Limits of PRM-Guided Tree Search for Mathematical Reasoning with LLMs

Tristan Cinquin*
University of Tübingen
tristan.cinquin@uni-tuebingen.de

Geoff Pleiss
University of British Columbia & Vector Institute
geoff.pleiss@stat.ubc.ca

Agustinus Kristiadi
Western University & Vector Institute
akristi@uwo.ca

Abstract

While chain-of-thought prompting with Best-of-N (BoN) selection has become popular for mathematical reasoning in large language models (LLMs), its linear structure fails to capture the branching and exploratory nature of complex problem-solving. In this work, we propose an adaptive algorithm to maximize process reward model (PRM) scores over the intractable action space, and investigate whether PRM-guided tree search can improve mathematical reasoning by exploring multiple partial solution paths. Across 23 diverse mathematical problems using Qwen2.5-Math-7B-Instruct with its associated PRM as a case study, we find that: (1) PRM-guided tree search shows no statistically significant improvements over BoN despite higher costs, (2) Monte Carlo tree search and beam search outperform other PRM-guided tree search methods, (3) PRMs poorly approximate state values and their reliability degrades with reasoning depth, and (4) PRMs generalize poorly out of distribution. This underperformance stems from tree search's greater reliance on unreliable PRM scores, suggesting different reward modeling is necessary before tree search can effectively enhance mathematical reasoning in LLMs.

1 Introduction

Mathematical reasoning involves understanding complex problems, decomposing them into manageable steps, and revisiting intermediate results until reaching a sound solution. Large language models (LLMs) have shown remarkable capabilities in solving mathematical problems by breaking solutions into reasoning steps through *chain-of-thought* (CoT) prompting [1, 2]. When combined with process reward models (PRMs) that evaluate individual reasoning steps, Best-of-N (BoN) identifies the most promising CoT from multiple candidates and has become widely adopted [3–6]. However, CoT's linear structure fails to capture the branching nature of mathematical reasoning, where multiple strategies are considered, partial arguments explored, and errors necessitate backtracking [7, 8]. Moreover, restricting PRM evaluation to complete CoTs misses opportunities for dynamic guidance.

The *tree-of-thought* (ToT) framework [9] addresses these limitations by exploring multiple partial reasoning paths and enabling revisions using a reward model to assess the correctness of intermediate solutions. Yet applying ToT with PRMs presents challenges: reasoning trees exhibit intractable branching factors and depth, while PRMs may fail to accurately evaluate intermediate steps [3].

This work proposes an adaptive algorithm to maximize PRM scores over the intractable action space and empirically investigates whether PRM-guided tree search can improve mathematical

*Work done while interning at the Vector Institute.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MATH-AI.

reasoning in LLMs. We evaluate tree search algorithms under varying PRM quality assumptions against BoN across 23 diverse mathematical problems, using Qwen2.5-Math-7B-Instruct and its associated PRM as our case study. Key findings reveal that: (1) PRM-guided tree search fails to outperform BoN despite higher costs; (2) Monte Carlo tree search and beam search outperform other PRM-guided tree search methods; (3) PRMs poorly approximate state values and reliability degrades with reasoning depth, suggesting credit assignment issues; and (4) PRMs exhibit limited out-of-distribution generalization. This underperformance stems from tree search's greater reliance on unreliable PRM scores to guide search, whereas BoN evaluates only complete CoTs. These results highlight the limitations of PRM-guided tree search and BoN, indicating that different reward models may be required for mathematical reasoning.

Limitations. This work demonstrates that PRM-guided tree search fails to outperform BoN due to PRM limitations: poor reasoning step value estimation, degraded reliability with reasoning depth, and limited out-of-distribution generalization. While our study focuses on a single PRM-model pair, we expect this underperformance to generalize to PRMs exhibiting similar pathologies.

2 Background

2.1 Mathematical reasoning as tree search

We formulate mathematical reasoning as search in a *tree*-structured Markov decision process $\mathcal{M} = (\mathbb{S}, \mathbb{A}, r, t)$ where actions $a \in \mathbb{A}$ are reasoning steps (text ending with an end-of-reasoning-step token), states $s \in \mathbb{S} \coloneqq \cup_{i=0}^{T} \mathbb{A}^i$ are partial reasoning sequences and transitions are deterministic $t(s, a, s') = \mathbb{1}[s' = s \oplus a]$ (here $s \oplus a$ denotes string concatenation). The root state is the prompt p, and $\mathbb{T}$ denotes terminal states containing predictions in the '`\boxed{x}`' format. The reward function assigns $r(s) = 1$ if $s \in \mathbb{T}$ contains the correct solution with valid intermediate reasoning steps, and $r(s) = 0$ otherwise. The value function $v(s) = r(s) + \max_{a \in \mathbb{A}} v(s \oplus a)$ indicates whether any continuation from state s leads to a state with reward 1. A LLM $\pi_{\boldsymbol{\theta}}$ defines a policy $a \sim \pi_{\boldsymbol{\theta}}(\cdot \mid s)$, while a process reward model $f_{\boldsymbol{\phi}}(s)$ estimates $p(v(s) = 1 \mid s)$. Our goal is finding s with $r(s) = 1$ using the LLM and the PRM. Since rewards are unavailable during search, the PRM's approximation quality determines the appropriate search strategy. We distinguish three scenarios:

Scenario 1: PRM as Value Function. If the PRM correctly ranks actions by their true $p(v(s') = 1 \mid s' = s \oplus a)$, optimal search recursively selects $a^* = \arg\max_{a \in \mathbb{A}} f_{\boldsymbol{\phi}}(s \oplus a)$. If actions $\mathbb{A}$ were practically enumerable, a greedy tree search algorithm would be optimal. However, enumerating $a \in \mathbb{A}$ is intractable and we propose an algorithm in Section 3 to adaptively resample actions $a \sim \pi_{\boldsymbol{\theta}}(\cdot \mid s)$ until we are confident that we have attained $\max_{a \in \mathbb{A}} f_{\boldsymbol{\phi}}(s \oplus a)$.

Scenario 2: PRM as Terminal Signal Only. In this "worst-case we can still work with" scenario, the PRM suffers from poor credit assignment, failing to properly estimate $p(v(s) = 1 \mid s)$ for intermediate states, yet PRM scores at terminal states still correlate with reward r. The optimal tree search algorithm returns $s^* \in \arg\max_{s \in \mathbb{T}} f_{\boldsymbol{\phi}}(s)$ among terminal states.

Scenario 3: PRM as Noisy Intermediate Signal. The PRM provides useful but unreliable guidance for intermediate states. For example, it may undervalue (i.e., $f_{\boldsymbol{\phi}}(s) < p(v(s) = 1 \mid s)$) optimal intermediate states leading to high-scoring terminal states, and overvalue (i.e., $f_{\boldsymbol{\phi}}(s) > p(v(s) = 1 \mid s)$) suboptimal intermediate states leading to poor terminal states. While maximizing PRM scores generally helps to reach valuable terminal nodes, the appropriate search objective remains unclear.

2.2 Tree search baselines

Best-of-N samples N chains-of-thought from the LLM policy (i.e., separate root-to-terminal paths with no shared intermediate state) and selects the CoT with the highest aggregated PRM score [3, 5]. Given a prompt p, we sample CoT $c_i = (p, a_1^{(i)}, \ldots, a_T^{(i)})$ as $a_{j+1}^{(i)} \sim \pi_{\boldsymbol{\theta}}(\cdot \mid p \oplus (\oplus_{k=1}^{j} a_k^{(i)}))$ for $i = 1, \ldots, N$ and return $\arg\max_{i \in \{1,\ldots,N\}} \Psi(\{f_{\boldsymbol{\phi}}(a_j^{(i)})\}_{j=1}^{T})$ where Ψ aggregates PRM scores.

Greedy best-first search (GBFS) expands the frontier state $s \in \mathbb{F}$ with highest heuristic value $h(s)$ at each step, where the frontier $\mathbb{F}$ contains all unexpanded states in the current search tree. Starting from the root, we repeatedly expand $s = \arg\max_{s \in \mathbb{F}} h(s)$ by sampling K actions from the LLM until reaching a terminal state. We use $h(s) = f_{\boldsymbol{\phi}}(s)$ and depth-aware $h(s) = f_{\boldsymbol{\phi}}(s) \cdot (M - d(s))$ to favor deeper states, where M is maximum depth and $d(s)$ is the depth of state s.

Beam search maintains the top-N states with highest (cumulative) PRM score in a beam $\mathbb{B}_t$. At each step t, we sample actions $a_j^i \sim \pi_{\boldsymbol{\theta}}(\cdot \mid s_j)$ $i = 1, \ldots, K$ for each state in the current beam $s_j \in \mathbb{B}_t$, score states $\cup_{j=1}^{N}\{s_j \oplus a_j^i\}_{i=1}^{K}$ with the PRM, and keep the N highest-scoring states for the next beam $\mathbb{B}_{t+1}$. For $N = 1$, this reduces to greedy search.

Monte Carlo tree search (MCTS) builds a search tree in four phases: (1) **Select**: traverse the search tree from root to leaf selecting high-value states and balancing exploration/exploitation; (2) **Expand**: add children to the selected leaf by sampling actions from $\pi_{\boldsymbol{\theta}}$; (3) **Rollout**: run LLM policy from the new state to a terminal state; (4) **Backpropagate**: update visit counts and average PRM scores of terminal states along the path back to the root. Repeat until computational budget is depleted.

3 Methods

In this section, we propose an adaptive algorithm to solve the intractable optimization problem $a = \arg\max_{a \in \mathbb{A}} f_{\boldsymbol{\phi}}(s \oplus a)$ (due to the unenumerable $|\mathbb{A}|$) from Scenario 1 (see Section 2.1). We formulate this optimization problem as a stopping problem: when should we stop sampling actions from the policy and commit to the action with highest observed $f_{\boldsymbol{\phi}}(s \oplus a)$?

Maximizing over the intractable $\mathbb{A}$ using Gittin's indices. At each state s, we sample independent actions $a_i \sim \pi_{\boldsymbol{\theta}}(\cdot|s)$ and evaluate PRM scores $f_i = f_{\boldsymbol{\phi}}(s \oplus a_i)$. We must then decide whether to **stop** and commit to the current maximum *observed* PRM score $m = \max_i f_i$ (payoff m), or **sample** again at cost c to improve the current estimate m for expected payoff $\mathbb{E}_{f \sim p(f|s)}[\max(m, f)] - c$ (since we select the largest value among $f \sim p(\cdot \mid s)$ and m and incur a cost c). This is an instance of the Pandora's box problem [10], whose optimal strategy **samples** if $\mathbb{E}_{f \sim p(f|s)}[\max(m, f)] - c > m$ and otherwise **stops**. This involves computing a Gittin's index m^* satisfying $\mathbb{E}_{f \sim p(f|s)}[\max(0, f - m^*)] = c$, then sampling if $m^* > m$ and stopping otherwise. However, $p(f \mid s)$ is intractable and estimating it requires the LLM samples we are trying to acquire sparingly. This motivates a strategy based on surrogate modeling and posterior inference inspired by Xie et al. [11] which we discuss next.

Bayesian surrogate approximation. We approximate $p(f \mid s)$ using a logit-Normal surrogate model $q(f \mid s, \boldsymbol{\psi})$ with parameters $\boldsymbol{\psi}$. Specifically, we encode prior beliefs in $p(\boldsymbol{\psi})$, update the posterior $p(\boldsymbol{\psi} \mid \mathcal{D}) \propto q(\mathcal{D} \mid s, \boldsymbol{\psi}) p(\boldsymbol{\psi})$ with observed PRM scores $\mathcal{D} = \{f_i | f_i \sim p(f \mid s)\}$, and use the posterior predictive $q(f \mid s, \mathcal{D}) = \int q(f \mid s, \boldsymbol{\psi}) p(\boldsymbol{\psi} \mid \mathcal{D}) d\boldsymbol{\psi}$ to approximate $p(f \mid s)$. We then compute the Gittin's index m^* by solving $\mathbb{E}_{f \sim q(f|s,\mathcal{D})}[\max(0, f - m)] = c$ under posterior beliefs $q(f \mid s, \mathcal{D})$ rather than $p(f \mid s)$. The left-hand side is the expected improvement over m [12], a standard Bayesian optimization acquisition function [13]. The Gittin's index represents the threshold where expected improvement equals cost c, thus smaller c induces more exploration. More details in Appendix A.

4 Results

4.1 Experimental setup

LLM & PRM. We use the Qwen2.5-Math-7B-Instruct [4] LLM with the recommended prompting strategy and sampling parameters, and the Qwen2.5-Math-PRM-7B process reward model [3].

Problems & metrics. We evaluate on 22 mathematical reasoning problems from Yang et al. [4] and AIME 2025 [14]. We report mean accuracy and rank across problems with standard errors. To address concerns about high variance in LLM evaluation [15], we test statistical significance between the top-performing method and all others using Wilcoxon signed-rank tests (insignificant if $p > 0.05$).

Tree search methods. We compare Best-of-N (`BoN`) with $N = 8$ chain-of-thoughts using last, minimum, average, product, maximum and sum aggregation functions Ψ; the proportion of answers containing at least one correct prediction among $N = 8$ CoTs (`PASS@N`); majority voting among predictions of $N = 8$ CoTs (`MAJ@N`); beam search with beam size $N = 1$ expanding the state with highest PRM value from K policy samples (`Greedy@K`); beam search with beam size $N = 4$ from $K = 6$ policy samples maximizing instantaneous (V) or cumulative (CV) PRM scores (`Beam@N`); greedy best-first search with $K = 8$ policy samples (`GBFS@K`); depth-aware GBFS (`GBFS_DA@K`; see Section 2); Monte Carlo tree search with $K = 8$ policy samples (`MCTS@K`) and our proposed method from Section 3 with constant and linear cost schedules to allow more exploration early in the search when the remaining sampling budget is large (`Gittins@cost`; more details in Appendix A).

4.2 Findings

1. PRM-guided tree search methods do not outperform Best-of-N despite higher costs. Best-of-N using terminal PRM scores (`BoN_Last@8`) achieves the highest mean rank and accuracy (see Tables 1 and 4). Among Best-of-N variants, average, minimum, and product aggregations perform comparably without significant differences. MCTS and beam search also show no significant performance degradation compared to the best method. However, tree search methods incur substantially higher computational costs, generating considerably more tokens (see Generated token in Table 3). Despite this increased cost, their final solutions contain fewer reasoning steps and tokens than Best-of-N solutions (see Reasoning steps and Out Tokens in Table 3). Except for GBFS, tree search methods reach approximately as many terminal states as Best-of-N (see Terminal states in Table 3).

Table 1: **Method mean accuracy and rank with standard errors across problems.** We bold results which are not significantly worse than the best ($p > 0.05$).

Method	Accuracy (p-value)	Rank (p-value)
`PASS@8`	**79.8 ± 4.7 (N/A)**	N/A
`MAJ@8`	71.4 ± 5.1 (0.010)	4.43 ± 0.51 (0.022)
`BoN_Last@8`	**72.7 ± 5.1 (N/A)**	**3.13 ± 0.38 (N/A)**
`BoN_Avg@8`	**72.1 ± 5.0 (0.444)**	**3.22 ± 0.31 (0.787)**
`BoN_Min@8`	**72.2 ± 5.0 (0.711)**	**3.26 ± 0.32 (0.608)**
`BoN_Prod@8`	**72.0 ± 5.0 (0.408)**	**3.26 ± 0.40 (0.795)**
`BoN_Sum@8`	67.6 ± 5.3 (0.000)	7.30 ± 0.72 (0.000)
`BoN_Max@8`	68.8 ± 5.4 (0.000)	7.35 ± 0.69 (0.000)
`Greedy@6`	71.6 ± 5.0 (0.043)	5.74 ± 0.76 (0.003)
`Greedy@20`	71.2 ± 5.0 (0.039)	5.09 ± 0.55 (0.009)
`Beam@4 (V)`	**71.8 ± 4.9 (0.126)**	**3.83 ± 0.42 (0.236)**
`Beam@4 (CV)`	**71.9 ± 4.8 (0.189)**	**4.00 ± 0.49 (0.221)**
`GBFS@8`	46.0 ± 4.1 (0.000)	10.09 ± 0.71 (0.000)
`GBFS_DA@8`	48.1 ± 4.6 (0.000)	10.00 ± 0.65 (0.000)
`MCTS@8`	**71.2 ± 5.0 (0.987)**	**3.26 ± 0.69 (0.856)**
`Gittins@0.05` (ours)	70.5 ± 5.2 (0.012)	5.96 ± 0.59 (0.001)
`Gittins@linear` (ours)	71.4 ± 5.1 (0.013)	4.74 ± 0.56 (0.011)

2. MCTS and beam search perform best among PRM-guided tree search methods. `MCTS@8` achieves the highest mean rank among tree search methods, with beam search variants (`Beam@4 (V)` and `Beam@4 (CV)`) performing comparably without significant differences ($p > 0.05$, see Table 2). For mean accuracy, beam search maximizing the cumulative PRM values performs best, followed by `Beam@4 (V)`, `Greedy@6`, `Gittins@linear` and `MCTS@8` with no significant performance gaps.

Table 2: **Method mean accuracy and rank with standard errors across problems for tree search methods.** We bold results which are not significantly worse than the best ($p > 0.05$).

Method	Accuracy (p-value)	Rank (p-value)
`Greedy@6`	**71.6 ± 5.0 (0.498)**	3.78 ± 0.41 (0.021)
`Greedy@20`	71.2 ± 5.0 (0.019)	3.43 ± 0.27 (0.032)
`Beam@4 (V)`	**71.8 ± 4.9 (0.601)**	**2.52 ± 0.20 (0.232)**
`Beam@4 (CV)`	**71.9 ± 4.8 (N/A)**	**2.57 ± 0.24 (0.286)**
`GBFS@8`	46.0 ± 4.1 (0.000)	6.70 ± 0.34 (0.000)
`GBFS_DA@8`	48.1 ± 4.6 (0.000)	6.48 ± 0.30 (0.000)
`MCTS@8`	**71.2 ± 5.0 (0.332)**	**2.26 ± 0.41 (N/A)**
`Gittins@0.05` (ours)	70.5 ± 5.2 (0.019)	4.00 ± 0.35 (0.006)
`Gittins@linear` (ours)	**71.4 ± 5.1 (0.398)**	3.09 ± 0.30 (0.048)

3. The PRM poorly approximates $p(v(s) = 1 \mid s)$ and reliability degrades with reasoning depth, limiting tree search effectiveness. Methods assuming the PRM accurately estimates the value across all states (`Greedy@K` and `Gittins@cost`, Scenario 1 in Section 2.1) perform significantly worse than Best-of-N and other tree search methods (Tables 1 and 2). Increasing policy samples ($K = 6$ to $K = 20$) or adaptive sampling (`Gittins@cost`) does not improve performance. This suggests either the LLM policy cannot generate high-scoring states or the PRM cannot identify them. Since `PASS@K` significantly outperforms Best-of-N, the LLM policy does generate correct solutions but the PRM fails to rank them highly, providing evidence that the PRM poorly approximates $p(v(s) = 1 \mid s)$.

Further analysis shows that point-biserial correlation of prediction correctness with PRM scores is initially high (≈ 0.5) near terminal states but deteriorates significantly for early reasoning steps (≈ 0.37 at 10 steps from termination; see All data in Figure 1). These findings suggest credit assignment issues in the offline reinforcement learning of the PRM. Moreover, this pattern explains why `MCTS@8`, which relies exclusively on terminal PRM scores, achieves the best average rank among tree search methods, and `Beam@8` performs best in accuracy by tolerating locally suboptimal steps that lead to higher-scoring future states. This suggests that the PRM operates between Scenarios 2 and 3: terminal scores are most reliable, but intermediate scores provide useful yet unreliable guidance.

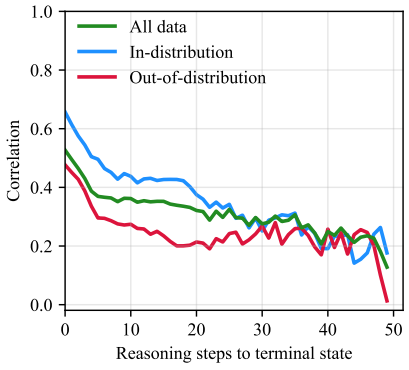

Figure 1: **Correlation of prediction correctness with PRM scores.** Correlation decreases with increasing distance in reasoning steps from terminal states.

4. The PRM shows limited out-of-distribution generalization. Correlation between PRM scores and correctness is consistently higher on in-distribution (ID) problems

on which the PRM is trained (GSM8K and MATH) than out-of-distribution (OOD) tasks (others problems; Figure 2 and Table 5). This generalization gap persists across most reasoning steps: correlation of prediction correctness with PRM scores on ID problems is considerably larger than on OOD tasks until ≈ 30 steps to termination, after which both ID and OOD performance converge to similarly low correlation levels (see ID and OOD in Figure 1). This limited generalization further constrains the practical utility of PRM-guided tree search across diverse mathematical domains.

5 Related work

Tree search with LLM self-evaluation Several works apply greedy best-first search [9], Monte Carlo tree search [16, 17] and beam search [18] to reasoning using the LLM policy model itself for state evaluation. Unlike these approaches, we use a process reward model trained by offline reinforcement learning on mathematics tasks to guide search.

PRM-guided tree search Zhang et al. [3] evaluate PRM aggregation methods and greedy search, finding greedy search generally inferior to Best-of-N. Our study extends this analysis with a more comprehensive evaluation, including additional tree search methods (MCTS, GBFS, beam search) and more than three times as many problems (23 vs. 7). Our findings challenge some of theirs results, notably that last-token aggregation outperforms product aggregation on average. More importantly, we provide evidence that PRM reliability degrades with reasoning depth, a key finding that explains why tree search methods fail to outperform Best-of-N.

Tree search for LLM training Recent work has used MCTS within reinforcement learning pipelines to improve both language models and PRMs. Zhang et al. [19], Wan et al. [20], Xie et al. [21], Guan et al. [22] employ MCTS under PRM supervision to generate high-scoring reasoning chains subsequently used to fine-tune both the policy and PRM through self-training. In contrast, our work focuses on pretrained LLM and PRM models without fine-tuning.

6 Conclusion

We proposed an adaptive algorithm to maximize PRM scores over the intractable action space, and empirically investigated PRM-guided tree search across 23 mathematical reasoning problems using Qwen2.5-Math-7B-Instruct and its associated PRM. Our findings show that PRM-guided tree search methods fail to outperform Best-of-N despite higher costs, with MCTS and beam search proving most effective among PRM-guided tree search approaches. We identify the underlying causes: PRMs poorly approximate state values, become less reliable with reasoning depth indicating credit assignment issues, and exhibit limited out-of-distribution generalization restricting their broader applicability. Tree search's underperformance stems from its reliance on these unreliable intermediate-step PRM scores to guide search, whereas BoN evaluates only complete CoTs. These results highlight the limitations of both PRM-guided tree search and BoN, revealing that current PRMs lack sufficient accuracy to guide dynamic mathematical reasoning and suggesting that different reward models may be required.

Acknowledgments and Disclosure of Funding

GP is supported by the Canada CIFAR AI Chair program. The authors are grateful to Johannes Zenn for many insightful discussions that helped shape the direction of this project, and to both Johannes Zenn and Arsen Sheverdin for their careful reading of the manuscript and valuable feedback that improved its clarity and presentation. The computational resources used in this work were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and by companies sponsoring the Vector Institute[2].

[2] https://vectorinstitute.ai/partnerships/

Bibliography

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

[2] Zayne Rea Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=w6nlcS8Kkn.

[3] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning, 2025. URL https://arxiv.org/abs/2501.07301.

[4] 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, 2024. URL https://arxiv.org/abs/2409.12122.

[5] 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 *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

[6] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

[7] 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*, 2025. URL https://openreview.net/forum?id=LdH0vrgAHm.

[8] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie 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. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui 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. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin 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, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha,

Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[9] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models, 2023.

[10] Martin L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979. URL https://onlinelibrary.wiley.com/doi/abs/0012-9682(197905)47:3<641:OSFTBA>2.0.CO;2-1.

[11] Qian Xie, Raul Astudillo, Peter I. Frazier, Ziv Scully, and Alexander Terenin. Cost-aware bayesian optimization via the pandora's box gittins index. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Ouc1F0Sfb7.

[12] Donald Jones, Matthias Schonlau, and William Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 12 1998. doi:10.1023/A:1008306431147.

[13] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023.

[14] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

[15] Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility, 2025. URL https://arxiv.org/abs/2504.07086.

[16] Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b, 2024. URL https://arxiv.org/abs/2406.07394.

[17] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=VTWWvYtF1R.

[18] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Self-evaluation guided beam search for reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Bw82hwg5Q3.

[19] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=8rcFOqEud5.

[20] Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[21] Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

[22] Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=5zwF1GizFa.

[23] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[24] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

[25] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[26] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[27] Vector Institute. Vector-Inference: Efficient llm inference on slurm clusters, 2025. URL https://github.com/VectorInstitute/vector-inference. GitHub repository, accessed 2025-09-04.

A Additional method details

We here provide additional details on the method presented in Section 3.

Maximizing over the intractable $\mathbb{A}$ using Gittin's indices. At each state s, we sample independent actions from the policy $a \sim \pi_{\boldsymbol{\theta}}(\cdot \mid s)$ and evaluate $f = f_{\boldsymbol{\phi}}(s \oplus a)$, which induces the distribution

$$p(\mathbf{f} = f \mid s) = \sum_{a \in \mathbb{A}} \mathbb{1}(f = f_{\boldsymbol{\phi}}(s \oplus a)) \pi_{\boldsymbol{\theta}}(a \mid s). \tag{A.1}$$

After sampling i actions from the policy, let $m_i = \max_{j \leq i} f_j$ denote the maximum PRM score observed so far. We face a choice between two actions:

1. **Sample** a new action $a_{i+1} \sim \pi(\cdot \mid s)$, observe $f_{i+1} = f_{\boldsymbol{\phi}}(s \oplus a_{i+1})$, and incur cost c
2. **Stop** and commit to the current best action with score m_i

The expected payoff for sampling is $\mathbb{E}_{f \sim p(f|s)}[\max(m_i, f)] - c$, while stopping yields payoff m_i. The optimal policy samples when the sampling payoff exceeds the stopping payoff:

$$\mathbb{E}_{f \sim p(f|s)}[\max(m_i, f)] - c > m_i \tag{A.2}$$

$$\Leftrightarrow \quad \mathbb{E}_{f \sim p(f|s)}[\max(0, f - m_i)] > c \tag{A.3}$$

The Gittins index m_i^* is defined as the unique solution to

$$\mathbb{E}_{f \sim p(f|s)}[\max(0, f - m_i^*)] = c \tag{A.4}$$

The optimal policy samples if $m_i^* > m_i$ and stops otherwise [10]. However, computing this expectation requires knowledge of $p(f \mid s)$ which is intractable due to the summation over the action space $\mathbb{A}$ (see Equation (A.1)). Since estimating this distribution would require the very LLM samples we aim to collect sparingly, we develop a Bayesian surrogate approach inspired by Xie et al. [11] which we discuss next.

Bayesian surrogate modeling We approximate $p(f \mid s)$ using a surrogate model $q(f \mid s, \boldsymbol{\psi})$ with parameters $\boldsymbol{\psi}$. Specifically, we encode our prior beliefs in $p(\boldsymbol{\psi})$, then update the posterior $p(\boldsymbol{\psi} \mid \mathcal{D}) \propto q(\mathcal{D} \mid s, \boldsymbol{\psi}) p(\boldsymbol{\psi})$ with observations $\mathcal{D} = \{f_i | f_i \sim p(f \mid s)\}$ before using the posterior predictive $q(f \mid s, \mathcal{D}) = \int q(f \mid s, \boldsymbol{\psi}) p(\boldsymbol{\psi} \mid \mathcal{D}) d\boldsymbol{\psi}$ to approximate $p(f \mid s)$. We then compute the Gittin's index m_i^* under posterior beliefs $q(f \mid s, \mathcal{D})$ rather than $p(f \mid s)$ by solving

$$\mathbb{E}_{f \sim q(f|s,\mathcal{D})}[\max(0, f - m_i)] = c \tag{A.5}$$

The left-hand side of Equation (A.5) is the expected improvement over the current maximum m [12], a standard Bayesian optimization acquisition function [13]. The Gittin's index m^* represents the threshold where expected improvement equals cost c, thus smaller c induces more exploration. More details in Algorithm 1.

Adaptive cost scheduling. To balance exploration and exploitation over the search horizon, we employ time-varying costs:

$$c(n) = c_1 + (c_2 - c_1) \times {}^{n}/_{B} \tag{A.6}$$

where $c_1 < c_2$ are initial and final costs, B is the total sampling budget, and n is the current sample count. This schedule promotes exploration early during search when the remaining sampling budget is large and exploitation as resources diminish.

Logit-Normal surrogate model Let $\mathcal{D}_n = \{f_i\}_{i=1}^n$ where $f_i \in [0, 1]$ be a collection of observations from $p(\mathbf{f} = f \mid s)$ at a given state s. We consider a logit-Normal likelihood model for our observations i.e.

$$q(f \mid s) = \mathcal{N}\left(\text{logit}(f); \mu, \sigma^2\right) \frac{1}{f(1-f)} \tag{A.7}$$

where $\frac{1}{f(1-f)}$ adjusts for the change of variable. We then specify a Normal-inverse-Gamma prior on the likelihood parameters μ and σ^2 i.e.

$$q(\mu, \sigma^2) = \mathcal{N}\left(m_0, v_0 \sigma^2\right) \mathcal{IG}(\alpha_0, \beta_0) \tag{A.8}$$

Algorithm 1 Adaptive PRM-guided tree search using Gittin's indices and surrogate modeling.

function GITTINS($f_{\boldsymbol{\phi}}$, $\pi_{\boldsymbol{\theta}}$, $\mathbb{T}$, s, K, c)
  repeat
    $\mathcal{D} \leftarrow \emptyset$
    repeat
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_i, f_{\boldsymbol{\phi}}(s_i)) \mid a_i \sim \pi_{\boldsymbol{\theta}}(\cdot \mid s), s_i = s \oplus a_i\}_{i=1}^{K}$ ▷ *Sample from LLM*
      $m \leftarrow \max_{(s,f)\in\mathcal{D}} f$ ▷ *Update current observed maximum*
      Compute posterior predictive $q(f \mid s, \mathcal{D})$ ▷ *Update posterior beliefs*
      Compute Gittin's index m^* by solving $\mathbb{E}_{q(f|\mathcal{D})}\left[\max(0, f - m^*)\right] = c$ using bisection
    until $m^* > m$
    $s \leftarrow \max_{(s,f)\in\mathcal{D}} f$ ▷ *Update current state*
  until $s \in \mathbb{T}$
  return s

The model is conjugate and both the posterior and the predictive posterior are available in closed form. The prior is chosen to yield a maximally uniform predictive prior.

The prior predictive is

$$q(f \mid \mu, \sigma^2) = \int q(r \mid \mu, \sigma^2) q(\mu, \sigma^2) d\mu d\sigma^2 = \mathcal{T}_{2a_0}\left(f; m_0, \frac{b_0(1+v_0)}{a_0}\right) \frac{1}{f(1-f)} \tag{A.9}$$

The posterior is then

$$q(\mu, \sigma^2 \mid \mathcal{D}) = \frac{\prod_{i=1}^n q(f_i \mid s) q(\mu, \sigma^2)}{\int \prod_{i=1}^n q(f_i \mid s) q(\mu, \sigma^2) d\mu d\sigma^2} \tag{A.10}$$

$$= \frac{C \prod_{i=1}^n \mathcal{N}\left(\text{logit}(f_i); \mu, \sigma^2\right) q(\mu, \sigma^2)}{C \int \prod_{i=1}^n \mathcal{N}\left(\text{logit}(f_i); \mu, \sigma^2\right) q(\mu, \sigma^2) d\mu d\sigma^2} \tag{A.11}$$

$$= \frac{\prod_{i=1}^n \mathcal{N}\left(\text{logit}(f_i); \mu, \sigma^2\right) q(\mu, \sigma^2)}{\int \prod_{i=1}^n \mathcal{N}\left(\text{logit}(f_i); \mu, \sigma^2\right) q(\mu, \sigma^2) d\mu d\sigma^2} \tag{A.12}$$

$$= \mathcal{N}\left(m_n, v_n \sigma^2\right) \mathcal{IG}(\alpha_n, \beta_n) \tag{A.13}$$

where $C = \prod_{i=1}^n \frac{1}{f_i(1-f_i)}$, $v_n^{-1} = v_0^{-1} + n$, $m_n = v_n^{-1}(v_0^{-1} m_0 + \sum_{i=1}^n \text{logit}(f_i))$, $a_n = a_0 + n/2$ and $b_n = b_0 + \frac{1}{2}[m_0^2 v_0^{-1} + \sum_{i=1}^n \text{logit}(f_i)^2 - m_n^2 v_n^{-1}]$.

The predictive posterior is

$$q(f \mid \mathcal{D}) = \int q(f \mid \mu, \sigma^2) q(\mu, \sigma^2 \mid \mathcal{D}) d\mu d\sigma^2 = \mathcal{T}_{2a_n}\left(f; m_n, \frac{b_n(1+v_n)}{a_n}\right) \frac{1}{f(1-f)} \tag{A.14}$$

Furthermore, the marginal likelihood has an analytical formulation:

$$q(\mathcal{D} \mid m_0, v_0, a_0, b_0) = \int \prod_{i=1}^n q(f_i \mid s) q(\mu, \sigma^2) d\mu d\sigma^2 \tag{A.15}$$

$$= \left[\prod_{i=1}^n \frac{1}{f_i(1-f_i)}\right] \left[\int \prod_{i=1}^n q(\text{logit}(f_i) \mid s) q(\mu, \sigma^2) d\mu d\sigma^2\right] \tag{A.16}$$

$$= C \left[\frac{v_n^{1/2} b_0^{a_0} \Gamma(a_n)}{v_0^{1/2} b_n^{a_n} \Gamma(a_0) \pi^{n/2} 2^n}\right] \tag{A.17}$$

where $C = \prod_{i=1}^n \frac{1}{f_i(1-f_i)}$.

Implementation details We estimate the expectation $\mathbb{E}_q\left[\max(0, f - m)\right]$ using quadrature reparameterizing the integral to logit-space

$$\mathbb{E}_q\left[\max(0, f - m)\right] = \int_m^1 (f - m) q(f \mid \mathcal{D}) df \tag{A.18}$$

$$= \int_{\text{logit}(m)}^{\text{logit}(1)} (\text{logit}^{-1}(l) - m) q(l \mid \mathcal{D}) dl \tag{A.19}$$

where $l = \text{logit}(f)$. We solve for the Gittin's index m^* using bissection search as done in Xie et al. [11]. Since the observations f take values in $[0, 1]$, we shrink them using $f_i = \epsilon + (1 - 2\epsilon) f_{\phi}(s_i)$ to avoid singularities at 0 and 1.

B Additional experimental setup details

Models and hyperparameters. We use Qwen2.5-Math-7B-Instruct [23] as our base language model, where actions $a \in \mathcal{A}$ correspond to text sequences terminated by '\n\n'. For process reward modeling, we use Qwen2.5-Math-PRM-7B trained on MATH and GSM8K problems [3]. Following the recommended configuration [23], we set temperature=0.7, top_p=0.8, and repetition_penalty=1.05 for text generation.

Implementation. Gittins indices are computed via bisection search and numerical integration performed using SciPy [24]. Tree structures are managed through NetworkX [25]. Models are served using vLLM [26] with our inference pipeline adapted from vector-inference [27]. Evaluation follows the protocol established by Yang et al. [4] using their official codebase.[3]

Computational resources. All experiments are conducted on NVIDIA RTX 6000 and A40 GPUs, with each model deployed on a single GPU.

C Additional experimental results

Table 3: **Performance metrics across 23 problems: mean accuracy with standard-error, mean rank with standard-error, reasoning steps, generated tokens (Gen.), output tokens (Out.), and terminal state coverage.** Bold entries indicate no significant difference from the best method ($p > 0.05$). Best-of-N with terminal PRM scores (`BoN_Last@8`) perform best, while tree search methods (MCTS, beam search) match accuracy but require substantially higher computational cost.

Method	Accuracy	Rank	Reasoning steps	Gen. Tokens ($\times 10^7$)	Out. Tokens ($\times 10^7$)	Terminal State (%)
`PASS@8`	**79.8 ± 4.7 (N/A)**	N/A	9.7	8.2	8.2	98.2
`MAJ@8`	71.4 ± 5.1 (0.010)	4.43 ± 0.51 (0.022)	9.7	8.2	8.2	98.2
`BoN_Last@8`	**72.7 ± 5.1 (N/A)**	**3.13 ± 0.38 (N/A)**	9.7	8.2	8.2	98.2
`BoN_Avg@8`	**72.1 ± 5.0 (0.444)**	**3.22 ± 0.31 (0.787)**	9.7	8.2	8.2	98.2
`BoN_Min@8`	**72.2 ± 5.0 (0.711)**	**3.26 ± 0.32 (0.608)**	9.7	8.2	8.2	98.2
`BoN_Prod@8`	**72.0 ± 5.0 (0.408)**	**3.26 ± 0.40 (0.795)**	9.7	8.2	8.2	98.2
`BoN_Sum@8`	67.6 ± 5.3 (0.000)	7.30 ± 0.72 (0.000)	9.7	8.2	8.2	98.2
`BoN_Max@8`	68.8 ± 5.4 (0.000)	7.35 ± 0.69 (0.000)	9.7	8.2	8.2	98.2
`Greedy@6`	71.6 ± 5.0 (0.043)	5.74 ± 0.76 (0.003)	8.9	5.6	0.9	97.9
`Greedy@20`	71.2 ± 5.0 (0.039)	5.09 ± 0.55 (0.009)	8.8	18.7	0.9	98.2
`Beam@4 (V)`	**71.8 ± 4.9 (0.126)**	**3.83 ± 0.42 (0.236)**	9.2	20.2	0.9	98.7
`Beam@4 (CV)`	**71.9 ± 4.8 (0.189)**	**4.00 ± 0.49 (0.221)**	9.4	21.0	0.9	96.5
`GBFS@8`	46.0 ± 4.1 (0.000)	10.09 ± 0.71 (0.000)	5.3	65.2	0.5	52.7
`GBFS_DA@8`	48.1 ± 4.6 (0.000)	10.00 ± 0.65 (0.000)	5.8	68.0	0.5	57.1
`MCTS`	**71.2 ± 5.0 (0.987)**	**3.26 ± 0.69 (0.856)**	8.8	89.4	0.7	97.8
`Gittins@0.05` (ours)	70.5 ± 5.2 (0.012)	5.96 ± 0.59 (0.001)	9.5	9.0	0.9	98.3
`Gittins@linear` (ours)	71.4 ± 5.1 (0.013)	4.74 ± 0.56 (0.011)	9.5	14.3	0.9	98.5

[3] https://github.com/QwenLM/Qwen2.5-Math

Table 4: **Accuracy by dataset with means and standard errors.** Bold indicates best performance per dataset. Bottom rows show overall means and win counts across problems.

Dataset	PASS@8	MAJ@8	BoN_Last@8	BoN_Avg@8	BoN_Min@8	BoN_Prod@8	BoN_Sum@8	BoN_Max@8	Greedy@6	Greedy@20	Gittins@0.05 (ours)	Gittins@1inear (ours)	Beam@4 (V)	Beam@4 (CV)	GBFS@8	GBFS_DA@8	MCTS@8
AIME25	20.0	13.3	13.3	16.7	16.7	20.0	13.3	6.7	16.7	10.0	13.3	16.7	20.0	16.7	6.7	3.3	**23.3**
AIME24	23.3	16.7	16.7	20.0	20.0	20.0	16.7	10.0	16.7	26.7	16.7	13.3	16.7	**30.0**	6.7	13.3	26.7
AMC23	82.5	60.0	**70.0**	67.5	**70.0**	**70.0**	62.5	62.5	62.5	62.5	55.0	**70.0**	67.5	65.0	40.0	37.5	65.0
SAT_MATH	100.0	96.9	**100.0**	**100.0**	**100.0**	**100.0**	93.8	93.8	**100.0**	96.9	**100.0**	96.9	96.9	96.9	62.5	75.0	96.9
AQUA	94.1	78.0	76.0	76.0	75.6	76.0	73.2	74.4	79.5	79.5	78.0	75.5	77.6	79.5	63.8	63.0	68.3
ASDIV	96.7	95.8	96.0	96.0	95.9	96.0	95.9	95.5	**96.1**	96.0	95.8	95.8	96.0	95.8	70.7	76.1	96.0
CARP_EN	63.1	61.8	61.7	61.9	**62.0**	**62.0**	61.7	61.5	61.0	61.2	61.1	61.1	61.1	61.5	36.1	39.2	61.5
CMATH	95.3	92.7	93.2	93.8	**94.0**	**94.0**	92.2	92.2	92.3	93.2	92.8	93.3	93.0	93.5	69.8	73.5	**94.0**
CN_MIDDLE_SCHOOL	82.2	78.2	**80.2**	79.2	79.2	79.2	76.2	76.2	**80.2**	78.2	77.2	77.2	77.2	78.2	54.5	61.4	79.2
GAOKAO_MATH_CLOZE	81.4	76.3	78.0	78.0	78.0	78.0	72.9	76.3	78.0	76.3	76.3	**79.7**	78.8	77.1	47.5	57.6	75.4
GAOKAO_MATH_QA	94.6	73.5	75.8	**77.8**	77.2	77.5	56.1	73.5	68.4	70.4	72.9	72.9	70.9	63.8	52.1	56.4	75.5
GAOKAO2023EN	80.8	72.2	73.0	71.9	72.2	71.4	69.9	67.0	69.4	69.6	69.6	71.7	71.7	70.1	36.6	40.5	**75.6**
GAOKAO2024_I	71.4	57.1	57.1	50.0	57.1	50.0	42.9	57.1	64.3	64.3	64.3	57.1	57.1	57.1	42.9	35.7	64.3
GAOKAO2024_II	85.7	64.3	**71.4**	64.3	57.1	57.1	50.0	50.0	**71.4**	57.1	57.1	64.3	64.3	64.3	35.7	28.6	57.1
GAOKAO2024_MIX	79.1	72.5	73.6	71.4	71.4	71.4	59.3	71.4	65.9	68.1	64.8	67.0	68.1	69.2	46.2	36.3	**79.4**
GSM8K	97.7	96.6	96.4	96.4	96.4	96.4	96.1	95.7	96.1	96.0	95.6	96.0	**96.7**	96.6	46.3	54.4	**96.7**
MAWPS	98.8	**98.5**	98.4	**98.5**	**98.5**	**98.5**	98.4	98.3	98.1	98.3	98.4	98.4	**98.5**	98.3	72.5	81.7	98.4
MINERVA_MATH	48.9	41.5	39.3	40.1	40.1	39.3	37.1	36.0	38.2	38.2	39.3	37.9	40.8	40.4	14.0	15.1	**42.1**
MMLU_STEM	90.2	72.9	**74.0**	73.4	73.7	72.9	69.0	70.2	72.0	72.4	72.2	**74.0**	73.6	73.5	54.6	43.0	31.7
SVAMP	96.7	94.6	95.0	95.1	95.2	95.1	94.2	93.4	94.9	95.6	95.5	95.9	95.6	95.7	68.7	76.4	**96.8**
TABMWP	98.6	96.1	96.4	96.5	96.4	**96.9**	95.0	94.6	95.5	96.0	95.3	96.1	96.6	96.7	63.3	68.3	96.2
OLYMPIADBENCH	61.9	44.7	**48.4**	47.3	46.1	46.7	42.1	42.7	44.1	45.5	43.3	44.9	46.1	46.1	21.6	21.9	47.9
MATH	92.3	87.0	88.2	87.6	87.5	87.3	85.5	84.5	86.0	86.4	86.7	86.4	86.7	86.8	44.7	48.8	**89.1**
Average	79.8 ± 4.7	71.4 ± 5.1	72.7 ± 5.1	72.1 ± 5.0	72.2 ± 5.0	72.0 ± 5.0	67.6 ± 5.3	68.8 ± 5.4	71.6 ± 5.0	71.2 ± 5.0	70.5 ± 5.2	71.4 ± 5.1	71.8 ± 4.9	71.9 ± 4.8	46.0 ± 4.1	48.1 ± 4.6	71.2 ± 5.0
Best Count	N/A	1/23	5/23	2/23	4/23	5/23	0/23	0/23	4/23	0/23	1/23	3/23	1/23	1/23	0/23	0/23	6/23

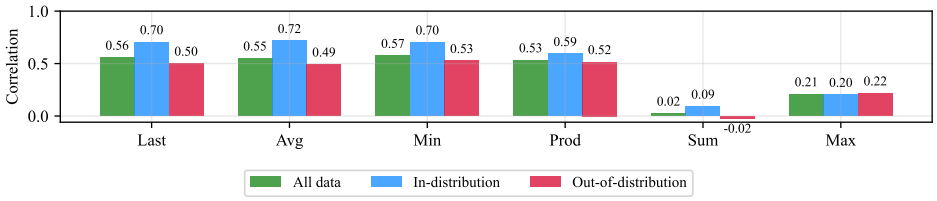

Figure 2: **Point-biserial correlation between PRM score aggregation methods and solution correctness.** Minimum aggregation shows highest correlation, followed by last reasoning-step scoring. Correlation is consistently higher on in-distribution (ID) problems on which the PRM is trained (GSM8K and MATH) than out-of-distribution (OOD) tasks.

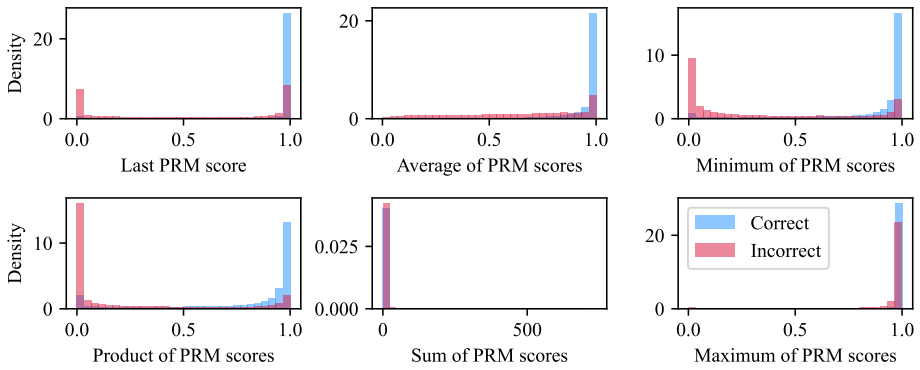

Figure 3: **PRM score distributions by aggregation method, conditioned on prediction correctness.** Last, average, minimum, and product aggregations effectively separate correct from incorrect predictions.

Table 5: **Point-biserial correlations between PRM aggregation methods and prediction correctness by dataset.** Bold indicates best performance per dataset. Correlations vary substantially across problems, with optimal aggregation being dataset-dependent. Overall, minimum aggregation performs best, followed by last-step scoring.

DATASET	BoN_Last@8	BoN_Avg@8	BoN_Min@8	BoN_Prod@8	BoN_Sum@8	BoN_Max@8
AIME25	0.494	0.474	**0.582**	0.444	0.283	0.095
AIME24	0.623	0.597	**0.680**	0.487	0.529	0.151
AMC23	0.668	0.700	**0.726**	0.626	0.170	0.280
SAT_MATH	**0.690**	0.375	0.487	0.330	0.069	0.039
AQUA	0.309	0.309	**0.342**	0.306	0.106	0.291
ASDIV	0.447	**0.466**	0.450	0.424	-0.049	0.116
CARP_EN	0.137	0.149	**0.154**	0.150	-0.026	0.060
CMATH	0.557	0.559	0.627	**0.637**	0.063	0.200
CN_MIDDLE_SCHOOL	0.432	0.419	**0.465**	0.444	-0.161	0.119
GAOKAO_MATH_CLOZE	0.455	0.500	**0.555**	0.482	0.094	0.098
GAOKAO_MATH_QA	0.510	0.454	**0.546**	0.488	-0.016	0.216
GAOKAO2023EN	0.561	0.590	**0.602**	0.540	0.148	0.175
GAOKAO2024_I	0.285	0.275	**0.438**	0.337	-0.064	0.126
GAOKAO2024_II	0.376	0.447	0.545	**0.584**	0.048	0.094
GAOKAO2024_MIX	0.482	0.422	**0.495**	0.343	0.081	0.187
GSM8K	0.560	0.595	**0.597**	0.565	0.099	0.120
MAWPS	0.333	**0.379**	0.322	0.296	0.015	0.091
MINERVA_MATH	0.349	0.416	**0.477**	0.474	-0.022	0.183
MMLU_STEM	**0.360**	0.280	0.281	0.266	0.085	0.182
SVAMP	0.700	**0.706**	0.694	0.673	0.109	0.239
TABMWP	0.542	**0.591**	0.588	0.570	0.090	0.211
OLYMPIADBENCH	0.571	0.603	**0.635**	0.578	0.213	0.145
MATH	0.708	**0.718**	0.703	0.585	0.130	0.200
BEST COUNT	2/23	5/23	14/23	2/23	0/23	0/23
AVERAGE	0.557	0.550	0.575	0.533	0.024	0.209