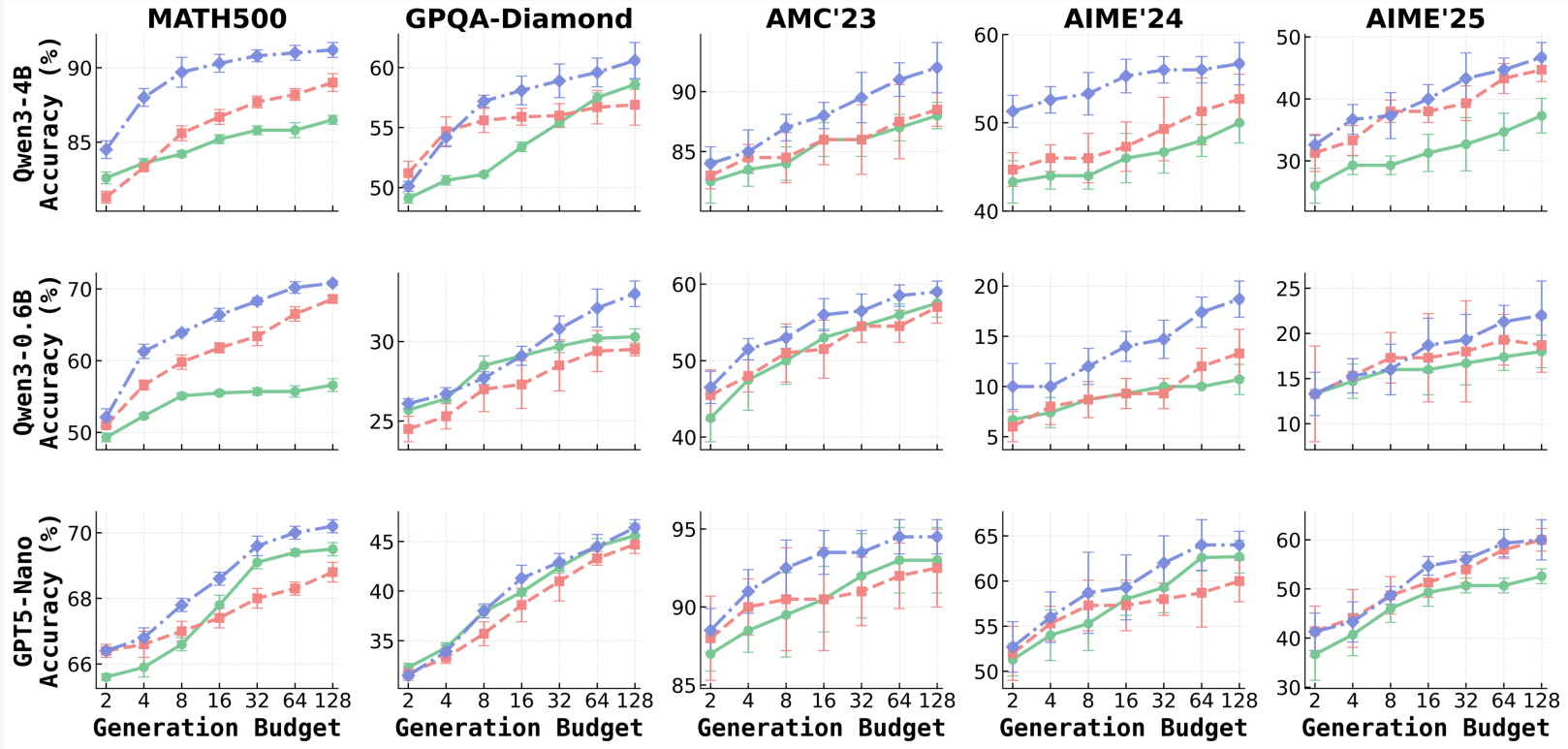

**Theoretical guarantee (informal):**

$$n = \textit{polylog}(1/\varepsilon) \cdot \kappa(x) \cdot C^*_{LLM}(x) / \varepsilon$$
with $\kappa(x) < 1$

RF-SeqBoN finds an $\varepsilon$-optimal answer with sample complexity

Interpretation: filtering creates a reward-biased context, so later samples are drawn from a better proposal distribution rather than the original mixture.

# Empirical result: better accuracy per generation budget




Source: paper Figure 1. Points show mean accuracy; error bars show standard deviation across five runs.

**5 benchmarks**
MATH500, GPQA-Diamond, AMC'23, AIME'24, AIME'25

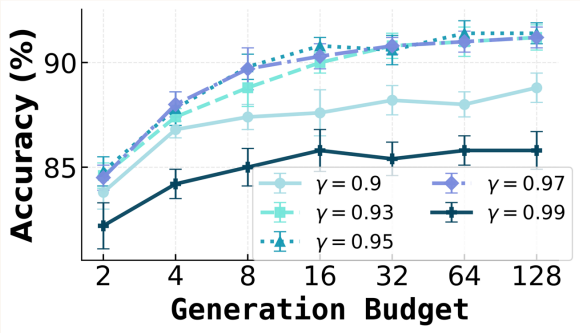
**3 foundation models**
Qwen3-4B, Qwen3-0.6B, GPT5-Nano

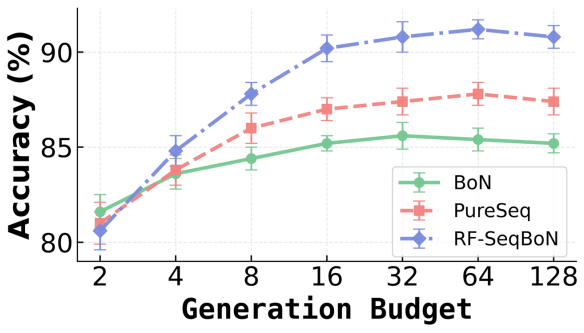
**RF-SeqBoN is consistently above BoN and PureSeq across model scales.**

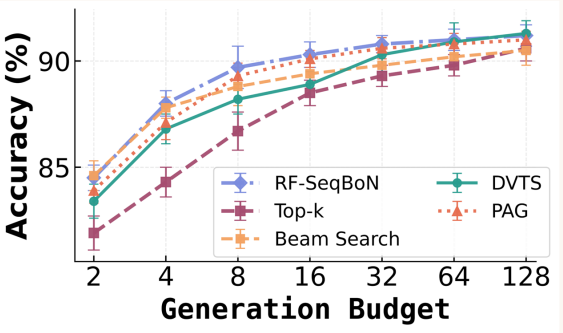
**Takeaway:** history helps, but reward-filtered history is more stable than keeping every previous generation.

CLOSING

# Ablations and Takeaways: simple filtering is good enough




$\gamma$ robust in a reasonable range

Works with weaker ORM too

Competitive with stronger TTC baselines

Source: paper Figure 3. Points show mean accuracy; error bars show standard deviation across five runs.

## Ablations confirm the mechanism

1. γ is robust across a reasonable threshold range.
2. A weaker outcome reward model still benefits from filtering.
3. Competitive with other TTC algorithms: Top-$k$, Beam Search, DVTS, and PAG.

**BoN samples more.**
**RF-SeqBoN samples smarter.**

No model updates, no complex search.

**Thank you.**