SCALING FLAWS OF VERIFIER-GUIDED SEARCH IN MATHEMATICAL REASONING

Fei Yu, Yingru Li[†], Benyou Wang[†]
School of Data Science
The Chinese University of Hong Kong, Shenzhen
Shenzhen, China
yufei21@outlook.com,szrlee@gmail.com,wangbenyou@cuhk.edu.cn

ABSTRACT

Large language models (LLMs) struggle with multi-step reasoning, where inference-time scaling has emerged as a promising strategy for performance improvement. Verifier-guided search outperforms repeated sampling when sample size is limited by selecting and prioritizing valid reasoning paths. However, we identify a critical limitation: scaling flaws, prevalent across different models (Mistral 7B and DeepSeekMath 7B), benchmarks (GSM8K and MATH), and verifiers (outcome value models and process reward models). As sample size increases, verifier-guided search exhibits diminishing advantages and eventually underperforms repeated sampling. Our analysis attributes this to verifier failures, where imperfect verifiers misrank candidates and erroneously prune all valid paths. These issues are further exacerbated in challenging and out-of-distribution problems, restricting search effectiveness. To mitigate verifier failures, we explore reducing reliance on verifiers and conduct preliminary investigations using two simple methods. Our findings reveal fundamental limitations in verifier-guided search and suggest future directions.

1 INTRODUCTION

Multi-step reasoning is challenging to LLMs (Hendrycks et al., 2021; Zheng et al., 2022). Recent studies have identified inference-time scaling (Brown et al., 2024; Snell et al., 2024; Wu et al., 2024b) as a promising strategy to enhance LLM performance on multi-step reasoning. By increasing inference-time computation through multiple attempts via repeated sampling (Brown et al., 2024), LLMs can solve more problems, with at least one attempt succeeds. Building on this insight, search-based approaches have emerged to guide computation toward more effective reasoning paths (Snell et al., 2024; Wu et al., 2024b).

Search reallocates computational resources by evaluating and selecting partial paths during generation. A common approach for path evaluation uses verifiers (Snell et al., 2024; Wu et al., 2024b), such as outcome value models (OVMs) (Yu et al., 2024) and process reward models (PRMs) (Lightman et al., 2024), to score and rank candidates, prioritizing valid paths. This makes verifier-guided search effective for challenging problems with sparse valid solutions, offering advantages over repeated sampling when the sample size is limited.

Obvervation of scaling flaws. However, we observe that verifier-guided search (e.g. OVM- and PRM-guided) might experience diminishing advantages and eventually underperforms repeated sampling as the sample size scales. Its performance improves more slowly than repeated sampling, ultimately making them less effective. We refer to this phenomenon as *scaling flaws of verifier-guided search*.

Identification of verifier failures. To understand the cause of scaling flaws, we analyze search failures and identify verifier selection failures as the main factor, where imperfect verifiers misrank and incorrectly prune all valid paths—an issue we term "verifier failures". Morever, verifier selection itself exhibits scaling issues: as the candidate size increases, valid paths become more widespread

[†]Corresponding to Yingru Li and Benyou Wang.

across the problem set, yet verifiers struggle to identify them, leading to their erroneous pruning. This contributes to the overall search scaling flaws.

Analysis of verifier failures. Our investigation shows that verifier failures and scaling flaws worsen in challenging and out-of-distribution problems. As problem difficulty and solution sparsity increase, scaling flaws intensify. This paradoxically undermines search, which is intended to outperform repeated sampling in such cases. Moreover, out-of-distribution problems, common in real-world deployment, exacerbate these challenges, highlighting fundamental limitations of verifier-guided search approaches.

Mitigating verifier failures. To explore potential approaches for mitigating verifier failures, we conduct a preliminary investigation into two simple methods that reduce reliance on verifiers, both of which demonstrate benefits.

Summary of contributions. (1) This work identifies and analyzes the scaling flaws of verifier-guided search (2) We pinpoint verifier failures as the primary cause of these flaws (3) Our analysis reveals that these issues become more severe for challenging and out-of-distribution problems, raising concerns about the development of verifier-guided search algorithms and their application in real-world settings (4) We suggest reducing reliance on verifiers and conduct preliminary investigations using two simple methods.

2 Related Works

Search algorithms Search algorithms often face a tradeoff between effectiveness and efficiency. Approaches like Monte Carlo Tree Search (Hao et al., 2023; Tian et al., 2024) improve effectiveness by incorporating backtracking, but at the cost of efficiency. Other methods prioritize efficiency with minimal sacrifice in effectiveness (Wu et al., 2024a). In this work, we use a simple beam search algorithm (Yu et al., 2024; Chen et al., 2024) for our experiments, focusing on highlighting challenges in the candidate evaluation and selection stage, orthogonal to these advanced techniques.

Candidate evaluation in search Candidate evaluation is a crucial stage that determines which paths are more valuable for further selection and exploration. Some methods rely on the some rule-based heuristics (Xin et al., 2024), with limited effectiveness. Some approaches involve lookahead techniques to assess candidates by simulating their subsequent outcomes (Snell et al., 2024; Wan et al., 2024), which significantly increases computational cost. Other methods incorporate external verifier models (Yu et al., 2024; Snell et al., 2024) to evaluate each candidate. In this work, we focus on the challenges and limitations of the this approach.

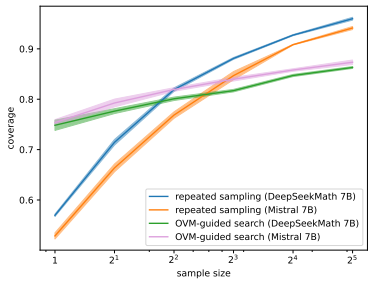
3 Background: Verifier-Guided Search

This section begins by defining mathematical reasoning questions and introducing two widely employed solution frameworks: repeated sampling and search. We then detail a specific search framework, beam search, and discuss two widely-used verifier types employed in the search process.

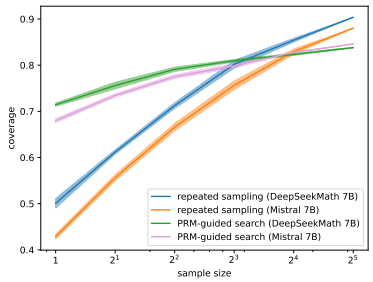
Definition. A mathematical reasoning question *q requires a step-by-step solution path $S = [s^1, \ldots, s^T, a]$ to be addressed, where s^i represents the i-th step, T is the number of steps, and a is the final answer.*

Multi-step reasoning (Cobbe et al., 2021; Hendrycks et al., 2021) suffers from error propagation issues–errors in earlier steps affect later ones, compromising the final answer. Recent studies show that LLMs can address more challenging problems through repeated sampling (Brown et al., 2024).

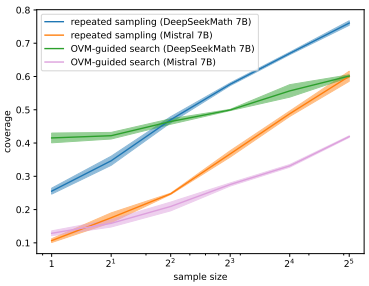
Repeated sampling LLMs can solve some challenging problems through multiple attempts (Cobbe et al., 2021; Brown et al., 2024), i.e. repeatedly sampling a set of solution paths $\{S_k\}_{k=1}^{K}$ from the generator. Increasing the number of attempts, K, often improves the coverage—the fraction of problems for which at least one sampled path is correct, but also requires more computation.



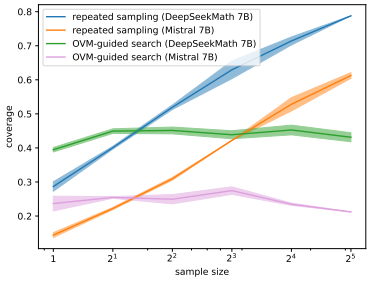
(a) OVM on GSM8K



(b) PRM on GSM8K



(c) OVM on MATH



(d) PRM on MATH

Figure 1: Scaling Flaws in OVM-guided search and PRM-guided search on GSM8K and MATH (scaling of sample sizes). While verifier-guided search outperforms repeated sampling initially, its performance increases at a slower rate, ultimately underperforming repeated sampling.

However, repeated sampling becomes inefficient for challenging problems, like competition-level mathematics problems (Hendrycks et al., 2021), where it often demands many more attempts to find a correct solution (Brown et al., 2024).

3.1 SEARCH

Search aims to explore correct solutions more efficiently than repeated sampling by pruning unpromising partial paths and discarding early errors. This paper focuses on *step-level beam search*, a widely used and sufficiently straightforward framework for illustrating the core concept.

Step-level beam search This framework intervenes in generation and selection at the step level and explore multiple paths in parallel. Given a question q, at each step t, the generator produces K candidates $\mathbb{S}^{(1:t)} = \left\{S_k^{(1:t)}\right\}_{k=1}^{K}$, where $S_k^{(1:t)} = [s_k^1, \ldots, s_k^t]$ is the k-th partial path. During the selection stage, a scoring function f evaluates these candidates, assigning scores $\mathbb{V}^{(1:t)} = \left\{v_k^{(1:t)}\right\}_{k=1}^{K}$, where $v_k^{(1:t)}$ is the score for $S_k^{(1:t)}$, ranking them for selection. The top b paths proceed to the next step, generating K/b new candidates each, maintaining a total of K candidates. This process repeats until all b paths terminate, yielding b full solution paths. See details in Appendix 1. The hyperparameter b controls the number of parallel paths. Larger b or K improve the ability to handle a wider range of problems.

Search using verifiers as scoring functions is particularly noteworthy (Yu et al., 2024; Chen et al., 2024; Snell et al., 2024). We refer to this approach as "verifier-guided search".

3.2 VERIFIERS

Verifiers (Lightman et al., 2024; Yu et al., 2024) are commonly employed as scoring functions to evaluate candidate, determining which paths to be further explored. In this work, we focus on the

two most widely used types of verifiers–Outcome-supervised Value Models (Yu et al., 2024) and Process-supervised Reward Models (Lightman et al., 2024).

Outcome-Supervised Value Model (OVM) The OVM (Yu et al., 2024) evaluates each candidate by estimating the probability of arriving at a correct answer from the given partial path. It assumes that each local step with the highest probability of success ultimately leads to the correct answer. We refer to search using OVM for evaluation as "OVM-guided search".

Process-Supervised Reward Model (PRM) The PRM (Lightman et al., 2024) evaluates each candidate by predicting its step correctness. It assumes that each correct local step guides to the correct final answer. We refer to search using PRM for evaluation as "PRM-guided search".

Verifiers play a key role in candidate evaluation and selection, directly influencing the search success. When they correctly identify valid paths, they can steer the search towards correct solutions more efficiently than repeated sampling.

However, we observe that although the search process initially shows advantages over repeated sampling, these advantages disappear as scaling, as shown in the next section.

4 Scaling Flaws of Verifier-Guided Search

In this section, we present extensive experiments showing that verifier-guided search suffers *scaling flaws*: it outperforms repeated sampling at small sample sizes but underperforms it at large sample sizes. These flaws are worse on more difficult and out-of-distribution problems.

4.1 Experimental Setup

Benchmarks We perform experiments on two mathematical reasoning datasets: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). The experiments are conducted under four distinct settings, including two in-distribution and two out-of-distribution (OOD) scenarios, as detailed below:

- **GSM8K**: The official training split is used for training, and the model is evaluated on the test split.
- **MATH**: The official training split, comprising 7,500 problems, is used for training, while evaluation is performed on the MATH500 (Lightman et al., 2024).
- **OOD-L4**: Training is conducted on MATH Level 1, Level 2, Level 3, and Level 5 problems, while evaluation is performed specifically on Level 4 problems within MATH500. This setting requires models to generalize to problems of median difficulty.
- **OOD-L5**: We train on MATH Level 1 - Level 4 problems and evaluate on Level 5 problems within MATH500. In this setting, models are required to generalize to solve more difficult problems.

Models We use Mistral 7B (Jiang et al., 2023) and DeepSeekMath 7B (Shao et al., 2024) for the GSM8K and MATH experiments, and exclusively use DeepSeekMath 7B for the two OOD settings. For each setting, the base models are trained on the corresponding training sets to serve as the generators. The OVMs used in each setting are initialized from these generators. For PRMs, we leverage the open-source Math-Shepherd dataset (Wang et al., 2024). Generators are first fine-tuned on a subset of this data, after which PRMs, initialized from the corresponding generators, are trained under supervision using process labels.

Scaling beam search We investigate the scaling laws of two factors: (1) the number of parallel explored paths b, with K/b fixed at 8, and (2) the number of generated candidates K, with b fixed at 8. For the comparison between beam search and repeated sampling, we align them in terms of "sample size", which represents the number of complete solution paths generated by each algorithm. For beam search, the sample size corresponds to the number of parallel explored paths, b, while for repeated sampling, it corresponds to the number of attempts. Each experiment is repeated three

times, and we report the average coverage (i.e. the fraction of problems for which at least one sampled path is correct) along with their standard deviation.

See implementation details in Appendix A.4.

4.2 SCALING FLAWS

The results of scaling verifier-guided search are presented in Figure 1-2. Notably, both OVM-guided and PRM-guided encounter scaling flaws across all settings.

Scaling flaws of verifier-guided search *Verifier-guided search encounters scaling flaws across benchmarks and models*. Both OVM-guided and PRM-guided search exhibit failures in scaling sample sizes (Figure 1) and generated candidate sizes (Figure 2). When scaling sample sizes, as shown in Figure 1, both OVM-guided and PRM-guided search initially outperforms repeated sampling, e.g. when the sample size is set to 1, on GSM8K and MATH. However, as the sample size increases, the performance of verifier-guided search increases at a slower rate compared to repeated sampling, ultimately underperforming repeated sampling.

For instance, in Figure 1(b), PRM-guided search based on either DeepSeekMath or Mistral initially achieves over 20% higher performance than repeated sampling when the sample size is 1. However, this advantage erodes as the sample size scales, and by a sample size of 16, verifier-guided search becomes inferior to repeated sampling, reaching approximately 5% lower performance when scaled to 32. Similarly, in Figure 1(c), OVM-guided search based on DeepSeekMath or Mistral is overtaken by repeated sampling at a sample size of 4, eventually falling behind by approximately 20% when the sample size reaches 32.

Moreover, increasing the number of generated candidates fails to improve and even degrades the performance of verifier-guided search, as shown in Figure 2.

Intensified on difficult problems *Scaling flaws are more severe on more difficult problems*. As shown in Figure 1 and Table 1, scaling flaws are more pronounced on MATH than on GSM8K and become increasingly severe as problem difficulty increases within MATH. In Figure 1, the performance degradation—measured as the gap between search and repeated sampling at a sample size of 32—is approximately 10% for OVM-guided search (both DeepSeekMath and Mistral) on GSM8K, increasing to around 20% on MATH. Similarly, for PRM-guided search, the performance degradation rises from about 5% on GSM8K to nearly 30% on MATH.

Furthermore, as observed in Table 1, consistent with previous research (Snell et al., 2024), verifier-guided search shows greater benefits over repeated sampling for moderate problems. For instance, at a sample size of 1, gains are larger for Level 2–Level 4 problems compared to Level 1 and Level 5. However, as problem difficulty increases, the performance degradation of both OVM- and PRM-guided search approximately upward monotonically. Notably, the loss gap exceeds 20% when comparing Level 1 to Level 5 problems, suggesting the increasing severity of scaling flaws.

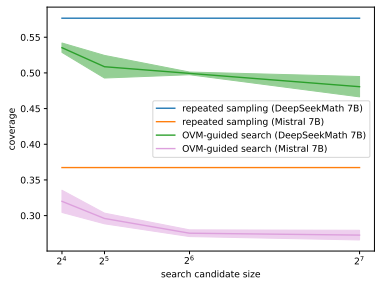
Table 1: Increased average coverage of search over repeated sampling across various problem difficulties on MATH and OOD settings (DeepSeekMath 7B). 'L': 'Level', #sample: sample size.

	#sample	L1	L2	L3	L4	L5	OOD-L4	OOD-L5
OVM	1	11.6%	23.0%	21.3%	17.7%	7.0%	14.1%	1.2%
	32	-3.1%	-6.0%	-15.6%	-19.3%	-23.9%	-25.8%	-32.8%
PRM	1	8.5%	18.9%	12.1%	13.5%	2.5%	8.1%	5.0%
	32	-11.6%	-24.4%	-42.5%	-43.8%	-37.8%	-46.1%	-39.1%

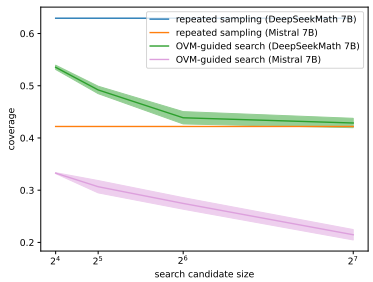
Intensified on OOD problems *Scaling flaws are more severe on OOD problems*. As shown in Figure 1, performance degradation at a sample size of 32 is more pronounced in OOD settings for both OVM- and PRM-guided search. For instance, the performance degradation of OVM-guided search on the in-distribution Level 4 setting is 19.3%, and it increases to 25.8% in the OOD-L4 setting. Similarly, the loss rises from 23.9% on Level 5 to 32.8% in the OOD-L5 setting. These results reveal the exacerbated impact of scaling flaws when generalizing to OOD problems.

A notable concern arises: these findings indicate that the performance degradation of verifier-guided search compared to repeated sampling as scaling is enhanced with increasing problem difficulty.

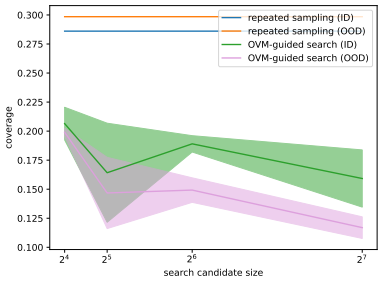
However, this contradicts the purpose of verifier-guided search, which is designed to improve performance in solving difficult problems. Furthermore, out-of-distribution scenarios—commonly encountered in real-world deployment—further exacerbate these scaling flaws.



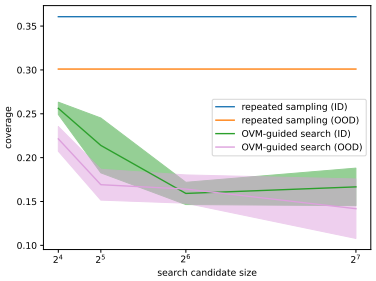
(a) OVM on MATH



(b) PRM on MATH



(c) OVM on OOD-L5



(d) PRM on OOD-L5

Figure 2: Scaling Flaws in OVM-guided search and PRM-guided search on MATH and OOD-L5 (scaling generated candidate size).

5 Verifier Failures

Section 4 observed scaling flaws in verifier-guided search, but the underlying cause remains unknown. This section conducts an in-depth analysis, identifying incorrect selection due to imperfect verifiers as the root cause of these flaws. In Section 5.2, we term this phenomenon as "verifier failures" and analyze its connection to search scaling flaws. In Section 5.3, we investigate the distribution of failed selection stages during the search, examining their correlation with the sparsity of candidate space.

5.1 Experimental setup

In this section, we analyze the selection stages from two perspectives: (1) only the first selection stage with a large number of candidates $K = 256$ to study the relationship between the number of candidates and the performance of verifier selection, including both OVM selection and PRM selection (2) analyze all selection stages during the OVM-guided search with $b = 8, K = 64$, as this configuration suffers from scaling flaws across benchmarks and models while maintaining an acceptable computational cost for valid path labeling.

The selection stages during the search are analyzed based on a single criterion: whether at least one valid path is selected when valid paths are available. A candidate is considered a valid path if it can lead to the correct final answer. To determine valid paths, we complete each partial path by rolling out multiple samples and verifying whether any of the rollouts successfully reach the correct answer. Specifically, we generate 4 rollouts per candidate for GSM8K and 16 rollouts for MATH and OOD settings.

5.2 VERIFIER FAILURES CAUSE SEARCH SCALING FLAWS

Search failures can arise from either the generation stage or the selection stage—specifically, when no valid candidates are generated or when valid paths produced during generation fail to be selected.

Generation vs. selection failures *Search failures are largely attributable to selection failures.* We analyze all search processes in which problems fail to be solved and attribute these failures to either generation or selection. A failure is attributed to the generation stage if there is at least one intermediate step where no valid partial paths are generated. Conversely, it is attributed to the selection stage if, at any intermediate step, valid paths are produced but fail to be selected. As shown in Table 2, a large proportion of OVM-guided search failures occur during the selection stage, highlighting it as a critical issue [1].

Table 2: Fraction of OVM-guided search failure sources across benchmarks and models ('dsm': 'deepseekmath', 'mst': 'mistral'; 'G': 'Generation', 'S': 'Selection').

	GSM8K		MATH		OOD-L4	OOD-L5
	dsm	mst	dsm	mst		
G	11.4%	16.5%	20.0%	22.9%	15.7%	18.8%
S	88.6%	83.5%	80.0%	77.1%	84.3%	81.2%

Selection failures in verifier-guided search are directly attributable to verifiers. When verifiers fail to differentiate between valid and invalid paths, and mistakenly assign low ranks to all valid paths, none of them will be further explored, resulting in a selection failure. We refer to this issue as "verifier failures". Such failures, which prune all valid paths as failing to select any, ultimately lead to search failures.

To validate the role of verifier failures in contributing to search scaling flaws, we examine the relationship between the success of the selection stage and the number of candidates. Specifically, we analyze the performance of verifier selection in correctly identifying and selecting at least one valid path as the number of candidates increases during the first selection stage. To ensure that the analysis accounts for the presence of valid paths in the candidate set, we use oracle selection performance as a baseline. This baseline serves as a reference for the maximum potential success of the selection process, independent of verifier performance.

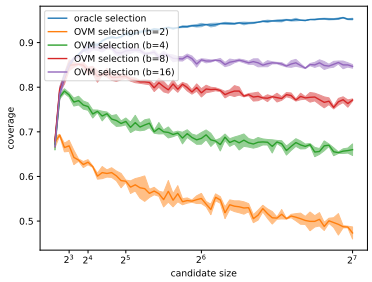
Verifier selection scaling failures *There are verifier selection scaling failures during the selection stage.* As shown in Figure 3, verifier selection exhibits scaling failures. Specifically, the performance of verifier selection improves only marginally, saturates, or even decreases as the candidate size increases, despite the presence of valid paths across more problems, as indicated by the oracle selection performance. This phenomenon is consistent across various beam sizes. While selecting and exploring more candidates improves robustness to verifier limitations—evidenced by the reduced gap between verifier selection and oracle selection performance—a significant gap persists even at the largest beam size tested, $b = 16$. These scaling failures suggest that verifier selection is a key bottleneck in the success of the selection process, and increasing the candidate size offers limited improvement in addressing this issue.

The scaling failure of verifier selection can explain the diminishing advantage of verifier-guided search. Initially, verifier-guided search is more efficient than repeated sampling, as it effectively selects valid paths and reallocates computational resources for several problems. However, as scaling increases, even though valid paths are available across a broader range of problems, verifiers fail to identify and select them. In contrast, repeated sampling explores more paths without being constrained by verifier failures, ultimately outperforming verifier-guided search at larger scales.

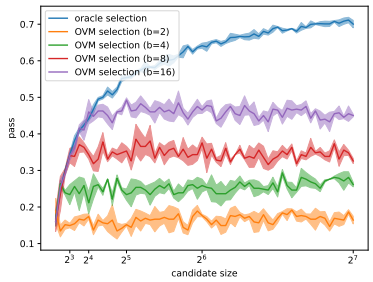
5.3 MORE CHALLENGING SCENARIOS

In this section, we analyze the failed selection stages during the search, showing that the search process is most hindered when valid paths are sparse.

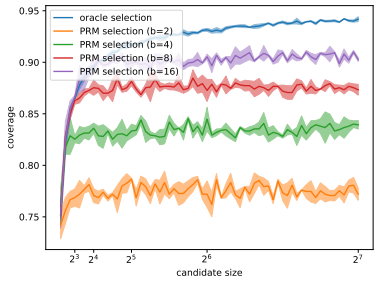
[1] We present only the results of OVM-guided search, as generation failures in PRM-guided search are expected to be similar due to the independence of the generation and selection stages.



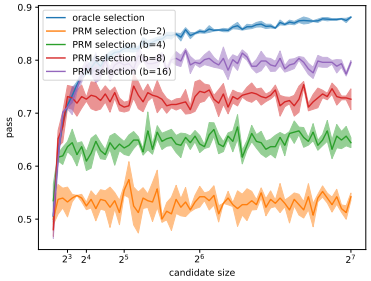
(a) OVM, MATH (DeepSeekMath 7B)



(b) OVM, OOD-L5



(c) PRM, MATH (DeepSeekMath 7B)



(d) PRM, OOD-L5

Figure 3: Scaling failures of verifier selection at the first selection stage across various beam sizes on MATH and OOD-L5.

Sparser candidate space *Verifier selection failures occur and block search more often when valid paths are sparse*. We investigate the failed selection stages during the search and examine the valid path sparsity of these stages. Valid path sparsity is defined as the fraction of valid paths among the candidates. First, we group the valid path sparsity across all selection stages of unsolved problems into four uniform categories. Next, we identify the specific failure stage in each search process where verifier failures occur. we use these groupings to plot the distribution of valid path sparsity across the identified failure stages.

As illustrated in Figure 4, the distribution of failed selection stages demonstrates a monotonic trend: as valid path sparsity decreases, the proportion of failed selection stages increases. This observation aligns with intuition, as identifying valid paths becomes increasingly difficult in sparser spaces.

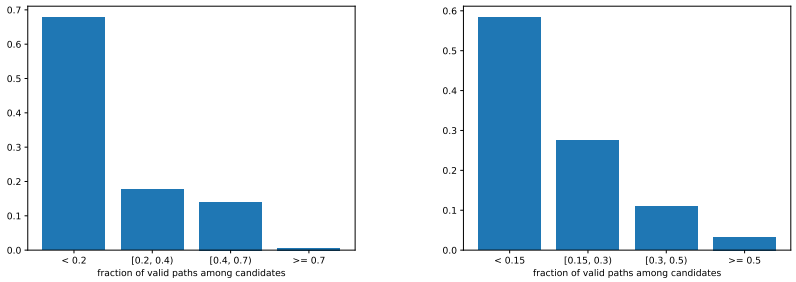
These findings reveal that verifier failures become increasingly significant during the search process, amplifying the risk of search failure when solving sparser correct solution spaces, where the identification and selection of valid paths become considerably harder.

Although search is expected to offer greater efficiency than repeated sampling in solving more challenging problems by reallocating computational resources through effective selection, our observations suggest that these challenging scenarios are more susceptible to verifier failures, thereby exacerbating scaling flaws.

6 Alleviating Verifier Failures

Imperfect verifiers can lead to verifier failures, obstructing the success of the search process. In this section, we explore two simple methods to alleviate verifier failures by reducing dependency on verifiers: stochastic selection and integration with one-time Monte Carlo rollout.

Experimental setup We evaluate these methods across all the selection stages of the search with $b = 8, k = 64$. For each method, we measure the accuracy improvement in the selection stage before and after its application.



(a) MATH (b) OOD-L5

Figure 4: Distribution of OVM failures across groups based on valid path sparisty on MATH and OOD-L5 (DeepSeekMath 7B).

Stochastic selection Imperfect verifiers can produce incorrect candidate rankings, potentially leading to misguided selection decisions. To mitigate the risk of over-reliance on erroneous rankings, we introduce stochasticity into the selection stage. Rather than deterministically selecting candidates based solely on verifier-predicted score rankings, we apply a softmax function to the candidates' scores and sample from the resulting probability distribution. This approach maintains a preference for high-scoring candidates while still allowing lower-scoring ones a chance to be selected, thereby reducing the risk of incorrectly pruning misranked valid paths.

As shown in Table 3, stochastic selection improves selection stage accuracy across all benchmarks and models, regardless of temperature, with a notable improvement of up to 11.2% on OOD-L4 and OOD-L5. Interestingly, on GSM8K, a lower temperature (0.1) yields equal or even greater accuracy gains compared to a higher temperature (10), whereas this trend reverses on MATH and OOD settings. This observation aligns with intuition: since MATH and OOD settings experience more severe verifier failures than GSM8K, reducing reliance on verifier selection through a higher temperature could be more beneficial in these scenarios.

One-time Monte Carlo rollout This method aims to enhance candidate evaluation by incorporating simulated rewards alongside verifier-predicted scores. Specifically, we perform a one-time rollout for each partial path $S^{(1:t)}$ until completion and obtain the reward of the resulting full path. [2] We then linearly combine this reward r with the verifier-predicted score $v^{(1:t)}$ using the formula $\lambda r + (1-\lambda)v^{(1:t)}$, where λ controls the balance between the reward and the verifier's evaluation.

As shown in Table 3, increasing λ generally results in higher accuracy gains. Notably, the highest accuracy gain is achieved when relying entirely on the simulated reward, without incorporating the verifier-predicted score. This underscores the limitations of verifiers in candidate evaluation.

Table 3: Accuracy gain over OVM on the selection stage through two inference-time modification methods (DeepSeekMath 7B).

	GSM8K	MATH	OOD-L4	OOD-L5
temperature in stochastic selection				
0.1	1.6%	2.8%	8.7%	5.3%
1	2.0%	5.4%	11.8%	10.3%
10	1.6%	6.1%	11.2%	11.2%
lambda for one-time Monte Carlo rollout				
0.5	1.3%	-0.8%	2.1%	1.8%
0.75	1.4%	-0.3%	3.0%	2.2%
1	1.8%	1.7%	6.0%	2.5%

[2]The reward is estimated by the same verifier based on the complete path.

7 CONCLUSION

We investigate the scaling flaws of verifier-guided search, identifying verifier failures as their underlying cause. While designed to enhance performance on challenging problems, these methods struggle with scalability as problem complexity grows and in real-world OOD settings. Relaxing the reliance on verifier scores could be a promising direction.

REFERENCES

Bradley C. A. Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *CoRR*, abs/2407.21787, 2024. doi: 10.48550/ARXIV.2407.21787. URL https://doi.org/10.48550/arXiv.2407.21787.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: process supervision without process. *CoRR*, abs/2405.03553, 2024. doi: 10.48550/ARXIV.2405.03553. URL https://doi.org/10.48550/arXiv.2405.03553.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

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 Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 8154–8173. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.507. URL https://doi.org/10.18653/v1/2023.emnlp-main.507.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023. doi: 10.48550/ARXIV.2310.06825. URL https://doi.org/10.48550/arXiv.2310.06825.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In Jason Flinn, Margo I. Seltzer, Peter Druschel, Antoine Kaufmann,

and Jonathan Mace (eds.), *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024. doi: 10.48550/ARXIV.2408.03314. URL https://doi.org/10.48550/arXiv.2408.03314.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. Toward self-improvement of llms via imagination, searching, and criticizing. *CoRR*, abs/2404.12253, 2024. doi: 10.48550/ARXIV.2404.12253. URL https://doi.org/10.48550/arXiv.2404.12253.

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 *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=C4OpREezgj.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.510. URL https://doi.org/10.18653/v1/2024.acl-long.510.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. An empirical analysis of compute-optimal inference for problem-solving with language models. *CoRR*, abs/2408.00724, 2024a. doi: 10.48550/ARXIV.2408.00724. URL https://doi.org/10.48550/arXiv.2408.00724.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024b.

Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *CoRR*, abs/2408.08152, 2024. doi: 10.48550/ARXIV.2408.08152. URL https://doi.org/10.48550/arXiv.2408.08152.

Fei Yu, Anningzhe Gao, and Benyou Wang. Ovm, outcome-supervised value models for planning in mathematical reasoning. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 858–875. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.55. URL https://doi.org/10.18653/v1/2024.findings-naacl.55.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=9ZPegFuFTFv.

A Appendix

Table 4: Summary of Notations Used in the Paper

Notation	Description
q	Mathematical reasoning question requiring a sequence of steps
S	Solution path for a question, $S = [s^1, \ldots, s^T, a]$
s^i	i-th step in a solution path
a	Final answer in a solution path
T	Number of steps in a solution path
y	Binary label (0 or 1) indicating the correctness of a
$S^{(1:t)}$	Partial solution path up to step t, $S^{(1:t)} = [s^1, \ldots, s^t]$
$\mathbb{S}^{(1:t)}$	Set of candidate partial paths $\mathbb{S}^{(1:t)} = \{S_k^{(1:t)}\}_{k=1}^K$
$v^{(1:t)}$	The score for the partial path $S^{(1:t)}$
$\mathbb{V}^{(1:t)}$	Set of scores for candidate partial paths $\mathbb{V}^{(1:t)} = \{v_k^{(1:t)}\}_{k=1}^K$
K	Number of candidates
b	Beam size

A.1 Discussion

This work focuses on scaling flaws related to coverage, rather than precision (Brown et al., 2024). While precision is important for single-response applications, it is often limited by reward models or selection rules for the final selection. Coverage, however, represents the upper bound of precision and directly equates to it in applications with oracle solution selection, such as automatic theorem proving (Zheng et al., 2022) and code generation (Chen et al., 2021).

Limitations We do not investigate the impact of scaling verifier sizes and the size of the training dataset. Larger verifier models and more extensive training data could potentially reduce verifier failures and alleviate scaling flaws.

Future work A promising direction is to reduce reliance on verifier selection, as discussed in this work. Another avenue is detecting verifier failures and adapting verifier usage accordingly. Uncertainty measures could be useful for identifying these failures.

A.2 Verifier Training

OVM training dataset construction OVMs are trained on automatically constructed datasets, where the correctness of the final answer serves as the label for each instance. The training dataset is constructed from the generator and the given question-answer pairs: For each pair $(q, a) \in \mathcal{Q}$, the generator produces n solution paths, resulting in $|\mathcal{Q}| \times n$ question-solution pairs. The label y for each solution S is determined by checking the correctness of the final answer, e.g. matching it to the ground truth a, with 1 indicating "correct" and 0 indicating "incorrect". This process generates a training dataset of (q, S, y) tuples for value models.

PRM training dataset PRMs are trained at a fine-grained step level, requiring annotations of step correctness. In this study, we use the open-source Math-Shepherd process data (Wang et al., 2024) to train the PRMs.

Both OVMs and PRMs are trained with mean squared losses.

A.3 STEP-LEVEL BEAM SEARCH

The algorithm is shown in Appendix 1.

Algorithm 1 Step-Level Beam Search

Input: Question q, Beam size b, Sampled steps per state K, Maximum step count T^{max}
Output: b solution sequences for q
Model: Generator and VM

Initialize step sequences $\mathbb{S} \leftarrow \{\}$
Sample initial steps $\{s_1^1, \ldots, s_K^1\}$
Select b steps via SELECTION(q, $\{s_1^1, \ldots, s_K^1\}$, b, VM) and add to $\mathbb{S}$
$t \leftarrow 1$
while sequences in $\mathbb{S}$ are not complete and $t < T^{max}$ **do**
    $\mathbb{S}_{\text{new}} \leftarrow \{\}$
    for each sequence $S^{(1:t)}$ in $\mathbb{S}$ **do**
        for $i = 1$ to K/b **do**
            $S_i^{(1:t+1)} = \text{Generator}(S_i^{(1:t)}; q)$
            $\mathbb{S}_{\text{new}} \leftarrow \mathbb{S}_{\text{new}} + S_i^{(1:t+1)}$
        end for
    end for
    $\mathbb{S}_{\text{new}} \leftarrow$ SELECTION(q, $\mathbb{S}_{\text{new}}$, b, VM)
    $\mathbb{S} \leftarrow \mathbb{S}_{\text{new}}$
    $t \leftarrow t + 1$
end while
return $\mathbb{S}$

A.4 IMPLEMENTATION DETAILS

A.4.1 OVMS

Training generators We train the base models (Mistral 7B or DeepSeekMath 7B) on the training sets of each setting. In MATH, we split the steps using period and newline characters. We normalize datasets to use the newline character as the marker for the end of each step across all tasks. In all datasets, supervised fine-tuning is performed for 2 epochs with a batch size of 128. We use a linear learning rate scheduler with a maximum learning rate of 2e-6 for Mistral 7B and 5e-5 for DeepSeekMath 7B. The AdamW optimizer (Loshchilov & Hutter, 2019) is used for training.

Building training dataset for OVMs The dataset construction process is introduced in Appendix A.2. We sample 50 solution paths per problem in GSM8K, and 100 solution paths per problem in MATH. For GSM8K, we follow the setup in (Yu et al., 2024), with a decoding temperature of 0.7 and top-k set to 50 for dataset collection. The maximum new token length is set to 400 for GSM8K. In MATH (including OOD settings), we use a decoding temperature of 1, top-p of 0.98, and a maximum new token length of 2000. As token sequences in MATH are long, we apply vllm (Kwon et al., 2023) to accelerate the generation process.

Training OVMs OVMs are initialized from the corresponding generator checkpoints and trained for one epoch, using the same learning rate scheduler as the generator training. The batch size is set to 128 in GSM8K and to 512 in MATH. The optimizer used for training is AdamW.

A.4.2 PRMS

We use the open-source Math-Shepherd dataset (Wang et al., 2024) to train both the generators and PRMs.

Data extraction We extract training problems for each setting. Specifically, for the GSM8K task, we extract all problems from the training split of GSM8K, and for the MATH task, we extract all problems from the training split of MATH. Similar extractions are performed for the OOD settings.

Data preprocessing Since the data format in Math-Shepherd is inconsistent, we normalize the solution paths. We detect steps in each path, normalize them to be split by a newline character, and summarize the final answer in the format of "The answer is xx". For MATH problems, the final answer is enclosed in "\boxed{}".

Training generators For each setting, we randomly select one correct solution for each training problem. If no correct solution is provided, we randomly select one other solution. The training parameters, including the number of epochs, learning schedule, batch size, and optimizer for each base model (Mistral 7B or DeepSeekMath 7B), are the same as those in Appendix A.4.1.

Training PRMs We use all solution paths and annotations provided in Math-Shepherd to train PRMs, which are initialized from the corresponding generator checkpoints and trained for one epoch. Same as above, the batch size is set to 128 in GSM8K and 512 in MATH, and AdamW is used for training. The maximum learning rates for Mistral 7B and DeepSeekMath 7B are 2e-6 and 5e-5, respectively.

We observe that the Math-Shepherd data is noisy and some steps are missing labels. We speculate that this might contribute to its inferior performance compared to OVM in this work.

A.4.3 STEP-LEVEL BEAM SEARCH

In GSM8K, we set the decoding temperature to 0.7, top-k to 50, maximum new token length to 400, and maximum number of steps to 10. In MATH, we set the decoding temperature to 1.0, top-p to 0.98, maximum new token length to 2000, and maximum number of steps to 30. During the beam search process, we prioritize selecting non-duplicate steps. We use vllm in MATH to accelerate token sequence generation.