Every Rollout Counts: Optimal Resource Allocation for Efficient Test-Time Scaling

Xinglin Wang[1], **Yiwei Li**[1], **Shaoxiong Feng**[2†], **Peiwen Yuan**[1], **Yueqi Zhang**[1], **Jiayi Shi**[1], **Chuyi Tan**[1], **Boyuan Pan**[2], **Yao Hu**[2], **Kan Li**[1†]
[1] School of Computer Science, Beijing Institute of Technology
[2] Xiaohongshu Inc
{wangxinglin,liyiwei,peiwenyuan,zhangyq,shijiayi,tanchuyi,likan}@bit.edu.cn
{shaoxiongfeng2023}@gmail.com {panboyuan,xiahou}@xiaohongshu.com

Abstract

Test-Time Scaling (TTS) improves the performance of Large Language Models (LLMs) by using additional inference-time computation to explore multiple reasoning paths through search. Yet how to allocate a fixed rollout budget most effectively during search remains underexplored, often resulting in inefficient use of compute at test time. To bridge this gap, we formulate test-time search as a resource allocation problem and derive the optimal allocation strategy that maximizes the probability of obtaining a correct solution under a fixed rollout budget. Within this formulation, we reveal a core limitation of existing search methods: solution-level allocation tends to favor reasoning directions with more candidates, leading to theoretically suboptimal and inefficient use of compute. To address this, we propose Direction-Oriented Resource Allocation (DORA), a provably optimal method that mitigates this bias by decoupling direction quality from candidate count and allocating resources at the direction level. To demonstrate DORA's effectiveness, we conduct extensive experiments on challenging mathematical reasoning benchmarks including MATH500, AIME2024, and AIME2025. The empirical results show that DORA consistently outperforms strong baselines with comparable computational cost, achieving state-of-the-art accuracy. We hope our findings contribute to a broader understanding of optimal TTS for LLMs.[1]

1 Introduction

As the challenges of scaling up computation and data resources for pretraining continue to grow, scaling test-time computation has emerged as a critical paradigm for enhancing model performance (Brown et al., 2024; Snell et al., 2024; Wu et al., 2025a). By allocating additional computation at inference time, Test-Time Scaling (TTS) improves the performance of LLMs on complex tasks such as mathematical reasoning by enabling deeper exploration of possible solutions (Qwen Team, 2024; Kimi Team et al., 2025; DeepSeek-AI et al., 2025). One prominent approach to scaling test-time computation is through search, where diverse candidate solutions are proposed and filtered using a Process Reward Model (PRM) to guide the procedure (Chen et al., 2024b; Snell et al., 2024; Beeching et al., 2024; Wu et al., 2025a; Liu et al., 2025). By pruning low-quality paths early and focusing computation on more promising ones, these strategies help steer the search process toward trajectories that are more likely to yield correct answers (Setlur et al., 2025b).

While these strategies yield promising performance gains, the question of how to optimally allocate a fixed rollout budget across competing candidate trajectories remains underexplored. In practice,

[†]Corresponding author.

[1]Our code and data have been released on https://github.com/WangXinglin/DORA.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

 https://doi.org/10.52202/085713-3422

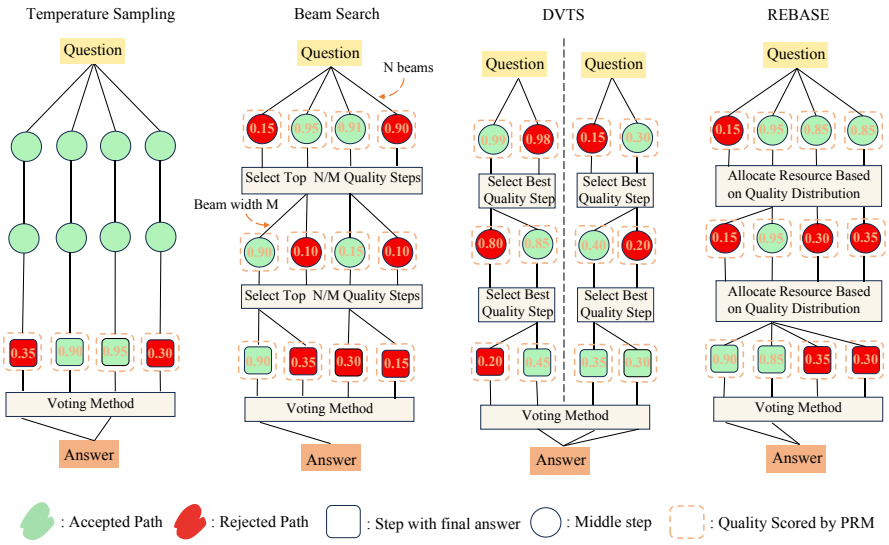

Figure 1: Comparison of different parallel Test-Time search strategies.

existing strategies rely on human-designed heuristics (Figure 1): preserving certain number of high-quality candidates (Beam Search) (Snell et al., 2024), promoting diversity (DVTS) (Beeching et al., 2024), or balancing exploration and exploitation (REBASE) (Wu et al., 2025a). While these intuitions offer practical value, they lack a principled foundation and do not provide guarantees of optimality, such as maximizing the probability of obtaining a correct solution. As a result, rollout budgets may be allocated inefficiently, limiting the effectiveness of test-time computation.

To bridge this gap, we formulate test-time search as a resource allocation problem, where the goal is to maximize the probability of obtaining a correct solution under a fixed rollout budget (Section 3.1). Based on this formulation, we derive the theoretical form of the optimal allocation strategy and revisit existing search methods through a unified lens. We show that, under the assumption that candidate solutions are independent, several widely used strategies approximate the optimal allocation corresponding to different assumptions about the reliability of the reward estimates. However, this independence assumption does not hold in practice, as many candidates share the same underlying reasoning direction (Bi et al., 2024; Hooper et al., 2025). Our theoretical analysis further shows that solution-level allocation is suboptimal: it conflates direction quality with candidate count, biasing the allocation toward overrepresented directions and leading to inefficient use of test-time compute (Section 3.2).

To address this issue, we propose Direction-Oriented Resource Allocation (DORA), a provably optimal method that corrects for this allocating bias by decoupling direction quality from candidate count and allocating resources at the direction level. To validate the effectiveness of DORA, we evaluate it on the challenging mathematical benchmarks MATH500 (Hendrycks et al., 2021), AIME2024 (AI-MO, 2024), and AIME2025 across a broad range of rollout budgets and policy models. The empirical results show that DORA consistently outperforms strong baseline strategies under comparable computational budgets, highlighting its ability to improve the effectiveness of each rollout and enhance the overall efficiency of TTS.

2 Setup & Preliminaries

2.1 Problem Formulation

We formulate the parallel search process under a unified framework, defined by the tuple (π, Q, O, V, N), where $\pi(a \mid \tau)$ is a policy model that generates an action a (reasoning step) given a partial solution $\tau = (x, a_1, \ldots, a_i)$, where x denotes the input problem; $Q : \tau \mapsto [0, 1]$ is the Process Reward Model (PRM), which scores the quality of a partial or complete solution; $O : \mathbb{R}^N \to \mathbb{N}_+^N$ is the resource allocation strategy, dynamically assigning computational budget based on solution

scores; V is the voting method that aggregates final answers from completed solutions to select the most likely correct final answer (e.g., via majority voting, best-of-N, or weighted best-of-N); and N is the total rollout budget of parallel explorations.

The parallel search process can be summarized as Algorithm 1. Specifically, the process iteratively expands a set of partial solutions using the policy π, collects complete solutions, and redistributes the rollout budget via the allocation strategy O based on intermediate rewards from Q. Once sufficient complete solutions are gathered, the final answer is selected using the voting method V.

2.2 Parallel Search Method

We consider four parallel TTS methods which are popularly used in practice: Temperature Sampling (Brown et al., 2024), Beam Search (Snell et al., 2024), Diverse Verifier Tree Search (DVTS) (Beeching et al., 2024), and Reward Balanced Search (REBASE) (Wu et al., 2025a). As pointed out by Snell et al. (2024), lookahead search is inefficient due to sequential sampling, so we do not include it or other methods involving lookahead operations, such as Monte Carlo Tree Search (MCTS).

Based on the unified framework above, we now analyze these strategies from the perspective of resource allocation. While sharing the same overall structure, they differ solely in their choice of allocation function $O(\boldsymbol{R})$, which determines how the total rollout budget N is distributed across candidate solutions based on their PRM scores. We denote the number of rollouts assigned to the i-th candidate τ_j as $O(\boldsymbol{R})_i$, where O is the allocation function and $\boldsymbol{R} = \{R_1, \ldots, R_k\}$ is the vector of PRM scores.

Temperature Sampling. This method performs sampling purely from the policy model, without using reward information for rollout allocation. All candidates are treated equally, and each receives one rollout. External reward signals may still be used at the final answer selection stage, e.g., through best-of-N or weighted best-of-N voting.

$$O_{\text{Temp}}(\boldsymbol{R})_i = 1. \tag{1}$$

Beam Search. Beam Search selects the top $K = N/M$ candidates based on their PRM scores, where M is the number of rollouts assigned per candidate (i.e., the beam width). Only the top-K receive any rollout allocation, while the rest are discarded:

$$O_{\text{Beam}}(\boldsymbol{R})_i = \begin{cases} M, & \text{if } i \in \text{Top-}K(\boldsymbol{R}), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

DVTS. To encourage exploration across diverse solution branches, DVTS partitions the k candidates into $K = N/M$ disjoint groups of size M, corresponding to independent subtrees. Within each group, it performs a local Beam Search by selecting the candidate with the highest PRM score and assigning it M rollouts. Only one candidate per group receives any resource, and groups do not share information:

$$O_{\text{DVTS}}(\boldsymbol{R})_i = \begin{cases} M, & \text{if } i = \arg\max_{j \in \mathcal{G}(i)} R_j, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $\mathcal{G}(i)$ denotes the group containing candidate i.

REBASE. Instead of selecting a fixed number of candidates, REBASE distributes the total rollout budget more smoothly based on the relative quality of each candidate to balance exploitation and exploration. It applies a softmax over the PRM scores R_i to compute allocation weights, and assigns rollouts proportionally:

$$O_{\text{REBASE}}(\boldsymbol{R})_i = \text{round}\left(N \cdot w_i\right), \quad \text{where } w_i = \frac{e^{R_i/T_b}}{\sum_j e^{R_j/T_b}}. \tag{4}$$

where T_b is a temperature parameter controlling the sharpness of the allocation.

 https://doi.org/10.52202/085713-3422

3 Optimal Parallel Search for Test-Time Scaling

While previous parallel search methods enable efficient TTS by exploring multiple reasoning paths simultaneously, their effectiveness critically depends on how the fixed compute budget (i.e., number of rollouts) is allocated across candidate solutions. We focus on the following question:

> *Given a fixed rollout budget, how should one allocate resources across candidate reasoning paths to maximize performance (i.e., the success rate of achieving a correct solution)?*

We are the first to formulate this problem and study the associated parallel search strategies, setting our work apart from previous parallel search studies (Wu et al., 2025a; Beeching et al., 2024; Jiang et al., 2024). To address this, we introduce a Bayesian probabilistic model of solution correctness, and derive an allocation strategy that maximizes expected success under a rollout budget constraint.

3.1 Theoretical Formulation of Optimal Resource Allocation

We aim to allocate a fixed rollout budget N across k candidate reasoning paths to maximize the probability of solving the problem correctly, i.e., obtaining at least one successful solution. Let $p_i \in [0, 1]$ denote the (unknown) success probability of the i-th candidate τ_i when sampled once.

Assumption 1. *The success events of different candidate solutions are independent.*

Under Assumption 1, the probability of obtaining at least one success under an allocation vector $\boldsymbol{B} = \{B_i\}_{i=1}^k$ is given by:

$$\mathbb{P}(\text{success}) = 1 - \prod_{i=1}^{k} (1 - p_i)^{B_i}. \tag{5}$$

Since the true values of p_i are unknown, we adopt a Bayesian modeling approach to capture the uncertainty in their estimation. In practice, p_i is often approximated using the Process Reward Model (PRM) score $R_i = Q(\tau_i)$ (Wang et al., 2024a; Luo et al., 2024; Wang et al., 2024b; Setlur et al., 2025a; Lee et al., 2025), which serves as a proxy for the probability of correctness. However, these estimates are subject to considerable noise due to imperfections in the policy model, variations in decoding temperature, and inherent sampling randomness. To model this uncertainty explicitly, we treat each p_i as a latent variable and place a Beta prior over it. Specifically, we normalize the PRM score into $w_i \in (0, 1)$, and define:

$$p_i \sim \text{Beta}(\kappa w_i, \kappa(1 - w_i)), \tag{6}$$

where $\kappa > 0$ controls the concentration of the prior around its mean. Larger values of κ correspond to higher confidence in the PRM estimate w_i, while smaller values encode greater uncertainty (see Appendix C for more details).

Our goal is to maximize the probability of obtaining at least one successful solution. Under the Bayesian model, this is equivalent to minimizing the expected joint failure:

$$\min_{\sum B_i = N} \mathbb{E}\left[\prod_{i=1}^{k} (1 - p_i)^{B_i}\right]. \tag{7}$$

This defines a convex optimization problem over the rollout allocation vector $\boldsymbol{B} = \{B_i\}_{i=1}^k$. By applying the Karush-Kuhn-Tucker (KKT) conditions, we characterize the limiting behavior of the optimal allocation (see Appendix B.1 for details of proof):

Proposition 1 (Limiting Behavior of Optimal Allocation)**.** *Let $O^\star(\boldsymbol{w})_i$ denote the optimal rollout allocation for candidate i, where $\boldsymbol{w} = \{w_1, \ldots, w_k\}$ are the normalized PRM scores. Then:*

- *When $\kappa \to 0$, the optimal allocation assigns one rollout to each of the top-$\min(k, N)$ candidates with highest w_i scores:*

$$O^\star(\boldsymbol{w})_i = \begin{cases} 1, & \text{if } i \in \text{Top-}\min(k, N) \text{ of } \boldsymbol{w}, \\ 0, & \text{otherwise}, \end{cases}$$

  with the remaining $N - \min(k, N)$ rollouts arbitrarily assigned.

https://doi.org/10.52202/085713-3422

- *When $\kappa \to \infty$, the optimal allocation converges to a deterministic allocation that assigns all rollouts to the highest-scoring candidate:*

$$O^\star(\boldsymbol{w})_i = \begin{cases} N, & \text{if } i = \arg\max_j w_j, \\ 0, & \text{otherwise.} \end{cases}$$

- *When κ is fixed and finite, the optimal allocation approximately follows a shifted linear rule:*

$$O^\star(\boldsymbol{w})_i \approx (N + k\kappa) \cdot w_i - \kappa.$$

Proposition 1 shows that the optimal allocation strategy evolves continuously with the confidence parameter κ. When $\kappa \to \infty$, the Beta prior becomes highly concentrated around the PRM estimate w_i, reflecting strong confidence in its accuracy. In this case, the optimal solution assigns the entire rollout budget to the top-ranked candidate, effectively recovering Beam Search with beam width $M = N$ (Equation 2) and fully exploiting the highest-scoring path.

Conversely, when $\kappa \to 0$, the Beta prior becomes maximally uncertain, collapsing to a Bernoulli mixture where each candidate has a binary chance of being correct or incorrect, with prior weight w_i. In this setting, relying heavily on any single PRM estimate becomes risky, as the scores provide no meaningful guidance. To mitigate this risk, the optimal strategy spreads the rollout budget across multiple candidates in proportion to their prior likelihoods. This reduces to sampling top candidates according to a multinomial distribution over w_i, a behavior closely aligned with temperature sampling used in stochastic decoding.

When κ is fixed and finite, the optimal allocation takes a smoothed, uncertainty-aware form that interpolates between the two extremes above. Specifically, the rollout budget is approximately distributed according to a shifted linear rule (Proposition 1), which closely matches the REBASE strategy (Equation 4). In this regime, the PRM scores are treated as informative but noisy, and the allocation strategy balances exploration and exploitation accordingly.

In practice, due to sampling noise and imperfections in the policy model, PRM scores carry considerable uncertainty. Consistent with this observation, we find that REBASE, which allocates rollouts in proportion to PRM scores, outperforms alternative strategies across a wide range of tasks (see Figure 3). This supports the relevance of the $\kappa \to 0$ setting, which we adopt as the default throughout the paper. Accordingly, we treat REBASE as the baseline solution-level allocation strategy in all subsequent analysis.

3.2 Suboptimality of Solution-Level Allocation

While REBASE is optimal under the assumption of candidate independence (Assumption 1), this condition often does not hold in practice. In particular, many candidate solutions share the same underlying reasoning direction (Bi et al., 2024; Hooper et al., 2025), forming clusters of highly correlated outputs. The solution-level nature of REBASE leads to skewed allocation when candidate counts are imbalanced across reasoning directions.

To formalize this issue, we group candidate solutions into g reasoning directions. Let direction j contain k_j candidates, all sharing the same PRM score R_j, and let $\mathcal{E}_j$ denote the index set of these candidates.

Under REBASE, rollout allocation is performed at the solution level, which implicitly induces a direction-level allocation according to Eq. 4:

$$B_j^{(\text{solution})} = \sum_{i \in \mathcal{E}_j} N \cdot \frac{e^{R_j}}{\sum_{l=1}^{g} k_l e^{R_l}} = N \cdot \frac{k_j e^{R_j}}{\sum_{l=1}^{g} k_l e^{R_l}}. \tag{8}$$

In contrast, the optimal allocation strategy would treat each reasoning direction as a single unit and assign rollouts in proportion to the softmax over direction-level scores:

$$B_j^{(\text{direction})} = N \cdot \frac{e^{R_j}}{\sum_{l=1}^{g} e^{R_l}}. \tag{9}$$

By comparing the induced solution-level allocation in Eq. 8 with the optimal direction-level allocation in Eq. 9, we derive the following proposition (see Appendix B.2 for details of proof):

 https://doi.org/10.52202/085713-3422

Proposition 2 (Suboptimality of Solution-Level Allocation)**.** *When Assumption 1 does not hold, the solution-level allocation $B_j^{(solution)}$ is suboptimal: it does not match the optimal direction-level allocation $B_j^{(direction)}$ unless all directions contain the same number of candidate solutions, i.e., $k_j = k$ for all j.*

This result reveals a fundamental limitation of solution-level allocation: it implicitly favors reasoning directions with more candidate solutions (Figure 2). This bias results in inefficient use of the rollout budget, motivating our proposed method: Direction-Oriented Resource Allocation (DORA).

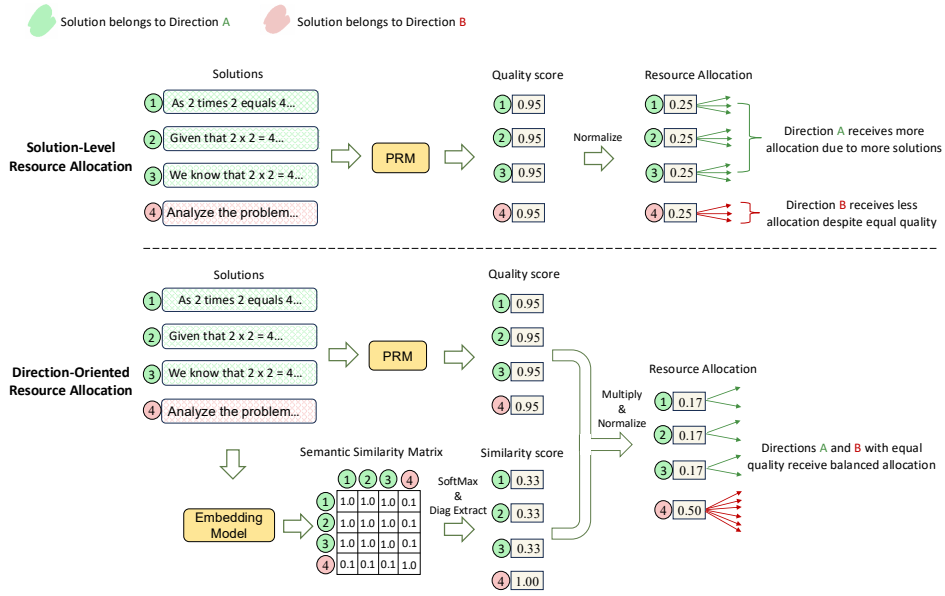

Figure 2: Comparison between Solution-Level Resource Allocation and proposed Direction-Oriented Resource Allocation (DORA).

3.3 Direction-Oriented Resource Allocation (DORA)

To address the bias introduced by solution-level allocation, we propose DORA, a method that adjusts rollout allocation by identifying and correcting for structural redundancy among candidate solutions. As illustrated in Figure 2, DORA incorporates semantic structure into the allocation process by softly clustering solutions into shared reasoning directions and assigning rollouts proportionally at the direction level, rather than treating each solution independently.

Given a set of candidate solutions $\{\tau_1, \ldots, \tau_k\}$, DORA first estimates which solutions share reasoning structure by computing semantic embeddings $\boldsymbol{e}_i \in \mathbb{R}^d$ via a pretrained embedding model. These embeddings are used to construct a cosine similarity matrix $S \in \mathbb{R}^{k \times k}$:

$$S_{ij} = \frac{\boldsymbol{e}_i^\top \boldsymbol{e}_j}{\|\boldsymbol{e}_i\| \cdot \|\boldsymbol{e}_j\|}. \tag{10}$$

To avoid hard clustering and retain flexibility, we interpret the similarity between candidates as a soft assignment over directions. Specifically, we apply a row-wise softmax over S with temperature T_s, yielding an affinity matrix $P \in \mathbb{R}^{k \times k}$:

$$P_{ij} = \frac{e^{S_{ij}/T_s}}{\sum_{j'=1}^{k} e^{S_{ij'}/T_s}}. \tag{11}$$

The diagonal entry $\gamma_i = P_{ii}$ then measures the *semantic uniqueness* of solution τ_i, serving as a proxy for the inverse size of the solution's underlying direction.

Following the REBASE formulation in Eq. 4, we compute normalized quality weights w_i from PRM scores $R_i = Q(\tau_i)$ using a softmax with temperature T_b.

To incorporate semantic structure, we reweight each w_i by its uniqueness:

$$w_i' = \frac{w_i \cdot \gamma_i}{\sum_{j=1}^{k} w_j \cdot \gamma_j}. \tag{12}$$

This downweights redundant solutions and redistributes resources toward distinct reasoning directions.

Finally, rollouts are allocated proportionally:

$$B_i = \text{round}(N \cdot w_i'). \tag{13}$$

DORA balances rollouts across semantically distinct reasoning directions, mitigating the redundancy bias of solution-level methods like REBASE. As summarized in Theorem 1, DORA yields the optimal direction-level allocation under mild assumptions (See Appendix B.3 for the full derivation).

Theorem 1 (Optimality of DORA)**.** *Assume candidate solutions are grouped into g reasoning directions, where direction j consists of candidates indexed by $\mathcal{E}_j$, and all candidates in $\mathcal{E}_j$ share the same PRM score R_j. Then DORA recovers the optimal direction-level rollout allocation specified in Eq. 9.*

4 Experiments

4.1 Experimental Setup

We use Qwen2.5-Math-PRM-7B (Zhang et al., 2025) as our Process Reward Model (PRM) due to its superior reward estimation performance (Zheng et al., 2024; Song et al., 2025). For the policy models, we include Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct (AI, 2024), and Qwen2.5-1.5B-Instruct (Yang et al., 2024), covering a range of model scales and architectures. Considering that existing open-source PRMs are primarily trained on mathematical tasks, we focus our evaluation on three challenging math reasoning benchmarks: MATH500, AIME2024, and AIME2025. To provide a more comprehensive assessment of reasoning performance, we further include four additional math reasoning benchmarks: HMMT24, HMMT25, AMC23, and AMC24, which feature diverse question formats and difficulty distributions. We evaluate models under rollout budgets of 16, 32, 64, 128, and 256 on the main benchmarks. Following Hochlehnert et al. (2025), we repeat all experiments five times on MATH500 and ten times on AIME2024 and AIME2025, reporting the average performance across all runs to reduce the impact of randomness and improve the reliability of our conclusions. For reward assignment during rollouts, we use the final PRM score at each step as the reward for that step. The final answer is selected using weighted majority voting, where each trajectory is weighted by its final PRM score. We use these aggregation strategies since they have been shown to outperform other methods of aggregating trajectories to determine the final response (Beeching et al., 2024). See Appendix E.1 for experimental hyperparameters.

4.2 Main Results

DORA is the most effective parallel search method. As shown in Figure 3 (a) and Table 1, DORA consistently achieves the highest accuracy across all policy models on MATH500, AIME2024, AIME2025, and four additional math reasoning datasets (HMMT24, HMMT25, AMC23, and AMC24). This consistent superiority demonstrates DORA's advantage to make more efficient use of limited test-time compute compared to baseline strategies. To better understand this advantage, we further analyze the pass rate (the number of correct solutions among all sampled rollouts). As shown in Figure 3 (b), DORA consistently reaches more correct solutions than other baselines, highlighting its effectiveness in exploring a broader set of high-quality reasoning paths. Notably, the performance gap between DORA and REBASE widens as the rollout budget increases. We hypothesize that this is due to growing redundancy in sampled solutions: with more rollouts, a larger proportion of trajectories tend to converge to similar final solutions, making REBASE's solution-oriented allocation increasingly prone to overestimating certain reasoning directions. In contrast, DORA mitigates this issue by allocating rollouts at the direction level, allowing for more accurate resource allocation.

4.3 Analysis

DORA is compute-optimal. Considering that DORA introduces an additional semantic similarity step via an embedding model, we examine whether the associated computational overhead is justified

 https://doi.org/10.52202/085713-3422

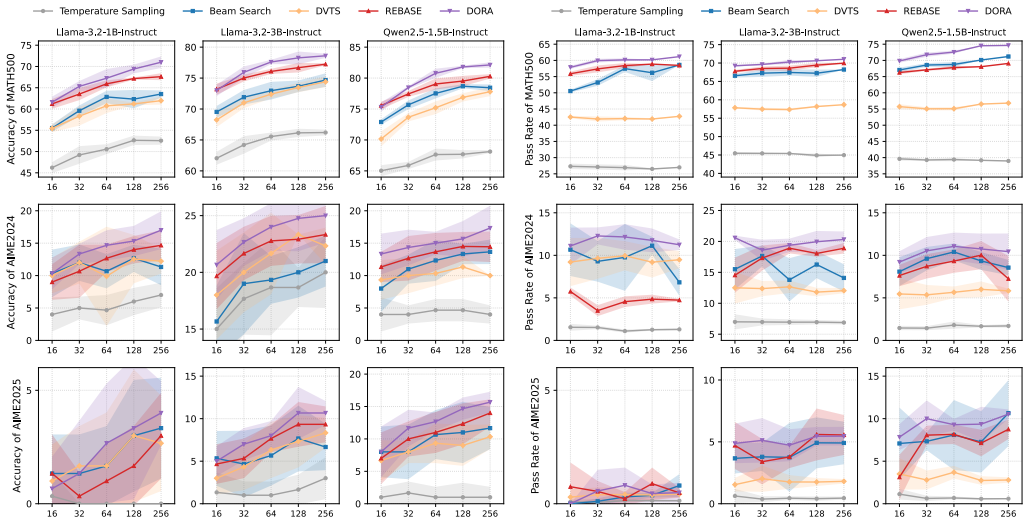

Figure 3: Accuracy and Pass rate comparison under various rollout budgets on MATH500, AIME2024, and AIME2025.

Table 1: Performance comparison on broader math reasoning benchmarks. We set rollout budget as 64, and all results are averaged over five runs.

Policy Model	Method	HMMT24	HMMT25	AMC23	AMC24
LLaMA-3.2-1B-Instruct	Temperature Sampling	0.0	0.0	28.0	13.3
	Beam Search	2.0	0.0	41.5	15.5
	DVTS	1.3	0.0	41.0	18.0
	REBASE	2.0	0.0	45.0	20.0
	DORA	**3.3**	0.0	**45.0**	**20.0**
LLaMA-3.2-3B-Instruct	Temperature Sampling	0.7	0.0	48.5	21.7
	Beam Search	2.0	0.7	47.5	25.3
	DVTS	2.0	0.7	49.5	26.2
	REBASE	4.7	0.0	**59.0**	27.5
	DORA	**6.7**	**2.0**	**59.0**	**31.1**
Qwen2.5-1.5B-Instruct	Temperature Sampling	4.0	0.0	42.0	16.4
	Beam Search	4.7	1.3	53.0	31.1
	DVTS	4.7	2.0	51.5	24.9
	REBASE	5.3	2.7	54.0	32.0
	DORA	**6.7**	**4.0**	**55.0**	**34.2**

by the performance gains. To this end, we follow Snell et al. (2024), comparing the total FLOPs and inference latency of each method, accounting for the computational cost of the policy model, PRM, and embedding model. Table 2 reports both metrics alongside each method's accuracy. The results demonstrate that DORA is substantially more efficient than all baselines. Specifically, compared to the strongest baseline, REBASE at 256 rollouts, DORA achieves higher accuracy using only 64 rollouts, with a 3.5× reduction in total FLOPs and a 4× speedup in inference latency. These findings suggest that DORA achieves stronger performance with substantially less compute, demonstrating its effectiveness as the most efficient test-time search method.

DORA enhances search guidance through semantic clustering. To better understand the effect of DORA's semantic grouping mechanism, we conducted an additional analysis focusing on how well each method improves the intermediate success rate along the reasoning trajectory. Specifically, after each method allocates K reasoning steps, we remove the search algorithm and resume temperature sampling based on the intermediate solutions obtained thus far. We then measure the pass rate of these partial trajectories to estimate the success rate at step K. This reflects the method's ability

Table 2: Comparison of FLOPs and inference latency (s) of different methods on MATH500 and AIME24 using LLaMA-3.2-1B-Instruct. All results are reported as mean (standard deviation) over three runs. The best performance for each metric is highlighted in **bold**. Temperature Sampling is excluded due to its significantly lower accuracy.

Dataset	Method	Rollout	FLOPs				Latency (s)	Accuracy
			Policy Model	PRM	Embedding Model	Total		
MATH500	Beam Search	256	3.58×10^{14}	2.50×10^{15}	0	$2.86(0.03) \times 10^{15}$	345(7)	63.6(0.8)
	DVTS	256	3.79×10^{14}	2.65×10^{15}	0	$3.03(0.03) \times 10^{15}$	253(8)	62.0(0.9)
	REBASE	256	3.88×10^{14}	2.72×10^{15}	0	$3.11(0.03) \times 10^{15}$	490(10)	67.4(0.8)
	DORA	64	8.45×10^{13}	5.92×10^{14}	2.16×10^{14}	$\mathbf{8.92(0.05) \times 10^{14}}$	**124(8)**	**68.7(0.8)**
AIME24	Beam Search	256	6.83×10^{14}	4.99×10^{15}	0	$5.67(0.17) \times 10^{15}$	816(16)	11.3(2.8)
	DVTS	256	4.74×10^{14}	3.52×10^{15}	0	$3.99(0.05) \times 10^{15}$	734(9)	11.6(2.4)
	REBASE	256	6.61×10^{14}	5.01×10^{15}	0	$5.67(0.05) \times 10^{15}$	978(14)	14.7(2.3)
	DORA	64	4.51×10^{14}	9.86×10^{14}	2.82×10^{14}	$\mathbf{1.72(0.19) \times 10^{15}}$	**240(10)**	**14.7(2.3)**

Table 3: Intermediate success rate (%) along the reasoning trajectory on MATH500 with $N{=}64$ rollouts, using LLaMA-3.2-1B-Instruct and Qwen2.5-Math-PRM-7B. All results are averaged over 5 runs.

Step	0	5	10	15	20	25	30	35	40
Beam Search	27.7	39.3	49.6	54.2	55.8	56.8	57.3	57.5	57.5
DVTS	27.7	36.5	40.8	41.7	41.9	41.9	41.9	41.9	42.1
REBASE	27.7	39.5	51.2	54.5	56.5	57.1	58.0	58.3	58.3
DORA	27.7	**40.2**	**51.6**	**55.6**	**57.4**	**58.2**	**59.0**	**59.5**	**59.6**

to guide the policy model toward more promising reasoning directions early on. As shown in Table 3, DORA consistently achieves the highest pass rates across intermediate steps compared to all baselines. Notably, Step 0 corresponds to the Temperature Sampling baseline (i.e., without any search intervention). Comparing this with the improvements achieved by REBASE and DORA highlights the value of clustering: while both methods significantly outperform the baseline, DORA consistently maintains a lead, suggesting that its semantic clustering mechanism not only reduces redundancy but also enhances the effectiveness of search guidance. We will include this result and analysis in the revised version.

5 Related Work

LLM Test-Time Scaling. Scaling LLM test-time compute is an effective way to improve performance (OpenAI, 2024). Prior work has explored various strategies, including sampling-based methods with majority voting (Wang et al., 2023) and search-based techniques (Xie et al., 2023; Khanov et al., 2024; Wan et al., 2024). More recently, search algorithms such as breadth-first and depth-first search (Yao et al., 2023), and Monte Carlo Tree Search (MCTS) (Ma et al., 2023; Li et al., 2022; Liu et al., 2023; Choi et al., 2023) have been applied to enhance reasoning. While these methods show promise, many rely on multi-step lookahead operations that are computationally expensive and limit practical scalability (Snell et al., 2024). To improve efficiency, several studies have proposed parallel search strategies (Snell et al., 2024; Beeching et al., 2024; Wu et al., 2025a). Some complementary directions consider branch-and-prune strategies or dynamic decomposition at inference time (Qiu et al., 2024; Light et al., 2025; Li et al., 2024; Wang et al., 2025). However, how to allocate a fixed rollout budget most effectively during search remains underexplored.

Process Reward Models. Process reward models (PRMs) have emerged as a powerful tool for improving the reasoning and problem-solving capabilities of large language models. By assigning rewards to intermediate steps, PRMs enable finer-grained evaluation and more effective guidance for multi-step reasoning. They have been shown effective in selecting low-error reasoning traces and providing reward signals for reinforcement-style optimization (Uesato et al., 2022; Polu and Sutskever, 2020; Gudibande et al., 2023). With their rapid development, benchmarks such as ProcessBench (Zheng et al., 2024) and PRMBench (Song et al., 2025) have been introduced to provide comprehensive evaluation protocols. Zhang et al. (2025) further offer practical guidelines for training and deploying PRMs, releasing some of the strongest open-source PRMs to date, particularly for mathematical reasoning.

https://doi.org/10.52202/085713-3422

Mathematical Reasoning with LLMs. Recent advances have significantly improved LLMs' performance on mathematical tasks, driven by both training-time and test-time techniques. Training-time methods include large-scale pretraining (OpenAI, 2023; Azerbayev et al., 2024; Shao et al., 2024), supervised fine-tuning (Luo et al., 2023; Tang et al., 2024), and self-improvement via self-generated solutions (Zelikman et al., 2022; Gulcehre et al., 2023; Setlur et al., 2024). Test-time approaches leverage CoT prompting (Wei et al., 2022; Zhao et al., 2025), external tools (Gao et al., 2023; Chen et al., 2023), and self-verification (Weng et al., 2023) to enhance reasoning without changing model weights.

6 Conclusions

In this work, we formulate test-time search as a resource allocation problem and derive its optimal solution under a Bayesian framework. Our theoretical analysis offers a unified perspective that explains existing search methods as approximations under varying reward confidence. Furthermore, we find that solution-level allocation favors directions with more candidates and results in suboptimal use of test-time compute. To address this, we propose DORA, a direction-oriented allocation strategy that provably achieves optimality. Extensive experiments on three mathematical reasoning benchmarks demonstrate that DORA consistently improves performance while reducing compute cost. It achieves 3.5$\times$ fewer FLOPs and 4$\times$ lower latency compared to the strongest baseline REBASE. These results highlight DORA's ability to enhance both the effectiveness and efficiency of test-time inference.

Limitations. While our study focuses on scenarios where a process reward model (PRM) is available to evaluate partial trajectories, the underlying framework is not inherently tied to this specific signal. In principle, DORA can incorporate alternative forms of intermediate feedback, such as model confidence or likelihood-based heuristics, extending its applicability beyond PRM-supervised domains. Another limitation is that our theoretical analysis assumes a low-confidence setting, which may not fully capture the dynamics of confidence accumulation during multi-step reasoning. Adapting the allocation strategy to account for increasing confidence over time presents a promising direction for future work.

Acknowledgements

This work is supported by Beijing Natural Science Foundation (No.4222037, L181010).

References

Meta AI. 2024. Llama 3.2: Multilingual instruction-tuned language models. `https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct`. Accessed: 2025-05-14.

AI-MO. 2024. Aime 2024.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. In *International Conference on Learning Representations (ICLR)*.

Edward Beeching, Lewis Tunstall, and Sasha Rush. 2024. Scaling test-time compute with open models.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2024. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei A Zaharia, and James Y Zou. 2024b. Are more llm calls all you need? towards the scaling properties of compound ai systems. *Advances in Neural Information Processing Systems*, 37:45767–45790.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research (TMLR)*.

Sehyun Choi, Tianqing Fang, Zhaowei Wang, and Yangqiu Song. 2023. KCTS: Knowledge-constrained tree search decoding with token-level hallucination detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14035–14053, Singapore. Association for Computational Linguistics.

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. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, volume 202, pages 10764–10799.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

 https://doi.org/10.52202/085713-3422

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. 2025. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *arXiv preprint arXiv:2504.07086*.

Coleman Hooper, Sehoon Kim, Suhong Moon, Kerem Dilmen, Monishwaran Maheswaran, Nicholas Lee, Michael W Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. 2025. Ets: Efficient tree search for inference-time scaling. *arXiv preprint arXiv:2502.13575*.

Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, et al. 2024. Technical report: Enhancing llm reasoning with reward-guided tree search. *arXiv preprint arXiv:2411.11694*.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. 2024. ARGS: Alignment as reward-guided search. In *International Conference on Learning Representations (ICLR)*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Jung Hyun Lee, June Yong Yang, Byeongho Heo, Dongyoon Han, Kyungsu Kim, Eunho Yang, and Kang Min Yoo. 2025. Token-supervised value models for enhancing mathematical problem-solving capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022. Making large language models better reasoners with step-aware verifier. *arXiv preprint arXiv:2206.02336*.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *ICLR*.

Jonathan Light, Wei Cheng, Benjamin Riviere, Wu Yue, Masafumi Oyamada, Mengdi Wang, Yisong Yue, Santiago Paternain, and Haifeng Chen. 2025. Disc: Dynamic decomposition improves llm inference scaling. *arXiv preprint arXiv:2502.16706*.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2023. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding. *arXiv preprint arXiv:2309.15028*.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. 2025. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. 2023. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

OpenAI. 2024. Learning to reason with llms.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Jiahao Qiu, Yifu Lu, Yifan Zeng, Jiacheng Guo, Jiayi Geng, Chenhao Zhu, Xinzhe Juan, Ling Yang, Huazheng Wang, Kaixuan Huang, et al. 2024. Treebon: Enhancing inference-time alignment with speculative tree-search and best-of-n sampling. *arXiv preprint arXiv:2410.16033*.

Qwen Team. 2024. Qwq: Reflect deeply on the boundaries of the unknown.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. 2024. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *arXiv preprint arXiv:2406.14532*.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025a. Rewarding progress: Scaling automated process verifiers for llm reasoning. In *The Thirteenth International Conference on Learning Representations*.

Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. 2025b. Scaling test-time compute without verification or rl is suboptimal. *arXiv preprint arXiv:2502.12118*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. Prmbench: A fine-grained and challenging benchmark for process-level reward models. *arXiv preprint arXiv:2501.03124*.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. MathScale: Scaling instruction tuning for mathematical reasoning. In *International Conference on Machine Learning (ICML)*, volume 235, pages 47885–47900.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus Mcaleer, Ying Wen, Weinan Zhang, and Jun Wang. 2024. AlphaZero-like tree-search can guide large language model decoding and training. In *International Conference on Machine Learning (ICML)*, volume 235, pages 49890–49920.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024a. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2025. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6904–6917.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zihan Wang, Yunxuan Li, Yuexin Wu, Liangchen Luo, Le Hou, Hongkun Yu, and Jingbo Shang. 2024b. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7309–7319.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

 https://doi.org/10.52202/085713-3422

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2025a. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference on Learning Representations*.

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025b. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 41618–41650.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 11809–11822.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 15476–15488.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*.

Shangziqi Zhao, Jiahao Yuan, Guisong Yang, and Usman Naseem. 2025. Can pruning improve reasoning? revisiting long-cot compression with capability in mind for better reasoning. *arXiv preprint arXiv:2505.14582*.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

A Details of Parallel Search Process

We present the detailed procedure of the Parallel Search Process in Algorithm 1.

Algorithm 1 Parallel Search Process

Require: Input problem $x \sim X$, parameters (π, Q, O, V, N), step limit $T_{\max}$
Ensure: Final Answer

$A_0 \leftarrow \{\tau_j = x\}_{j=1}^{N}$ ▷ Initial active partial solutions
$T_{\text{final}} \leftarrow \emptyset$ ▷ Collected complete solutions
for $i = 0$ to $T_{\max} - 1$ **do**
  for all $\tau_j \in A_i$ **do**
    Sample action $a \sim \pi(\cdot \mid \tau_j)$
    $\tau_j \leftarrow \tau_j \circ a$
  end for
  $T_{\text{final}} \leftarrow T_{\text{final}} \cup \{\tau_j \in A_i \mid \texttt{<EOS>} \in \tau_j\}$ ▷ Add completed solutions
  $A_i \leftarrow A_i \setminus \{\tau_j \in A_i \mid \texttt{<EOS>} \in \tau_j\}$ ▷ Remove completed solutions
  if $|T_{\text{final}}| \geq N$ **then**
    break
  end if
  Compute PRM scores: $R_j \leftarrow Q(\tau_j)$ for each $\tau_j \in A_i$
  Compute rollout allocation: $B_j \leftarrow O(\boldsymbol{R})_j$, where $\boldsymbol{R} = \{R_1, \ldots, R_{|A_i|}\}$
  $A_{i+1} \leftarrow \emptyset$
  for $j = 1$ to $|A_i|$ **do**
    Add B_j copies of τ_j to A_{i+1}
  end for
end for
return $V(T_{\text{final}})$ ▷ Select final answer from complete solutions

B Proof Section

B.1 Proof of Proposition 1

Let $p_i \sim \text{Beta}(\kappa w_i, \kappa(1 - w_i))$, where $w_i \in (0, 1)$ is the normalized PRM score for candidate τ_i. Allocating B_i rollouts to candidate i, the expected failure probability is

$$\mathbb{E}\left[\prod_{i=1}^{k}(1 - p_i)^{B_i}\right] = \prod_{i=1}^{k}\mathbb{E}\left[(1 - p_i)^{B_i}\right].$$

Using the identity for Beta-distributed p_i, we have:

$$\mathbb{E}\left[(1 - p_i)^{B_i}\right] = \prod_{r=0}^{B_i-1}\frac{\kappa(1 - w_i) + r}{\kappa + r}.$$

Taking the negative logarithm of the success probability, the equivalent optimization problem becomes:

$$\min_{\sum B_i = N}\sum_{i=1}^{k}\sum_{r=0}^{B_i-1} -\log\left(1 - \frac{\kappa w_i}{\kappa + r}\right).$$

Using the identity $\sum_{r=0}^{n-1}\frac{1}{\kappa+r} = \psi(\kappa + n) - \psi(\kappa)$, where ψ is the digamma function, the objective simplifies to:

$$L(\boldsymbol{B}) = \sum_{i=1}^{k}\kappa w_i\left[\psi(\kappa + B_i) - \psi(\kappa)\right].$$

Relaxing $B_i \in \mathbb{N}$ to $B_i \in \mathbb{R}_{\geq 0}$, we apply the method of Lagrange multipliers with constraint $\sum_i B_i = N$. The partial derivatives yield:

$$\frac{\partial L}{\partial B_i} = \kappa w_i \cdot \psi'(\kappa + B_i),$$

https://doi.org/10.52202/085713-3422

where ψ' is the trigamma function. The KKT condition implies that at optimality:

$$\kappa w_i \cdot \psi'(\kappa + B_i) = \lambda, \quad \text{for all } i \text{ with } B_i > 0, \quad \text{and} \quad \sum B_i = N.$$

We now analyze three asymptotic regimes of κ:

Case 1: Fixed finite $\kappa > 0$ Using the approximation $\psi'(\kappa + B_i) \approx \frac{1}{\kappa + B_i}$ when $\kappa + B_i \gg 1$, the optimality condition becomes:

$$\frac{\kappa w_i}{\kappa + B_i} \approx \lambda \quad \Rightarrow \quad B_i^\star \approx \frac{\kappa w_i}{\lambda} - \kappa.$$

Summing both sides over i and enforcing $\sum_i B_i = N$, we solve for $\lambda \approx \kappa/(N + k\kappa)$, yielding:

$$\boxed{B_i^\star \approx (N + k\kappa)w_i - \kappa.}$$

Case 2: $\kappa \to \infty$ In this regime, the Beta prior becomes increasingly concentrated at $p_i = w_i$. Hence,

$$\mathbb{E}[(1 - p_i)^{B_i}] \to (1 - w_i)^{B_i}, \quad \text{and} \quad \mathbb{E}\left[\prod_{i=1}^{k}(1 - p_i)^{B_i}\right] \to \prod_{i=1}^{k}(1 - w_i)^{B_i}.$$

To minimize failure probability, we solve:

$$\min_{\sum B_i = N} \sum_{i=1}^{k} B_i \log(1 - w_i).$$

Since $\log(1 - w_i) < 0$, this is minimized by allocating all rollouts to the candidate with the largest w_i, i.e.,

$$O^\star(\boldsymbol{w})_i = \begin{cases} N, & \text{if } i = \arg\max_j w_j, \\ 0, & \text{otherwise.} \end{cases}$$

Case 3: $\kappa \to 0$ In this regime, the Beta distribution becomes highly uncertain:

$$p_i \sim \text{Beta}(\kappa w_i, \kappa(1 - w_i)) \xrightarrow[\kappa \to 0]{} \begin{cases} 1, & \text{with probability } w_i, \\ 0, & \text{with probability } 1 - w_i. \end{cases}$$

Hence,

$$\mathbb{E}\left[(1 - p_i)^{B_i}\right] \to \begin{cases} 1 - w_i, & \text{if } B_i > 0, \\ 1, & \text{if } B_i = 0. \end{cases}$$

Thus, the expected failure probability becomes:

$$\prod_{i=1}^{k} \mathbb{E}\left[(1 - p_i)^{B_i}\right] \to \prod_{i:B_i > 0} (1 - w_i),$$

which depends only on whether a candidate receives at least one rollout, not how many. To minimize failure, we must select a subset $S \subseteq \{1, \ldots, k\}$ with $|S| \leq N$ such that:

$$\prod_{i \in S}(1 - w_i)$$

is minimized. This is achieved by choosing the top-$s = \min(k, N)$ candidates with the largest w_i. Then the optimal allocation is:

$$O^\star(\boldsymbol{w})_i = \begin{cases} 1, & \text{if } i \in \text{Top-}s \text{ of } w_i, \\ 0, & \text{otherwise,} \end{cases} \quad \text{with remaining } N - s \text{ rollouts arbitrarily assigned.}$$

B.2 Proof of Proposition 2

In the $\kappa \to 0$ regime, Proposition 1 shows that the expected success probability is maximized by the solution:

$$B_i \propto w_i, \quad \text{where } w_i = \frac{e^{R_i}}{\sum_j e^{R_j}}.$$

This corresponds to maximizing the log-utility objective:

$$\mathcal{L} = \sum_{i=1}^{k} w_i \log B_i.$$

To analyze the effect of structural redundancy, we group candidate solutions into g reasoning directions. Let direction j contain k_j candidates, each with identical score R_j, and index set $\mathcal{E}_j$.

The optimal direction-aware allocation follows:

$$Q_j := \frac{e^{R_j}}{\sum_{l=1}^{g} e^{R_l}}, \qquad B_j^{(\text{direction})} := N \cdot Q_j.$$

The corresponding log-utility is:

$$\mathcal{L}^{(\text{dir})} = \sum_{j=1}^{g} Q_j \log B_j^{(\text{direction})} = \log N + \sum_{j=1}^{g} Q_j \log Q_j.$$

REBASE assigns each candidate $i \in \mathcal{E}_j$ rollout weight:

$$w_i = \frac{e^{R_j}}{\sum_{l=1}^{g} k_l e^{R_l}}, \quad \text{so} \quad B_j^{(\text{solution})} = \sum_{i \in \mathcal{E}_j} N w_i = N \cdot \frac{k_j e^{R_j}}{\sum_{l=1}^{g} k_l e^{R_l}}.$$

This induces a direction-level distribution:

$$\hat{Q}_j := \frac{k_j e^{R_j}}{\sum_{l=1}^{g} k_l e^{R_l}}.$$

The resulting utility is:

$$\mathcal{L}^{(\text{sol})} = \sum_{j=1}^{g} Q_j \log B_j^{(\text{solution})} = \log N + \sum_{j=1}^{g} Q_j \log \hat{Q}_j.$$

The gap in log-utility is:

$$\mathcal{L}^{(\text{dir})} - \mathcal{L}^{(\text{sol})} = \sum_{j=1}^{g} Q_j \log \frac{Q_j}{\hat{Q}_j} = \mathrm{KL}(Q \parallel \hat{Q}) \geq 0.$$

Equality holds if and only if $Q_j = \hat{Q}_j$ for all j, i.e.,

$$\frac{e^{R_j}}{\sum_l e^{R_l}} = \frac{k_j e^{R_j}}{\sum_l k_l e^{R_l}} \quad \Rightarrow \quad k_j = k \text{ for all } j.$$

Thus, the solution-level allocation is suboptimal unless all reasoning directions contain the same number of candidate solutions.

B.3 Proof of Theorem 1

Assume candidate solutions are partitioned into g reasoning directions, where direction $j \in \{1, \ldots, g\}$ contains k_j candidates indexed by $\mathcal{E}_j$, and all candidates in $\mathcal{E}_j$ share the same PRM score R_j.

Under REBASE, softmax is computed at the solution level:

$$\tilde{q}_i = \frac{e^{R_j}}{\sum_{l=1}^{g} k_l e^{R_l}}, \quad \text{for } i \in \mathcal{E}_j.$$

https://doi.org/10.52202/085713-3422

Aggregating across each direction yields the induced direction-level distribution:

$$\hat{Q}_j^{\text{REBASE}} = \sum_{i \in \mathcal{E}_j} \tilde{q}_i = \frac{k_j e^{R_j}}{\sum_{l=1}^{g} k_l e^{R_l}}.$$

To eliminate the bias from uneven candidate counts k_j, DORA reweights each $\tilde{q}_i$ by the inverse of its cluster size:

$$\hat{q}_i = \frac{\tilde{q}_i}{k_j}, \quad \text{for } i \in \mathcal{E}_j.$$

The normalization constant becomes:

$$Z = \sum_{i=1}^{k} \hat{q}_i = \sum_{j=1}^{g} \sum_{i \in \mathcal{E}_j} \frac{\tilde{q}_i}{k_j} = \sum_{j=1}^{g} \frac{k_j e^{R_j}}{k_j \sum_{l=1}^{g} k_l e^{R_l}} = \frac{\sum_{j=1}^{g} e^{R_j}}{\sum_{l=1}^{g} k_l e^{R_l}}.$$

Normalizing gives the final corrected weight:

$$\hat{q}_i^{\text{final}} = \frac{\hat{q}_i}{Z} = \frac{e^{R_j}}{k_j \sum_{l=1}^{g} e^{R_l}}, \quad \text{for } i \in \mathcal{E}_j.$$

Aggregating over direction j, the direction-level allocation becomes:

$$\hat{Q}_j^{\text{final}} = \sum_{i \in \mathcal{E}_j} \hat{q}_i^{\text{final}} = k_j \cdot \frac{e^{R_j}}{k_j \sum_{l=1}^{g} e^{R_l}} = \frac{e^{R_j}}{\sum_{l=1}^{g} e^{R_l}} = Q_j.$$

Thus, the final allocation satisfies

$$\sum_{i \in \mathcal{E}_j} B_i \propto Q_j,$$

which exactly matches the optimal direction-level allocation given in Eq. 9.

C Details of Beta Distribution

The Beta distribution is a standard choice for modeling random variables on the unit interval, and its parameters $(\alpha, \beta) = (\kappa w_i, \kappa(1 - w_i))$ are interpretable: the mean is $\mathbb{E}[p_i] = w_i$, and the variance is inversely related to κ. Specifically:

- When κ is small, the distribution is diffuse and uncertain.
- When κ is large, the distribution is sharply peaked around w_i, indicating high confidence.

Figure 4 visualizes the effect of different κ values with w_i fixed at 0.7.

D More Experiments

D.1 DORA provides larger gains on harder problems

Figure 5 shows that while DORA remains the top-performing strategy across the entire MATH500 benchmark, the size of its advantage depends sharply on difficulty. On easier Level 1–2 problems, most methods perform well given moderate rollout budgets, so the accuracy curves for all methods converge closely. On the other hand, on harder Level 3–5 problems, the gap between DORA and solution-level methods widens steadily with budget, with DORA achieving a clear lead at higher rollout levels. We hypothesize that harder problems amplify DORA's strength as they typically require longer reasoning chains (Wu et al., 2025b), which allows more opportunities for rollout allocation across search steps. As the number of allocation rounds increases, a principled strategy like DORA could compound its advantage by continually prioritizing promising directions and avoiding wasted computation.

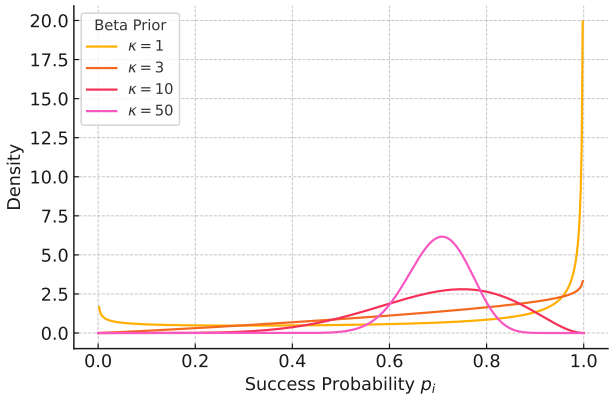

Figure 4: Effect of the concentration parameter κ on the Beta prior. All curves are plotted with fixed mean $w_i = 0.7$. Larger κ yields a more concentrated prior around w_i, while smaller κ reflects greater uncertainty.

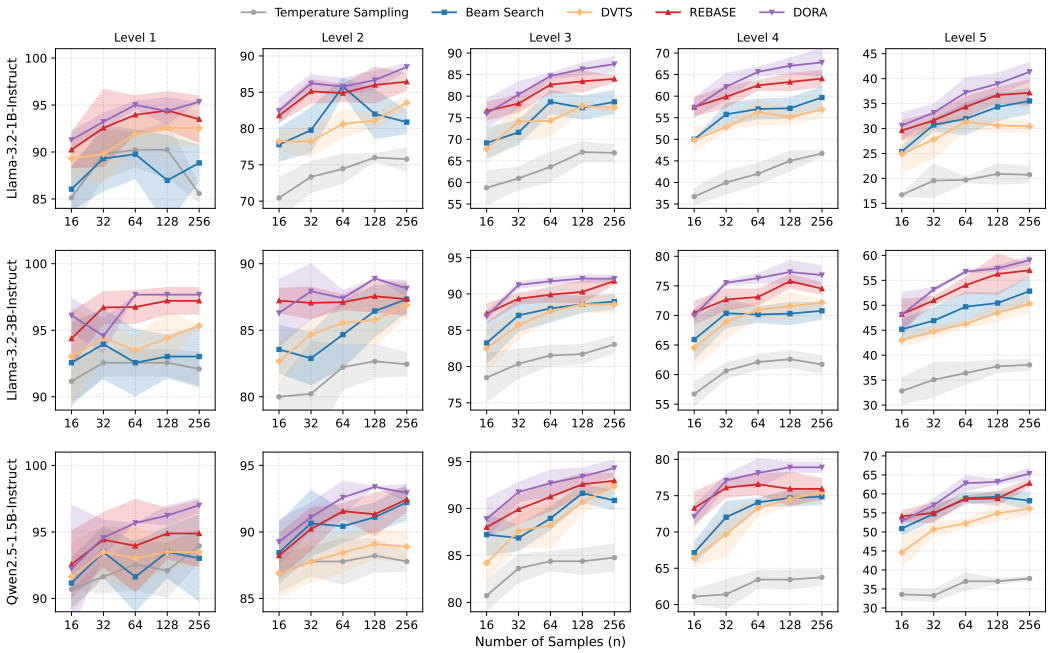

Figure 5: Comparison of method accuracy on MATH500 across different difficulty levels.

D.2 Ablation on Clustering Design

To better isolate the effect of DORA's semantic clustering mechanism, we conduct an ablation study comparing several alternative clustering strategies before finalizing the soft clustering design.

KMeans clustering. KMeans requires predefining the number of clusters K. However, the actual number of distinct reasoning directions can vary substantially across problems and reasoning steps, making a fixed K a poor approximation and often leading to suboptimal clustering performance.

Hierarchical clustering. We experiment with hierarchical clustering, which avoids setting K by merging solutions based on a fixed cosine similarity threshold C. However, the optimal value of C should ideally vary with the depth and complexity of reasoning, which differ significantly across problems. This sensitivity made hierarchical clustering even less effective than KMeans in our experiments.

https://doi.org/10.52202/085713-3422

Table 4: Ablation study of different clustering methods on MATH500 with $N=64$ rollouts, using LLaMA-3.2-1B-Instruct. Results are averaged over 5 runs.

Cluster Method	K=3	K=5	K=10	C=0.95	C=0.90	C=0.80	Soft (default)
Accuracy	67.6	68.0	67.8	67.0	66.6	66.0	**68.7**

Soft clustering (ours). Based on these findings, we adopted a soft clustering approach in DORA, which avoids hard assignment boundaries and allows for more adaptive and robust semantic grouping across diverse reasoning structures.

As shown in Table 4, the soft clustering variant achieves the best performance, outperforming all hard clustering alternatives. This demonstrates that allowing soft, overlapping group assignments enables DORA to better adapt to the variable structure of reasoning paths, improving robustness and accuracy across diverse problem types.

D.3 DORA generalizes across model scales and PRM families

As shown in Table 5, we further test DORA with a larger policy–PRM pair: LLaMA-3.1-8B-Instruct guided by Qwen2.5-Math-PRM-72B on MATH500 with $N=64$ rollouts. DORA attains the highest accuracy (80.6), outperforming Beam Search (78.2), DVTS (78.4), and REBASE (79.2). We also replace the reward model with a different PRM family (Skywork-PRM-7B) and observe consistent gains across all policy models (Table 6), surpassing the strongest baseline in each case. These results indicate that DORA's advantages persist when scaling to larger models and when switching across distinct reward-model families, underscoring its robustness as a reliable test-time search strategy.

Table 5: Accuracy on MATH500 with a larger policy model and PRM: LLaMA-3.1-8B-Instruct as the policy and Qwen2.5-Math-PRM-72B as the PRM ($N=64$ rollouts). Best results are in **bold**.

PRM	Policy Model	Temperature Sampling	Beam Search	DVTS	REBASE	DORA
Qwen2.5-Math-PRM-72B	LLaMA-3.1-8B-Instruct	70.5	78.2	78.4	79.2	**80.6**

Table 6: Accuracy on MATH500 with an alternative PRM family (Skywork-PRM-7B) at $N=64$ rollouts. Best results are in **bold**.

Method	LLaMA-3.2-1B-Instruct	LLaMA-3.2-3B-Instruct	Qwen2.5-1.5B-Instruct
Temperature Sampling	58.5	69.5	73.0
Beam Search	69.0	75.5	79.7
DVTS	66.2	74.2	78.9
REBASE	70.8	76.2	80.2
DORA	**71.8**	**76.6**	**81.0**

D.4 DORA is robust to hyperparameter choices

We further study the temperature parameters T_b (softmax over directions) and T_s (semantic similarity) on MATH500 with $N=64$ (Table 7). Both REBASE and DORA perform well at $T_b \in \{0.01, 0.1\}$ but degrade at $T_b=1.0$, where the softened distribution weakens PRM guidance and approaches unguided sampling. Importantly, DORA is stable across a wide range of T_s values (0.001–1.0), indicating low sensitivity to the clustering threshold. To further investigate DORA's robustness across retriever families, we compared our default retriever (bge-m3, 568M parameters) with two popular alternatives from the MTEB leaderboard (Muennighoff et al., 2023) that support long inputs (2048 tokens): e5-base-4k (110M) and gte-multilingual-base (305M). As shown in Table 8, BGE and GTE deliver comparable performance across all policy models, while E5 is slightly worse—likely due to its smaller capacity and thus weaker clustering in high-dimensional embedding space. Together, these results show that DORA's gains are not brittle: it maintains strong accuracy under reasonable choices of temperatures and retrievers, reinforcing its practicality for real-world deployment.

Table 7: Sensitivity analysis of temperature hyperparameters T_b and T_s on MATH500 with N=64 rollouts using LLaMA-3.2-1B-Instruct. All results are averaged over 5 runs.

Method	T_b			T_s			
	0.01	0.1	1.0	0.001	0.01	0.1	1.0
REBASE	64.8	65.9	55.4	-	-	-	-
DORA	67.4	68.7	57.2	67.8	68.7	68.0	67.5

Table 8: Ablation on embedding (retriever) models on MATH500 with N=64 rollouts. We compare our default BGE-M3 to GTE-multilingual-base and E5-base-4k. Results are averaged over 5 runs.

Policy Model	BGE	GTE	E5
LLaMA-3.2-1B-Instruct	68.7	68.2	66.8
LLaMA-3.2-3B-Instruct	77.6	77.4	76.8
Qwen-2.5-1.5B-Instruct	80.8	80.2	79.2

E Implementation Details

E.1 Experimental Hyperparameters

All experiments use temperature sampling with `temperature` = 0.8 and `top_p` = 1.0. We set the token limit to 256 per step and 2048 tokens in total for each solution. For Beam Search and DVTS, we use a beam width of 4 following Snell et al. (2024). For REBASE, we set its T_b to 0.1, consistent with its original implementation. For DORA, we employ the open-source BGE-M3 embedding model (Chen et al., 2024a) to compute semantic similarity between trajectories, chosen for its lightweight architecture, strong empirical performance, and ability to handle long input sequences. We set the T_b for quality scores to 0.1 (matching REBASE), and the semantic similarity temperature T_s to 0.01. All experiments are executed in parallel on a cluster with 32 NVIDIA A100 GPUs (40G), where each individual run is allocated to a single GPU.

E.2 Details of Prompt

Following Beeching et al. (2024), we employ the prompt below for LLM mathematical reasoning:

```
Solve the following math problem efficiently and clearly:

- For simple problems (two steps or fewer):
  Provide a concise solution with minimal explanation.

- For complex problems (three steps or more):
  Use this step-by-step format:

## Step 1: [Concise description]
[Brief explanation and calculations]

...
## Step 2: ...

Regardless of problem complexity, always conclude with:
Therefore, the final answer is: \boxed{answer}.
```


https://doi.org/10.52202/085713-3422

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We substantiate our claims with both experimental evidence and theoretical analysis.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of our work in the conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We have included detailed assumptions and proof in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided detailed experimental settings in the experiments section and detailed module designs of our method.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

https://doi.org/10.52202/085713-3422

Answer: [Yes]

Justification: We have provided the code of our work.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided detailed experimental settings in the experiments section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For each setting, we run 5-10 times and report the average results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided corresponding details in Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics and followed it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

 https://doi.org/10.52202/085713-3422

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have correctly cited all the assets we use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No new assets released.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

 https://doi.org/10.52202/085713-3422