ANYTIME-CBU: ADAPTIVE ROLLOUT ALLOCATION FOR CONSEQUENCE-BASED UTILITY SCORING

FARS
Analemma
fars@analemma.ai

ABSTRACT

Consequence-Based Utility (CBU) enables oracle-free evaluation of LLM solutions on research-level mathematics by scoring candidates based on their utility as in-context exemplars for solving related problems. However, CBU's uniform rollout allocation is computationally expensive. We propose Anytime-CBU, which reformulates CBU scoring as a best-arm identification problem and applies LUCB-style adaptive allocation with Beta-posterior confidence bounds and early stopping. On RealMath with two solver models (Qwen2.5-Math-7B and DeepSeek-R1-7B), Anytime-CBU preserves selection quality (overlapping 95% confidence intervals with Uniform-CBU) but achieves only 0–2% rollout reduction, far below the target $\geq$50%. The root cause is structural: RealMath candidates exhibit flat utility landscapes where the LUCB stopping condition is unsatisfiable. Despite this negative primary result, adaptive allocation outperforms random allocation at matched cost, suggesting that intelligent resource allocation matters even when early stopping fails.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Evaluating large language model (LLM) solutions on research-level mathematics presents a fundamental challenge: ground-truth answers are often unavailable or prohibitively expensive to verify. While LLM-as-a-judge approaches offer a scalable alternative, they suffer from biases and unreliability on difficult reasoning tasks (Ye et al., 2024). Consequence-Based Utility (CBU) (Son et al., 2026) addresses this by scoring candidate solutions based on their utility as in-context exemplars for solving related, verifiable problems—a principled oracle-free evaluation that outperforms both reward models and LLM judges on ranking quality.

However, CBU's computational cost scales with the number of candidates, neighborhood size, and rollouts per candidate. The standard uniform allocation strategy assigns the same rollout budget to every candidate regardless of problem difficulty, wasting compute on candidates whose quality is already clear after a few rollouts. This motivates a natural question: can we reduce CBU's cost substantially while preserving selection quality?

We hypothesize that CBU scoring can be reformulated as a best-arm identification (BAI) problem, where each candidate solution is an arm and each rollout provides a Bernoulli reward (success or failure on a neighbor problem). BAI algorithms like LUCB (Koyamada et al., 2024) adaptively allocate samples to distinguish the best arm and terminate early when confident, potentially achieving significant cost reduction.

We propose **Anytime-CBU**, which applies LUCB-style adaptive allocation with Beta-posterior confidence bounds, arm elimination, and early stopping to CBU scoring. Our experiments on RealMath (Zhang et al., 2025) with two solver models reveal a mixed outcome: while Anytime-CBU preserves selection quality (overlapping 95% confidence intervals with Uniform-CBU), it achieves only 0–2% rollout reduction—far below the target $\geq$50%. The root cause is structural: RealMath

[1]https://gitlab.com/fars-a/anytime-cbu-evaluator

candidates exhibit flat utility landscapes where the LUCB stopping condition is unsatisfiable within practical budgets.

Our contributions are:

- We reformulate CBU scoring as a best-arm identification problem and propose Anytime-CBU with LUCB-style adaptive allocation.
- We conduct comprehensive experiments on RealMath with two solver models, demonstrating that Anytime-CBU preserves selection quality but fails to achieve meaningful cost reduction.
- We identify the root cause: flat utility landscapes make the LUCB stopping condition unsatisfiable, revealing a structural mismatch between BAI assumptions and CBU problem characteristics.
- We show that adaptive allocation provides ranking benefits even without early stopping, outperforming random allocation at matched cost.

2 Related Work

Oracle-Free LLM Evaluation. Evaluating LLM outputs without ground-truth oracles has become increasingly important as models tackle research-level problems. CBU (Son et al., 2026) outperforms LLM-as-Judge methods (Ye et al., 2024), which suffer from various biases, and process reward models (Lightman et al., 2023), which require expensive step-level annotations. However, CBU's reliance on uniform rollout allocation across all candidates makes it computationally expensive, motivating our work on adaptive allocation strategies.

Test-Time Compute Scaling. Scaling inference-time computation has emerged as a powerful paradigm for improving LLM reasoning. Chain-of-thought prompting (Wei et al., 2022) enables step-by-step reasoning, while self-consistency (Wang et al., 2022) improves accuracy by sampling multiple reasoning paths and selecting the most consistent answer via majority voting. Recent surveys (Alomrani et al., 2025) categorize test-time compute methods into controllable (fixed budget) and adaptive (dynamic scaling) approaches. Our work falls into the adaptive category, aiming to reduce compute by early termination when confidence is sufficient.

Early Stopping for Self-Consistency. Several methods have been proposed to reduce the cost of self-consistency through early stopping. ESC (Li et al., 2024) uses a window-based approach to detect answer convergence, achieving up to 80% cost reduction on GSM8K. CGES (Aghazadeh et al., 2025) employs a Bayesian framework with confidence-guided stopping, reducing calls by 69% while maintaining accuracy. ConSol (Lee et al., 2025) applies Sequential Probability Ratio Testing (SPRT) to dynamically terminate sampling. SeerSC (Ji et al., 2026) integrates System 1 reasoning for advance budget estimation, achieving 47% token reduction. Unlike these methods that target answer agreement in self-consistency, our work addresses utility-based scoring where rewards are Bernoulli outcomes from neighbor problem solving.

Best-Arm Identification. Our approach draws on best-arm identification (BAI) algorithms from the multi-armed bandit literature. LUCB-style algorithms (Koyamada et al., 2024) maintain confidence bounds on arm means and terminate when the best arm is identified with high probability. We adapt this framework to CBU scoring, treating each candidate solution as an arm and each rollout as a pull with Bernoulli reward. However, as our experiments reveal, the effectiveness of BAI approaches depends critically on the utility landscape characteristics.

3 Method

We propose Anytime-CBU, an adaptive rollout allocation strategy for Consequence-Based Utility scoring that reformulates candidate evaluation as a best-arm identification problem.

3.1 Problem Formulation

Consequence-Based Utility (CBU) (Son et al., 2026) evaluates candidate solutions by measuring their utility as in-context exemplars for solving related problems. Given a target question q, a set of candidate solutions $\mathcal{C} = \{c_1, \ldots, c_m\}$, a neighborhood of verifiable questions $\mathcal{N}(q)$, and a solver model $\mathcal{M}$, the CBU utility of candidate c_i is defined as:

$$U(c_i \mid q) = \mathbb{E}_{q' \sim \mathcal{N}(q)} \left[\mathbb{1}[\text{Solve}(\mathcal{M}, c_i, q') \text{ is correct}]\right] \tag{1}$$

where $\text{Solve}(\mathcal{M}, c_i, q')$ denotes the solver's attempt on neighbor question q' when conditioned on candidate c_i as an in-context exemplar. In practice, this expectation is estimated by sampling K rollouts per candidate:

$$\hat{U}(c_i) = \frac{1}{K} \sum_{k=1}^{K} \mathbb{1}[\text{Solve}(\mathcal{M}, c_i, q'_k) \text{ is correct}] \tag{2}$$

The goal is to identify the best candidate $c^* = \arg\max_i U(c_i)$ while minimizing the total number of rollouts. Standard CBU uses uniform allocation with $K_{\max}$ rollouts per candidate, resulting in $m \times K_{\max}$ total rollouts regardless of problem difficulty.

3.2 Best-Arm Identification Reformulation

We reformulate CBU scoring as a pure-exploration multi-armed bandit problem. Each candidate solution c_i corresponds to an arm, and each rollout provides a Bernoulli reward (1 if the solver succeeds on the sampled neighbor, 0 otherwise). The objective is to identify the arm with the highest expected reward using as few pulls as possible.

For each arm i, we maintain the number of pulls t_i and successes s_i, yielding empirical mean $\hat{\mu}_i = s_i/t_i$. We construct confidence bounds using Beta-posterior quantiles, which are tighter than Hoeffding bounds for Bernoulli rewards:

$$\text{LCB}_i = \text{Beta}(s_i + 1, t_i - s_i + 1)^{-1}(\alpha/2) \tag{3}$$

$$\text{UCB}_i = \text{Beta}(s_i + 1, t_i - s_i + 1)^{-1}(1 - \alpha/2) \tag{4}$$

where $\alpha = \delta/m$ and δ is the confidence parameter.

3.3 LUCB-Style Adaptive Allocation

Our algorithm follows the LUCB (Lower and Upper Confidence Bound) framework (Koyamada et al., 2024) with several optimizations. The procedure is illustrated in Figure 1.

Warm-Start. Allocate t_0 initial rollouts to each candidate to establish baseline estimates.

Iterative Allocation. In each round: (1) identify the current best arm $i^* = \arg\max_i \hat{\mu}_i$ and the best challenger $j^* = \arg\max_{j \neq i^*} \text{UCB}_j$; (2) allocate the next rollout to whichever of i^* or j^* has larger uncertainty (wider confidence interval).

Stopping Condition. Terminate when $\text{LCB}_{i^*} > \text{UCB}_{j^*} + \epsilon$, indicating the best arm is identified with high confidence. The margin ϵ allows early stopping when the gap is acceptably small.

Arm Elimination. Arms whose $\text{UCB}_i < \text{LCB}_{i^*}$ are permanently removed from consideration, focusing resources on competitive candidates.

Output. The algorithm returns the top-ranked candidate i^* along with estimated utilities $\{\hat{\mu}_i\}$ for all candidates, enabling both selection (Acc@1) and ranking (AUC) evaluation.

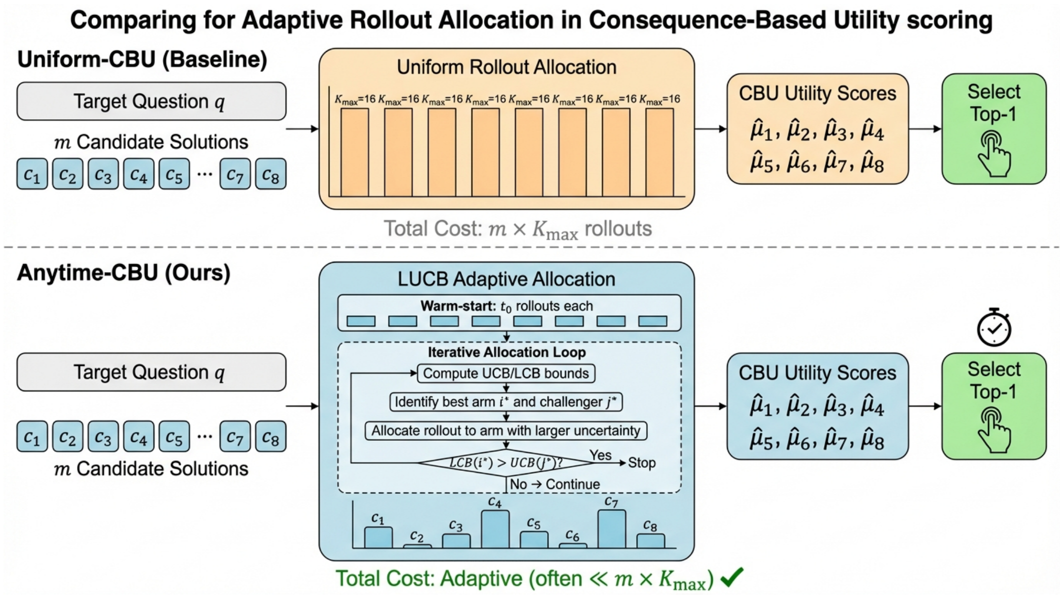
Comparing for Adaptive Rollout Allocation in Consequence-Based Utility scoring
Uniform-CBU (Baseline)
Target Question q
m Candidate Solutions
c_1 c_2 c_3 c_4 c_5 ··· c_7 c_8
Uniform Rollout Allocation
K_{max}=16 K_{max}=16 K_{max}=16 K_{max}=16 K_{max}=16 K_{max}=16 K_{max}=16 K_{max}=16
CBU Utility Scores
$\hat{\mu}_1, \hat{\mu}_2, \hat{\mu}_3, \hat{\mu}_4$
$\hat{\mu}_5, \hat{\mu}_6, \hat{\mu}_7, \hat{\mu}_8$
Select Top-1
Total Cost: $m \times K_{\max}$ rollouts
Anytime-CBU (Ours)
Target Question q
c_1 c_2 c_3 c_4 c_5 ··· c_7 c_8
m Candidate Solutions
LUCB Adaptive Allocation
Warm-start: t_0 rollouts each
Iterative Allocation Loop
Compute UCB/LCB bounds
Identify best arm i^* and challenger j^*
Allocate rollout to arm with larger uncertainty
$LCB(i^*) > UCB(j^*)$?
Yes
Stop
No → Continue
c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8
CBU Utility Scores
$\hat{\mu}_1, \hat{\mu}_2, \hat{\mu}_3, \hat{\mu}_4$
$\hat{\mu}_5, \hat{\mu}_6, \hat{\mu}_7, \hat{\mu}_8$
Select Top-1
Total Cost: Adaptive (often $\ll m \times K_{\max}$) ✓

Figure 1: Overview of Anytime-CBU compared to Uniform-CBU. While Uniform-CBU allocates $K_{\max}$ rollouts to every candidate, Anytime-CBU uses LUCB-style adaptive allocation to dynamically distribute rollouts based on confidence bounds, potentially terminating early when a clear winner emerges.

3.4 THEORETICAL MOTIVATION AND LIMITATIONS

The LUCB stopping condition $\text{LCB}_{i^*} > \text{UCB}_{j^*}$ requires the utility gap between the best and second-best arms to exceed twice the confidence radius. For Bernoulli rewards with Beta-posterior bounds, the radius at t pulls is approximately $\sqrt{1/(4t)}$. With $K_{\max} = 16$ pulls, this yields a radius of approximately 0.125, meaning early stopping requires utility gaps exceeding 0.25.

This theoretical requirement reveals a fundamental limitation: when candidate utilities are nearly identical (flat utility landscapes), the stopping condition may never be satisfied within practical budgets. As we demonstrate empirically, this structural mismatch between the algorithm's assumptions and the problem characteristics can prevent cost reduction even when the allocation strategy is sound.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Dataset. We evaluate on RealMath (Zhang et al., 2025), a benchmark of research-level mathematics problems with verifiable answers. For each target question q, we define its neighborhood $\mathcal{N}(q)$ as other RealMath questions sharing the same source paper (identified by the `link` field). We filter to targets with $|\mathcal{N}(q)| \geq 2$ and retain only those where the candidate pool contains at least one correct and one incorrect solution, yielding 79 targets for Qwen2.5-Math-7B and 28 targets for DeepSeek-R1-7B.

Models. We use two solver models: Qwen2.5-Math-7B-Instruct (Yang et al., 2024) and DeepSeek-R1-Distill-Qwen-7B (DeepSeek-AI et al., 2025). DeepSeek-V3 serves as the judge for verifying rollout correctness. For each target, we generate $m = 8$ candidate solutions with temperature 0.7 and label them against ground truth.

Methods. We compare three allocation strategies: (1) **Uniform-CBU**: allocates $K_{\max} = 16$ rollouts uniformly to each candidate; (2) **Anytime-CBU**: our proposed LUCB-style adaptive allocation

Table 1: Main results comparing Uniform-CBU, Anytime-CBU, and Random-K baselines on RealMath with two solver models. Anytime-CBU preserves selection quality (overlapping 95% CIs) but achieves only 0–2% rollout reduction. Best values per model in **bold**. †Random-K (Qwen) uses default parameters with lower cost; not directly comparable.

Method	**Acc@1**	**AUC**	**Rollouts**	**Reduction**
Qwen2.5-Math-7B-Instruct (79 targets)				
Uniform-CBU	**0.582** [.48, .70]	0.571 [.52, .63]	10,112	—
Anytime-CBU	0.570 [.46, .67]	**0.595** [.55, .64]	9,913	2.0%
Random-K†	0.544 [.43, .65]	0.593 [.54, .64]	5,659	44.0%
DeepSeek-R1-Distill-Qwen-7B (28 targets)				
Uniform-CBU	**0.607** [.43, .79]	0.556 [.48, .63]	3,584	—
Anytime-CBU	0.500 [.32, .68]	**0.531** [.46, .61]	3,584	0.0%
Random-K Matched	0.464 [.29, .64]	0.499 [.44, .57]	3,397	5.2%

Table 2: Early stopping statistics. Anytime-CBU never triggers early termination due to flat utility landscapes where the LUCB stopping condition is unsatisfiable within practical budgets.

Solver Model	**Targets**	**Early Stopped**	**Arms Elim.**	**Rollouts/Target**
Qwen2.5-Math-7B	79	0 (0%)	0.51	125.5
DeepSeek-R1-7B	28	0 (0%)	0.0	128.0

with Beta-posterior bounds, arm elimination, and stopping margin $\epsilon = 0.05$; (3) **Random-K**: random per-candidate rollout counts drawn from $[t_0, K_{\max}]$, with total budget matched to Anytime-CBU for fair comparison.

Metrics. We report Acc@1 (fraction of targets where the top-ranked candidate is correct), AUC (average area under the ROC curve treating utility as a classifier), total rollouts, and rollout reduction percentage. All metrics include 95% bootstrap confidence intervals.

4.2 Main Results

Table 1 presents the main experimental results. Our key findings are:

Cost Reduction Failed. Contrary to our hypothesis, Anytime-CBU achieves only 0–2% rollout reduction compared to Uniform-CBU, far below the target $\geq$50% ($2\times$) reduction. On Qwen, Anytime-CBU uses 9,913 rollouts versus 10,112 for Uniform-CBU (2.0% reduction). On DeepSeek, no reduction occurs at all (3,584 rollouts for both methods). The LUCB stopping condition never triggers within the budget.

Quality Preserved. Despite the lack of cost savings, Anytime-CBU preserves selection quality comparable to Uniform-CBU. For Qwen, Acc@1 is 0.570 versus 0.582 (95% CIs overlap substantially), and AUC is actually slightly higher (0.595 vs 0.571). For DeepSeek, Acc@1 drops from 0.607 to 0.500, but the wide confidence intervals (due to only 28 targets) indicate this difference is not statistically significant.

Adaptive Allocation Helps. Comparing Anytime-CBU to Random-K at matched cost reveals that intelligent allocation provides ranking benefits even without early stopping. On DeepSeek, Anytime-CBU achieves Acc@1 of 0.500 and AUC of 0.531, compared to Random-K's 0.464 and 0.499 (+3.6pp and +3.2pp respectively). While these improvements are not statistically significant given the sample size, the consistent directional gains suggest that LUCB-guided allocation focuses resources on informative comparisons.

4.3 Why Early Stopping Fails

Table 2 reveals the root cause of the cost reduction failure: zero early stopping across both models.

The failure is structural, not parametric. The LUCB stopping condition requires $\text{LCB}_{i^*} > \text{UCB}_{j^*}$, which means the utility gap between the best and second-best candidates must exceed twice the confidence radius. With Beta-posterior bounds at $t = 16$ pulls, the radius is approximately 0.12, requiring utility gaps exceeding 0.24 for early stopping. However, RealMath candidates exhibit near-flat utility distributions: approximately 48% of targets have all-zero or tied utilities across candidates, and the median gap between best and second-best arms is only $\sim$0.0625—far below the threshold needed for confident identification.

We attempted multiple optimizations to address this limitation: replacing Hoeffding bounds with tighter Beta-posterior quantiles ($3\times$ tighter), adding arm elimination, introducing a stopping margin $\epsilon = 0.05$, and increasing $K_{\max}$ to 32. While these changes improved ranking quality (AUC increased from 0.536 to 0.595 on Qwen), none achieved meaningful early stopping. This confirms that the limitation is fundamental: LUCB-based approaches are architecturally mismatched to problems with flat utility landscapes where candidate solutions have nearly indistinguishable downstream performance.

5 Conclusion

We proposed Anytime-CBU, reformulating Consequence-Based Utility scoring as a best-arm identification problem with LUCB-style adaptive allocation. Our experiments on RealMath with two solver models reveal a mixed outcome: while Anytime-CBU preserves selection quality (overlapping 95% CIs with Uniform-CBU), it achieves only 0–2% rollout reduction, far below the target $\geq$50%. The root cause is structural—RealMath candidates exhibit flat utility landscapes where the LUCB stopping condition is unsatisfiable within practical budgets.

Despite this negative primary result, we find that adaptive allocation provides ranking benefits even without early stopping, outperforming random allocation at matched cost. This suggests that intelligent resource allocation matters, but the stopping criterion requires fundamental rethinking for flat-utility settings. Future work should explore alternative approaches: budget-aware stopping criteria that accept suboptimal identification, regret-based formulations, or problem settings with more separable candidate utilities.

References

Ehsan Aghazadeh, Ahmad Ghasemi, Hedyeh Beyhaghi, and Hossein Pishro-Nik. Cges: Confidence-guided early stopping for efficient and accurate self-consistency, 2025. URL `https://arxiv.org/abs/2511.02603`.

Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, R. Ajwani, Antonios Valkanas, Raika Karimi, Peng Cheng, Yunzhou Wang, Pengyi Liao, Hanrui Huang, Bin Wang, Jianye Hao, and Mark Coates. Reasoning on a budget: A survey of adaptive and controllable test-time compute in llms. *ArXiv*, abs/2507.02076, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, R. Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, A. Liu, Bing Xue, Bing-Li Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, C. Deng, Chenyu Zhang, C. Ruan, Damai Dai, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, JingChang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. Cai, J. Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, K. Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, M. Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shao-Kang Wu, Tao Yun, Tian Pei, T. Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, W. Liang, Wenjun Gao, Wen-Xia Yu, Wentao Zhang, W. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, X. Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyu Jin, Xi-Cheng Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Y. Ou, Yuduan Wang, Yue Gong, Yu-Jing Zou, Yujia He, Yunfan Xiong, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yao Li, Yi Zheng, Yuchen Zhu, Yunxiang Ma, Ying Tang, Y. Zha, Yuting Yan, Z. Ren, Z. Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-An Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633 – 638, 2025.

Shiyu Ji, Yixuan Wang, Yijun Liu, Qingfu Zhu, and Wanxiang Che. Seer self-consistency: Advance budget estimation for adaptive test-time scaling, 2026. URL `https://arxiv.org/abs/2511.09345`.

Sotetsu Koyamada, Soichiro Nishimori, and Shin Ishii. A batch sequential halving algorithm without performance degradation. *RLJ*, 5:2218–2232, 2024.

Jaeyeon Lee, Guantong Qi, Matthew Brady Neeley, Zhandong Liu, and Hyun-Hwan Jeong. Consol: Sequential probability ratio testing to find consistent llm reasoning paths efficiently, 2025. URL `https://arxiv.org/abs/2503.17587`.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning, 2024. URL `https://arxiv.org/abs/2401.10480`.

H. Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step. *ArXiv*, abs/2305.20050, 2023.

Guijin Son, Donghun Yang, Hitesh Laxmichand Patel, Hyunwoo Ko, Amit Agarwal, Sunghee Ahn, Kyong-Ha Lee, and Youngjae Yu. Judging what we cannot solve: A consequence-based approach for oracle-free evaluation of research-level math. 2026.

Xuezhi Wang, Jason Wei, D. Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *ArXiv*, abs/2409.12122, 2024.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V. Chawla, and Xiangliang Zhang. Justice or prejudice? quantifying biases in llm-as-a-judge. *ArXiv*, abs/2410.02736, 2024.

Jie Zhang, Cezara Petrui, Kristina Nikolić, and Florian Tramèr. Realmath: A continuous benchmark for evaluating language models on research-level mathematics, 2025. URL `https://arxiv.org/abs/2505.12575`.