DIVTHINK: PARALLEL THINKING VIA DIVERSIFIED CHAIN-OF-THOUGHT SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time reasoning has become a central paradigm for scaling Large Language Models (LLMs), but parallel Chain-of-Thought (CoT) sampling often wastes computation on redundant trajectories that repeat similar intermediate steps and converge to the same final answer. Common alternatives either refine trajectories sequentially, as in MCMC power sampling, or require costly post-training, as in GRPO. To bridge this gap, we propose DIVTHINK, a training-free decoding method that coordinates parallel CoT trajectories through lightweight repulsive logit adjustments. At each step, DIVTHINK downweights tokens already preferred by peer trajectories, encouraging complementary reasoning paths while preserving the base model's local preferences and the throughput advantage of parallel generation. Across mathematical reasoning, code generation, and scientific question-answering benchmarks, DIVTHINK improves accuracy by 7.6 points over Base@8 and 4.0 points over Low-temperature@8 on average, with gains up to 22.5 points, while using far fewer generated tokens than MCMC Power Sampling and avoiding the rollout and optimization cost of GRPO-style post-training.

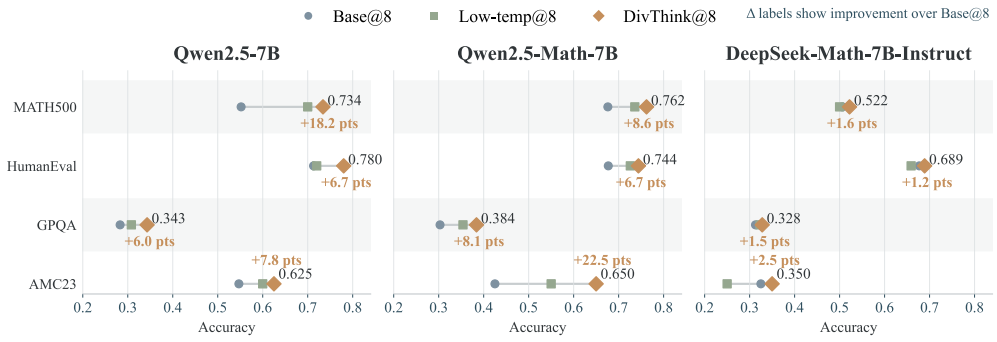

Figure 1: Accuracy comparison across model families and benchmarks. The grouped bars report performance on MATH500, HumanEval, GPQA, and AMC23 for Qwen2.5-Math-7B, Qwen2.5-7B, and DeepSeek-Math-7B-Instruct.

1 INTRODUCTION

Large language models (LLMs) have rapidly evolved into general-purpose reasoning systems capable of solving mathematical problems, generating code, answering scientific questions, and performing multi-step decision making (Bai et al., 2023; Liu et al., 2024; Abdin et al., 2024; Touvron et al., 2023). A central driver of these capabilities is *test-time reasoning*, in which models allocate additional inference computation to construct intermediate reasoning steps before committing to a final answer (Muennighoff et al., 2025; Snell et al., 2024; Yang et al., 2026a). Chain-of-Thought (CoT) prompting instantiates this paradigm by eliciting natural-language rationales, while subsequent methods further improve reasoning by sampling, searching over, or aggregating multiple candidate trajectories (Wei et al., 2022; Wang et al., 2022; Yao et al., 2023; Brown et al., 2024).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

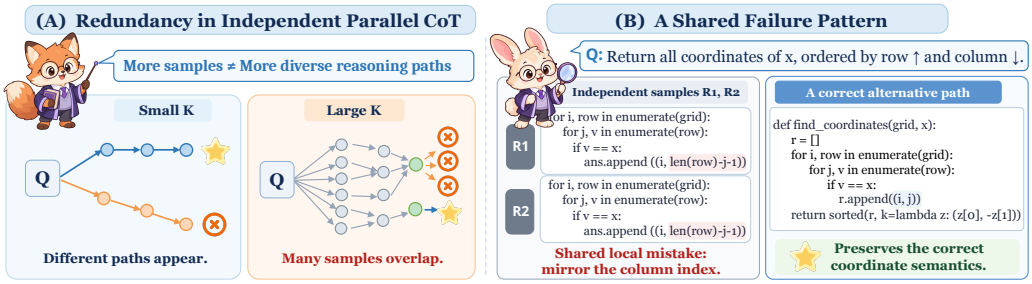

Figure 2: **Redundancy limits the value of parallel reasoning.** **(A)** Increasing the number of independently sampled trajectories does not necessarily increase reasoning-path diversity: as the sampling budget grows, many trajectories can overlap around the same dominant reasoning mode. **(B)** Such redundancy can reproduce a shared local mistake across multiple samples. In the illustrated code-generation example, independent trajectories repeatedly mirror the column index, whereas an alternative reasoning path preserves the original coordinate semantics and yields the correct solution. This motivates coordinating parallel trajectories to encourage complementary reasoning paths rather than repeated variants of the same one.

Recent progress suggests that the effectiveness of test-time reasoning depends not only on how much extra computation is used, but also on where that computation is allocated. GRPO-style methods spend computation before deployment by generating rollouts, computing reward or verifier signals, curating data, and updating the policy (Shao et al., 2024; Guo et al., 2025). MCMC-based power sampling avoids post-training, but uses inference-time computation to iteratively resample and refine individual trajectories (Karan & Du, 2025). Standard parallel CoT sampling is simpler and highly parallelizable, yet it usually draws trajectories independently from the same distribution (Wang et al., 2022; Brown et al., 2024). These approaches illustrate different ways of allocating test-time compute: post-training modifies the policy before deployment, MCMC-based methods sequentially refine individual trajectories, and standard parallel sampling generates multiple attempts independently. This leaves open how to coordinate concurrent trajectories specifically to reduce redundancy while preserving complementary reasoning coverage.

This exposes a bottleneck in parallel test-time reasoning: *redundant reasoning*. Independent trajectories often follow similar intermediate steps and converge to the same final answer, especially when the model distribution is peaked. Such redundancy is harmless when the dominant reasoning path is correct, but it limits the value of additional samples when that path is wrong. In this case, increasing the number of trajectories may mostly repeat the same error rather than explore genuinely complementary solution paths, as illustrated in Figure 2.

A natural response is to increase sampling diversity, but naive diversity is not enough for reasoning. Raising temperature encourages exploration, but can weaken local coherence and introduce noisy continuations; lowering temperature improves confidence, but can further collapse samples toward the same high-probability path (Wang et al., 2022). Effective parallel reasoning therefore calls for a lightweight mechanism that coordinates trajectories during decoding without discarding the base model's next-token preferences. This motivates our central question:

How can we make parallel Chain-of-Thought trajectories diversify during generation, so that multiple attempts explore complementary reasoning paths instead of repeating the same thought?

We answer this question with DIVTHINK, a training-free decoding method for coordinated parallel CoT generation. At each decoding step, DIVTHINK first obtains the base next-token distribution for each trajectory, then applies a repulsive logit adjustment that downweights tokens already strongly preferred by the other trajectories. This soft diversification keeps each trajectory guided by the original model distribution while encouraging trajectories to take different local reasoning directions.

Rather than modifying the policy through post-training, sequentially refining individual trajectories, or leaving parallel attempts independent, DIVTHINK coordinates the parallel trajectory set directly during decoding. The resulting procedure reduces redundancy among simultaneous reasoning attempts while preserving the throughput advantage of parallel generation.

Our contributions are summarized as follows.

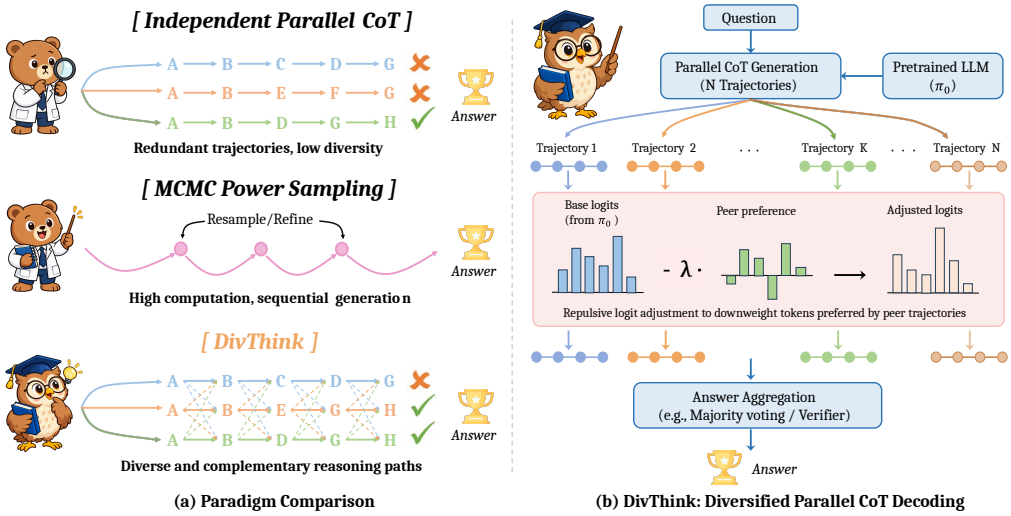

Figure 3: **Overview of DIVTHINK. (a)** Standard parallel sampling generates trajectories independently and can repeatedly follow the same reasoning path, while MCMC Power Sampling broadens exploration through sequential resampling. DIVTHINK instead coordinates the retained trajectories during batched parallel decoding. **(b)** At each step, DIVTHINK aggregates the peer next-token distributions into a normalized repulsion field and subtracts it from each trajectory's base logits. Tokens favored by other trajectories are downweighted through a repulsive logit adjustment, reducing local redundancy while preserving the base model's preferences.

· We identify redundant reasoning as a key bottleneck in parallel CoT generation: additional samples can spend computation repeating the same dominant path rather than expanding candidate-set coverage.

· We propose DIVTHINK, a training-free decoding method that applies lightweight repulsive logit adjustments across trajectories, encouraging complementary reasoning paths while preserving the base model's local preferences and batched parallel generation.

· Across mathematical reasoning, code generation, and scientific question answering on three 7B backbones, DIVTHINK improves accuracy by 7.6 points over Base@8 and 4.0 points over Low-temperature@8 on average, with gains up to 22.5 points, while using substantially fewer generated tokens than MCMC Power Sampling.

2 RELATED WORK

2.1 INFERENCE-TIME SAMPLING

Inference-time sampling has emerged as an alternative to training-time optimization for improving LLM reasoning (Li et al., 2023; Manvi et al., 2024; Kang et al., 2024; Liu et al., 2023). Repeated sampling shows that drawing more candidate solutions can yield predictable coverage gains, particularly when verifiers are available (Brown et al., 2024). Beyond independent sampling, recent work demonstrates that stronger reasoning can be elicited from base models through sharpened likelihood distributions and MCMC-style procedures, without RL, curated data, or external verifiers (Karan & Du, 2025; Ji et al., 2026; Azizi et al., 2026). Analyses of RLVR further suggest that post-training may partly surface reasoning capabilities already present in the base distribution (Shao et al., 2025).

Other approaches improve inference by adapting the decoding distribution or allocating test-time computation more strategically, including contextual temperature control (Wang et al., 2020), search-guided decoding (Feng et al., 2023), and compute-optimal combinations of sampling, search, and revision (Snell et al., 2024).

2.2 PARALLEL THINKING

Parallel thinking has emerged as a test-time scaling paradigm in which language models explore multiple reasoning trajectories rather than extending a single sequential chain (Pan et al., 2025; Fu et al., 2025; Wang et al., 2022). Repeated sampling and self-consistency improve coverage by aggregating multiple candidate solutions (Brown et al., 2024; Wang et al., 2022), but typically treat trajectories as independent draws from the same model distribution. Consequently, parallel samples may remain redundant when probability mass concentrates around a few dominant reasoning modes.

More structured approaches organize parallelism through search, branching, or architectural support. Tree of Thoughts performs deliberative search over intermediate reasoning states (Yao et al., 2023), while Adaptive Parallel Reasoning introduces explicit `spawn()` and `join()` operations (Pan et al., 2025). Recent work further studies parallelism through concurrent generation, native decomposition and synthesis, or parallel streams as an additional scaling axis (Rodionov et al., 2026; Yang et al., 2026b; Wen et al., 2025; Pan et al., 2025).

Concurrent work also couples parallel reasoning trajectories directly during decoding. ThinkMerge averages next-token logits across parallel traces at synchronization points to produce a single coherent output (Wang et al., 2026). In contrast, DIVTHINK uses cross-trajectory interaction repulsively to preserve complementary reasoning paths rather than promote consensus. Additional comparisons with recent related methods are provided in Appendix A.3.

3 METHODOLOGY

Figure 3 presents DIVTHINK, a training-free method that coordinates parallel CoT trajectories through repulsive logit adjustments. At each step, it downweights tokens favored by peer trajectories while retaining the frozen model's logits as the quality signal, reducing redundancy and expanding reasoning coverage.

3.1 PRELIMINARIES

Given an input query q, a frozen language model P_θ generates K parallel Chain-of-Thought (CoT) trajectories $\mathcal{T} = \{T^{(1)}, \ldots, T^{(K)}\}$. At decoding step t, trajectory k has history $h_{t-1}^{(k)}$ and base logits $\mathbf{l}_t^{(k)} \in \mathbb{R}^{|\mathcal{V}|}$ over vocabulary $\mathcal{V}$. We denote its untempered base next-token distribution as

$$\mathbf{p}_t^{(k)} = \mathrm{Softmax}(\mathbf{l}_t^{(k)}). \tag{1}$$

Standard parallel CoT sampling treats trajectories as independent:

$$P(\mathcal{T}) = \prod_{k=1}^{K} P_\theta(T^{(k)}). \tag{2}$$

Although effective for improving pass@k, this independent factorization often produces redundant trajectories when the model distribution is sharply peaked. Our goal is to improve the effective diversity of the sampled trajectory set without training, external verifiers, or additional model calls.

3.2 SET-LEVEL REPULSIVE OBJECTIVE

For $K > 1$, at decoding step t, we denote the next-token choices of all parallel trajectories by $X_t = \{x_t^{(1)}, \ldots, x_t^{(K)}\}$, and their corresponding histories by $H_{t-1} = \{h_{t-1}^{(1)}, \ldots, h_{t-1}^{(K)}\}$. Standard parallel decoding samples these tokens independently under the base model. In contrast, DIVTHINK is motivated by a set-level distribution that balances base-model likelihood with cross-trajectory repulsion:

$$\begin{aligned} \pi(X_t \mid H_{t-1}) \propto \prod_{k=1}^{K} P_\theta(x_t^{(k)} \mid h_{t-1}^{(k)}) \\ \cdot \exp(-\lambda R(X_t)) \,. \end{aligned} \tag{3}$$

Algorithm 1 DIVTHINK Decoding

Input: Query q, model P_θ, K, λ, τ, maximum length T
$h_0^{(1)} = \cdots = h_0^{(K)} = q$; $A_1 = \{1, \ldots, K\}$
for $t = 1, \ldots, T$ **do**
  for $k \in A_t$ **do**
    $\mathbf{l}_t^{(k)} \leftarrow P_\theta(\cdot \mid h_{t-1}^{(k)})$
    $\mathbf{p}_t^{(k)} \leftarrow \mathrm{Softmax}(\mathbf{l}_t^{(k)})$
  end for
  for $k \in A_t$ **do**
    $\mathbf{r}_t^{(k)} \leftarrow \frac{1}{|A_t|-1} \sum_{j \in A_t \setminus \{k\}} \mathbf{p}_t^{(j)}$; set $\mathbf{r}_t^{(k)} = \mathbf{0}$ if $|A_t| = 1$
    $x_t^{(k)} \sim \mathrm{Softmax}\Big((\mathbf{l}_t^{(k)} - \lambda \mathbf{r}_t^{(k)})/\tau\Big)$
    $h_t^{(k)} \leftarrow h_{t-1}^{(k)} \circ x_t^{(k)}$
  end for
  $A_{t+1} \leftarrow \{k \in A_t : x_t^{(k)} \neq \mathrm{EOS}\}$; **break** if $A_{t+1} = \varnothing$
end for
return $\{h^{(1)}, \ldots, h^{(K)}\}$

The redundancy score is defined as

$$R(X_t) = \frac{1}{K-1} \sum_{1 \le i < j \le K} I\Big(x_t^{(i)} \approx x_t^{(j)}\Big), \tag{4}$$

where the basic token-level instantiation uses

$$I\Big(x_t^{(i)} \approx x_t^{(j)}\Big) = \mathbf{1}\Big[x_t^{(i)} = x_t^{(j)}\Big]. \tag{5}$$

The factor $1/(K-1)$ expresses redundancy as an average interaction with each trajectory's peers, matching the per-trajectory repulsion field below. The first factor in Eq. (3) preserves the quality signal from the base model, while the exponential factor downweights configurations in which multiple trajectories choose redundant local continuations. Rather than sampling exactly from Eq. (3), DIVTHINK uses it to derive the lightweight one-step logit update below. Appendix A.1 gives the full target/proposal correction and mean-field derivation.

3.3 SYNCHRONOUS REPULSIVE UPDATE

For each trajectory k, DIVTHINK constructs a **repulsion field** from the next-token distributions of the other trajectories:

$$\mathbf{r}_t^{(k)} = \frac{1}{K-1} \sum_{j \neq k} \mathbf{p}_t^{(j)}. \tag{6}$$

For a candidate token v, the scalar $r_t^{(k)}(v)$ is the mean peer probability of choosing v. It is therefore a soft estimate of how likely trajectory k would collide with the other trajectories if it also chose v. To penalize such collisions, we reweight the base distribution by an exponential cost,

$$\tilde{p}_t^{(k)}(v) \propto p_t^{(k)}(v) \exp\Big(-\lambda r_t^{(k)}(v)\Big). \tag{7}$$

Since $p_t^{(k)}(v) \propto \exp(l_t^{(k)}(v))$, where $l_t^{(k)}(v)$ denotes the v-th component of $\mathbf{l}_t^{(k)}$, Eq. (7) is equivalent to the logit update

$$\hat{\mathbf{l}}_t^{(k)} = \mathbf{l}_t^{(k)} - \lambda \mathbf{r}_t^{(k)}. \tag{8}$$

The subtraction is thus intentional: tokens preferred by peer trajectories receive larger costs and lower adjusted logits, whereas adding $\lambda \mathbf{r}_t^{(k)}$ would encourage agreement and increase redundancy.

Here, $\lambda \ge 0$ controls the trade-off between staying close to the base model and encouraging diversity among reasoning paths: larger values more strongly penalize tokens preferred by peer trajectories. During decoding, the same averaging is applied over the currently active peers; if only one trajectory remains active, we set its repulsion field to zero. The detailed derivation is provided in Appendix A.1.

Table 1: **DIVTHINK matches or outperforms strong inference-time and post-training baselines across model families and tasks.** We report accuracy and relative token usage on MATH500, HumanEval, GPQA, and AMC23. Bold entries mark GRPO results and the best-performing training-free method within each model block, and underlining indicates cases where DIVTHINK outperforms GRPO. Token multipliers denote the total number of generated tokens relative to standard single-sample inference; for MCMC Power Sampling, this includes all tokens generated during iterative resampling.

	MATH500		HumanEval		GPQA		AMC23	
	Accuracy	#Token	Accuracy	#Token	Accuracy	#Token	Accuracy	#Token
Qwen2.5-Math-7B								
Base@1	0.496	1.0×	0.329	1.0×	0.278	1.0×	0.300	1.0×
Low-temperature@1	0.690	0.9×	0.512	1.2×	0.353	1.0×	0.500	1.0×
MCMC Power Sampling@1	0.748	20.7×	0.573	8.5×	**0.389**	15.8×	0.625	51.1×
Base@8	0.676	8.0×	0.677	8.9×	0.303	8.0×	0.425	7.1×
Low-temperature@8	0.736	7.4×	0.726	10.3×	0.354	9.0×	0.550	6.8×
DIVTHINK@8 (ours)	**0.762**	7.2×	**0.744**	9.2×	0.384	8.8×	**0.650**	6.6×
GRPO (MATH)@1	**0.785**	1.0×	**0.537**	1.1×	**0.399**	1.1×	–	–
Qwen2.5-7B								
Base@1	0.498	1.0×	0.329	1.0×	0.278	1.0×	0.150	1.0×
Low-temperature@1	0.628	1.1×	0.524	1.2×	0.303	1.3×	0.425	1.23×
MCMC Power Sampling@1	0.706	23.7×	0.622	10.2×	0.318	18.9×	0.525	60.5×
Base@8	0.552	7.9×	0.713	9.4×	0.283	8.2×	0.547	7.1×
Low-temperature@8	0.700	9.1×	0.720	9.3×	0.308	9.5×	0.600	9.4×
DIVTHINK@8 (ours)	**0.734**	8.9×	**0.780**	9.0×	**0.343**	9.0×	**0.625**	8.5×
GRPO (MATH)@1	**0.740**	1.1×	**0.561**	1.0×	**0.354**	1.1×	–	–
DeepSeek-Math-7B-Instruct								
Base@1	0.402	1.0×	0.317	1.0×	0.318	1.0×	0.300	1.0×
Low-temperature@1	0.454	1.1×	0.482	1.2×	0.303	0.9×	0.175	0.8×
MCMC Power Sampling@1	0.472	24.8×	0.506	9.1×	0.323	17.7×	0.325	26.3×
Base@8	0.506	7.8×	0.677	8.3×	0.313	8.8×	0.325	7.3×
Low-temperature@8	0.500	8.3×	0.659	8.6×	0.323	6.5×	0.250	8.3×
DIVTHINK@8 (ours)	**0.522**	8.4×	**0.689**	8.2×	**0.328**	6.5×	**0.350**	8.5×
GRPO (MATH)@1	**0.492**	1.2×	**0.524**	1.3×	**0.333**	1.1×	**0.225**	1.6×

3.4 SAMPLING AND DECODING

After applying the repulsive update, each active trajectory samples its next token from the adjusted distribution:

$$x_t^{(k)} \sim \mathrm{Softmax}\left(\hat{\mathbf{l}}_t^{(k)}/\tau\right), \tag{9}$$

where τ is the decoding temperature after the repulsive adjustment. The sampled token is appended to the trajectory history, and decoding repeats until an end token is generated or the maximum length is reached. Finished trajectories are excluded from later repulsion fields.

In practice, DIVTHINK outputs K completed reasoning trajectories and can use standard aggregation strategies. For deterministic-answer tasks, we use majority voting over extracted final answers; for tasks with executable or rule-based checking, such as code generation, we evaluate pass@k.

4 EXPERIMENTS

4.1 EXPERIMENT SETUP

Models. We evaluate DIVTHINK across four base-model settings spanning general-purpose and math-specialized backbones. Qwen2.5-7B (Bai et al., 2023) serves as the general reasoning model, while Qwen2.5-Math-7B (Yang et al., 2024) and DeepSeek-Math-7B-Instruct (Shao et al., 2024) represent math-oriented instruction-tuned models. For these three 7B models, we additionally compare against their corresponding GRPO post-trained versions using the implementation of Shao et al. (2025), which post-trains each model on the MATH training split. To examine whether DIVTHINK extends beyond

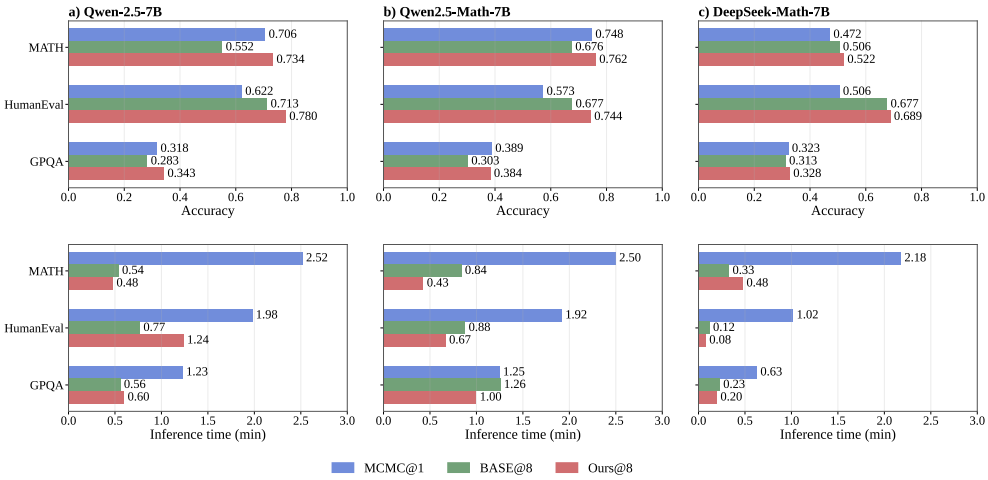

Figure 4: Inference-time comparison across decoding methods.

Table 2: Accuracy of Qwen3-14B under parallel decoding with eight trajectories. Bold values indicate the best result on each benchmark.

Method	MATH500	HumanEval	GPQA	AMC23
Qwen3-14B				
Base@8	**0.870**	**0.920**	0.530	0.700
Low-temperature@8	0.858	0.890	0.540	0.700
DIVTHINK@8 (ours)	**0.870**	0.909	**0.545**	**0.750**

the 7B scale, we further evaluate the larger Qwen3-14B model (Yang et al., 2025). Hyperparameter details are provided in Appendix A.4.

Benchmarks. We evaluate on four complementary benchmarks covering mathematics, code generation, and knowledge-intensive question answering. MATH500 (Hendrycks et al., 2021) and AMC23 test multi-step mathematical reasoning; HumanEval (Chen et al., 2021) measures functional correctness for generated code; and GPQA (Rein et al., 2023) evaluates expert-level scientific question answering. More detailed dataset descriptions are provided in Appendix A.5.

Baselines. We compare with four representative baselines: base sampling, low-temperature sampling (Wang et al., 2020), MCMC power sampling (Karan & Du, 2025), and GRPO (Shao et al., 2025). Base and low-temperature sampling capture standard independent parallel decoding. MCMC power sampling is a training-free inference-time refinement method based on iterative resampling, while GRPO is a post-training baseline for reinforcement-learning-based reasoning improvement. We also examined high-temperature sampling in preliminary experiments, but it frequently produced substantially longer and less coherent trajectories, leading to poor accuracy and excessive token usage. We therefore omit it from the main comparison, where it does not provide a meaningful accuracy–efficiency baseline.

Metrics. For deterministic-answer tasks, we report final accuracy after majority voting over generated trajectories. Specifically, for MATH500, AMC23, and GPQA, we extract the final answer from each trajectory and use the majority answer as the prediction. For HumanEval, where correctness is determined by executable unit tests, we report pass@k, which measures whether at least one of the k generated candidates solves the task.

4.2 MAIN RESULTS

Overall analysis. Tables 1 and 2 support three main conclusions. First, under a comparable final-trajectory budget, DIVTHINK is the strongest or a competitive training-free method across most 7B model–task pairs, improving accuracy by an average of 7.6 percentage points over Base@8 and 4.0

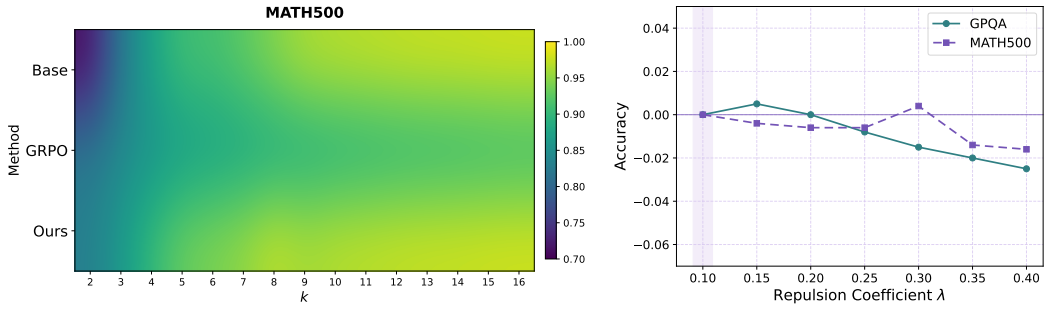

Figure 5: Ablation studies. Left: pass@k scaling on MATH500 with Qwen2.5-Math-7B. Right: accuracy changes across repulsion coefficients λ on GPQA and MATH500 with Qwen2.5-7B.

points over Low-temperature@8, with a maximum gain of 22.5 points. These gains therefore do not come from simply returning more candidates. Second, DIVTHINK achieves this accuracy with substantially less generated computation than iterative MCMC refinement, improving the practical accuracy–compute frontier. Third, the Qwen3-14B results show that the method continues to provide value at a larger scale, but the benefit becomes task dependent: it matches Base@8 on MATH500 (0.870), improves GPQA from 0.530 to 0.545 and AMC23 from 0.700 to 0.750, while Base@8 remains stronger on HumanEval. This pattern is consistent with repulsion being most useful when independent trajectories remain redundant; when the base candidate set is already strong, additional diversification offers less headroom and can slightly reduce local quality. The multi-seed evaluation in Figure 6 further confirms the robustness of these gains: DIVTHINK outperforms both parallel-sampling baselines under every seed, with statistically significant paired improvements.

Comparison with standard parallel sampling. Base@8 and Low-temperature@8 improve individual samples but do not optimize the candidate set jointly, allowing trajectories to repeat the same dominant reasoning pattern. DIVTHINK instead introduces a set-level objective that encourages complementary local choices while retaining the model's preferences. Its gains show that trajectory quality and candidate-set coverage are complementary, with less benefit when the base candidate set is already near saturation.

Comparison with MCMC Power Sampling. MCMC Power Sampling explores by repeatedly revising trajectories, whereas DIVTHINK diversifies the retained candidates directly. On Qwen2.5-7B MATH500, DIVTHINK improves accuracy from 0.706 to 0.734 while reducing token usage from 23.7× to 8.9×. This suggests that, when parallel decoding is available, coordinating final trajectories can be more efficient than sequentially replacing intermediate proposals.

Comparison with GRPO. GRPO improves the policy through offline optimization, whereas DIVTHINK coordinates trajectories only at inference. Their differing strengths indicate that post-training and decoding-time diversification are complementary rather than interchangeable. Crucially, DIVTHINK applies directly to an existing checkpoint without rollout collection, reward optimization, or parameter updates.

Inference-time efficiency. Figure 4 shows that the token savings translate into wall-clock gains rather than being offset by the coordination step. DIVTHINK remains close to Base@8 in runtime and is 4.5–5.8× faster than MCMC on MATH500 across the three 7B backbones; aggregated over the plotted model–task pairs, it reduces runtime by roughly 3×. This confirms that averaging peer distributions adds modest overhead relative to generation itself and preserves the throughput advantage of parallel decoding. Taken together with Table 1, the result places DIVTHINK on a favorable practical frontier: it improves the utility of parallel samples without introducing a sequential refinement loop or requiring post-training.

4.3 ABLATION STUDIES

Effect of the trajectory number k. We conduct the experiment on MATH500 by varying the number of parallel trajectories from $k = 2$ to $k = 16$. Figure 5 (left) reports pass@k, directly measuring how candidate-set coverage scales with the parallel inference budget. DIVTHINK reaches

the high-coverage region with fewer trajectories and is strongest in the moderate-budget regime, where the marginal value of each additional sample matters most. Because all methods approach saturation as k grows, the gap naturally narrows at large budgets. This behavior is consistent with the proposed mechanism: coordinated decoding improves how quickly the candidate set covers a correct solution, rather than changing the eventual ceiling obtained from sufficiently many samples.

Sensitivity to the repulsion λ. Figure 5 (right) shows a broad stable region for moderate repulsion. Although the best coefficient differs between GPQA and MATH500, performance changes little across $\lambda \in [0.10, 0.30]$, indicating that the method does not rely on a sharply tuned task-specific value. Accuracy degrades only when repulsion becomes large, particularly beyond $\lambda = 0.35$. This pattern is consistent with the intended trade-off: moderate repulsion reduces overlap among peer trajectories while retaining the base model's local quality signal, whereas excessive repulsion can over-penalize otherwise plausible continuations.

Multi-Seed Evaluation. To assess robustness to decoding randomness, we evaluate Qwen2.5-7B on HumanEval with eight trajectories across five random seeds.

As shown in Figure 6, DIVTHINK achieves the highest pass@8 under all five seeds, indicating that the improvement is not driven by a single favorable decoding run. This consistent ordering is also compatible with our hypothesis that coordinated diversification makes a fixed parallel budget more effective by reducing repeated exploration of the same failure modes. Across the five seeds, DIVTHINK increases mean pass@8 from 0.688 for Base@8 and 0.727 for Low-temperature@8 to 0.760, corresponding to gains of 7.20 and 3.28 percentage points, respectively. Both paired improvements are statistically significant ($p < 0.001$ and $p = 0.029$), supporting the robustness of the observed gains to decoding randomness in this setting.

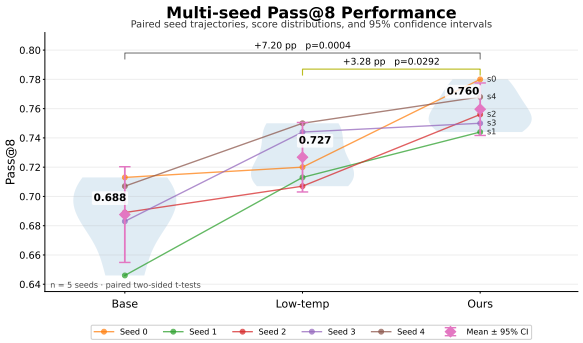

Figure 6: Multi-seed HumanEval pass@8 for Qwen2.5-7B. Error bars show 95% CIs over five decoding seeds.

4.4 QUALITATIVE CASE STUDY.

Figure 7 illustrates how reducing redundancy improves candidate-set coverage in code generation. All eight Base@8 samples make the same mistake: after locating a match at column index j, they replace the observed coordinate with `len(row)` $- j - 1$. Although syntactically valid, this transformation violates the task specification. In contrast, DIVTHINK preserves (i, j) and applies the required ordering only after collecting all coordinates, producing a solution that passes the executable verifier. The case shows that the benefit comes from avoiding a shared erroneous decision, rather than arbitrary surface variation, thereby creating opportunities to reach a valid path and improve pass@k coverage. Further details are provided in Appendix A.2.

5 CONCLUSION

We presented DIVTHINK, a training-free decoding method that improves parallel CoT reasoning by coordinating trajectories during generation. Through a lightweight repulsive logit adjustment, DIVTHINK reduces redundant local choices while preserving the base model's preferences, turning independent samples into complementary reasoning attempts.

Across diverse benchmarks and model backbones, DIVTHINK consistently improves over standard Base@8 and Low-temperature@8 sampling, with average gains of 7.6 and 4.0 percentage points and gains up to 22.5 points. It also uses far fewer generated tokens than MCMC Power Sampling while avoiding the rollout and optimization costs of GRPO-style post-training. These results suggest that reducing redundancy among parallel trajectories is an effective and practical direction for efficient test-time scaling.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and improving the clarity and readability of the manuscript. They were not used to formulate the research hypotheses, design the methodology, generate the experimental results, or develop theoretical claims. All AI-assisted text was reviewed and verified by the authors. The authors take full responsibility for the final content of the paper.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Seyedarmin Azizi, Erfan Baghaei Potraghloo, Minoo Ahmadi, Souvik Kundu, and Massoud Pedram. Power-smc: Low-latency sequence-level power sampling for training-free llm reasoning. *arXiv preprint arXiv:2602.10273*, 2026.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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. *arXiv preprint arXiv:2103.03874*, 2021.

Xiaotong Ji, Rasul Tutunov, Matthieu Zimmer, and Haitham Bou Ammar. Scalable power sampling: Unlocking efficient, training-free reasoning for llms via distribution sharpening. *arXiv preprint arXiv:2601.21590*, 2026.

Sheng Jia, Xiao Wang, and Shiva Kasiviswanathan. Training large language models to reason in parallel with global forking tokens. In *International Conference on Learning Representations*, volume 2026, pp. 84820–84846, 2026.

Jikun Kang, Xin Zhe Li, Xi Chen, Amirreza Kazemi, Qianyi Sun, Boxing Chen, Dong Li, Xu He, Quan He, Feng Wen, et al. Mindstar: Enhancing math reasoning in pre-trained llms at inference time. *arXiv preprint arXiv:2405.16265*, 2024.

Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. *arXiv preprint arXiv:2510.14901*, 2025.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in neural information processing systems*, 36:41451–41530, 2023.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

Rohin Manvi, Anikait Singh, and Stefano Ermon. Adaptive inference-time compute: Llms can predict if they can do better, even mid-generation, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models. *arXiv preprint arXiv:2504.15466*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Gleb Rodionov, Roman Garipov, Alina Shutova, George Yakushev, Erik Schultheis, Vage Egiazarian, Anton Sinitsin, Denis Kuznedelev, and Dan Alistarh. Hogwild! inference: Parallel llm generation via concurrent attention. *Advances in Neural Information Processing Systems*, 38:46592–46633, 2026.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

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

Ruiqi Shi and Sinno Jialin Pan. Diverse text decoding via iterative reweighting. In *The Fourteenth International Conference on Learning Representations*, 2026.

Weijie Shi, Yue Cui, Yaguang Wu, Jingzhi Fang, Shibo Zhang, Mengze Li, Sirui Han, Jia Zhu, Jiajie Xu, and Xiaofang Zhou. Semantic-guided diverse decoding for large language model. *Advances in Neural Information Processing Systems*, 38:148182–148229, 2026.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Haonan Wang, Chao Du, Kenji Kawaguchi, and Tianyu Pang. Think in parallel, answer as one: Logit averaging for open-ended reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 77445–77462, 2026.

Pei-Hsin Wang, Sheng-Iou Hsieh, Shih-Chieh Chang, Yu-Ting Chen, Jia-Yu Pan, Wei Wei, and Da-Chang Juan. Contextual temperature for language modeling. *arXiv preprint arXiv:2012.13575*, 2020.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Parathinker: Native parallel thinking as a new paradigm to scale llm test-time compute. *arXiv preprint arXiv:2509.04475*, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Wenkaishao Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for llm reasoning. *Advances in Neural Information Processing Systems*, 38: 43605–43631, 2026a.

Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. Multiverse: Your language models secretly decide how to parallelize and merge generation. *Advances in Neural Information Processing Systems*, 38:100773–100804, 2026b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

A APPENDIX

A.1 MEAN-FIELD DERIVATION OF THE REPULSIVE LOGIT UPDATE

This appendix explains why the logit update in Eq. (8) is a reasonable one-step approximation to a repulsive joint decoding objective. The derivation is organized in four parts: we first define the independent proposal, then introduce a repulsive target distribution, then approximate the joint collision cost by a per-trajectory expected cost, and finally convert that cost into a logit correction.

Independent proposal. At decoding step t, let

$$X_t = \{x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(K)}\}$$

denote the next-token choices of the K parallel trajectories, and let $H_{t-1} = \{h_{t-1}^{(1)}, \dots, h_{t-1}^{(K)}\}$ denote their histories. Standard parallel decoding samples each next token independently under the base model, giving the proposal distribution

$$Q(X_t \mid H_{t-1}) = \prod_{k=1}^{K} P_\theta(x_t^{(k)} \mid h_{t-1}^{(k)}).$$

This proposal preserves the base-model preference for each trajectory, but it contains no term that discourages two trajectories from choosing the same local continuation.

Repulsive target. To express the desired diversification effect, we define a target distribution that keeps the base-model proposal while penalizing duplicated next-token choices:

$$\pi(X_t \mid H_{t-1}) \propto Q(X_t \mid H_{t-1}) \exp(-\mathcal{L}_{\text{rep}}(X_t)),$$

where

$$\mathcal{L}_{\text{rep}}(X_t) = \frac{\lambda}{K-1} \sum_{1 \le i < j \le K} \mathbf{1}\left[x_t^{(i)} = x_t^{(j)}\right].$$

Here, $\lambda \ge 0$ controls the strength of the repulsion, and the factor $1/(K-1)$ matches the normalized repulsion field used in the main text. When $\lambda = 0$, π reduces to independent parallel sampling. When $\lambda > 0$, configurations with repeated token choices are downweighted. Equivalently, the correction from the independent proposal to the repulsive target is

$$\frac{\pi(X_t \mid H_{t-1})}{Q(X_t \mid H_{t-1})} \propto \exp(-\mathcal{L}_{\text{rep}}(X_t)). \tag{A.1}$$

Direct sampling from π would require coupled sampling over the full joint space of $|\mathcal{V}|^K$ token configurations. DIVTHINK therefore does not sample from this target exactly; it uses the target only to derive a cheap per-trajectory approximation.

Mean-field collision cost. Consider trajectory k and a candidate token $v \in \mathcal{V}$. If trajectory k chooses v, its collision cost with the other trajectories is

$$\frac{\lambda}{K-1} \sum_{j \ne k} \mathbf{1}\left[v = x_t^{(j)}\right].$$

Before the next tokens of the other trajectories are sampled, their exact values are unknown. The mean-field approximation replaces each unknown indicator by its expectation under the corresponding base next-token distribution $p_t^{(j)}$:

$$\mathbb{E}_{x_t^{(j)} \sim p_t^{(j)}}\left[\mathbf{1}\left[v = x_t^{(j)}\right]\right] = p_t^{(j)}(v).$$

Thus the expected repulsion cost assigned to token v for trajectory k is

$$c_t^{(k)}(v) = \frac{\lambda}{K-1} \sum_{j \ne k} p_t^{(j)}(v).$$

This cost is high for tokens that other trajectories are likely to choose, and low for tokens that are not strongly preferred by the peers.

From cost to logits. Using the expected cost above, the adjusted distribution for trajectory k can be written as the base distribution reweighted by an exponential penalty:

$$\tilde{p}_t^{(k)}(v) \propto p_t^{(k)}(v) \exp\Big(-c_t^{(k)}(v)\Big).$$

Since $p_t^{(k)}(v) \propto \exp(\ell_t^{(k)}(v))$, we have

$$\begin{aligned}\tilde{p}_t^{(k)}(v) &\propto \exp\Big(\ell_t^{(k)}(v)\Big) \exp\Big(-c_t^{(k)}(v)\Big) \\ &= \exp\Big(\ell_t^{(k)}(v) - c_t^{(k)}(v)\Big).\end{aligned}$$

Therefore, multiplying the base probability by an exponential repulsion penalty is equivalent to subtracting the expected collision cost from the logit:

$$\hat{\ell}_t^{(k)}(v) = \ell_t^{(k)}(v) - \frac{\lambda}{K-1} \sum_{j \neq k} p_t^{(j)}(v).$$

In vector form,

$$\hat{\ell}_t^{(k)} = \ell_t^{(k)} - \lambda \frac{1}{K-1} \sum_{j \neq k} p_t^{(j)},$$

which is exactly Eq. (8) with the repulsion field in Eq. (6).

Interpretation. The derivation shows that DIVTHINK is not an accept–reject method and does not run an MCMC chain. Instead, it uses the other trajectories' base distributions as soft forecasts of their likely next tokens, converts the resulting expected collision cost into a logit penalty, and samples once from the adjusted distribution. This keeps decoding lightweight while explicitly discouraging redundant local choices among parallel reasoning trajectories.

A.2 QUALITATIVE CASE STUDY

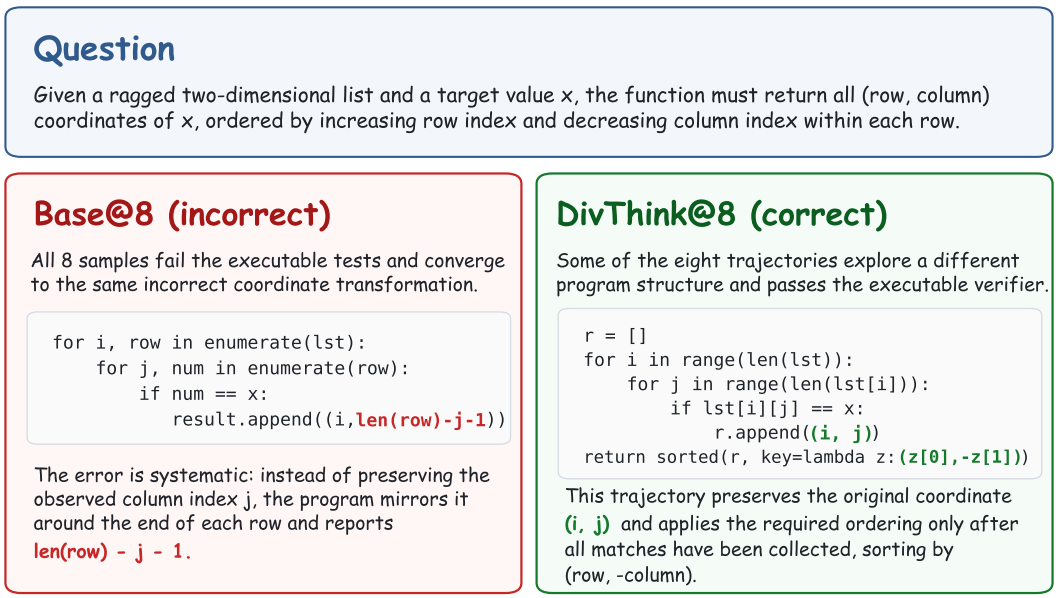

Figure 7: **Qualitative case study on HumanEval.** Base@8 collapses to a shared incorrect coordinate transformation, whereas DIVTHINK@8 explores a different program path that preserves the correct coordinate semantics and passes the executable tests.

Figure 7 illustrates how reducing redundancy can improve candidate-set coverage in code generation. All eight Base@8 samples converge to the same program-level mistake: after locating a match at

column index j, they replace the observed coordinate with the mirrored index $\texttt{len(row)} - j - 1$. Although syntactically valid and locally plausible, this transformation changes the coordinate and violates the task specification. In contrast, DIVTHINK follows a different program invariant: it preserves the observed coordinate (i, j) and postpones ordering until all coordinates have been collected, using a final sort by row ascending and column descending. This alternative trajectory passes the executable verifier.

The benefit comes not from arbitrary surface variation, but from avoiding a shared intermediate decision that causes independent samples to collapse onto the same failure mode. When the dominant reasoning pattern is incorrect, coordinated diversification creates additional opportunities to enter a valid solution path, which is particularly useful for pass@k evaluation.

A.3 ADDITIONAL RELATED WORK

Diversity-oriented decoding. Recent decoding methods explicitly target semantic diversity beyond independent sampling. SemDiD encourages semantically distinct generations through semantic guidance and inter-group repulsion, while OverRIDE suppresses semantic patterns observed in previously generated responses through iterative reweighting (Shi et al., 2026; Shi & Pan, 2026). These methods share our broader motivation of reducing redundancy and promoting diverse generations, but differ in how diversity is induced. DIVTHINK directly couples concurrently generated trajectories through their native next-token distributions, without semantic embeddings or iterative adaptation over completed responses.

Training-based parallel diversity. A complementary line of work promotes diverse reasoning modes through post-training. SSFT formulates parallel reasoning as a set-of-next-token-prediction problem and learns global forking tokens, while GFPO further optimizes these reasoning modes (Jia et al., 2026). DIVTHINK instead induces complementary trajectories directly at inference time and does not modify model parameters.

Coupled parallel decoding. ThinkMerge is particularly close in that it also introduces direct interaction among simultaneously generated reasoning trajectories at the next-token level (Wang et al., 2026). It averages logits across traces at synchronization points to aggregate information into a single coherent continuation. DIVTHINK operates in the same general decoding-time interaction setting but with a different objective: rather than encouraging agreement across trajectories, its repulsive update discourages redundant local choices so that multiple retained trajectories explore complementary reasoning paths.

A.4 IMPLEMENTATION DETAILS

For all methods, we cap the maximum generation length at $T_{\max} = 3072$ tokens, following the setting used in MCMC Power Sampling (Karan & Du, 2025); generation may terminate earlier when the model emits an EOS token. For low-temperature decoding, we set the sampling temperature to $1/\alpha = 0.25$. For MCMC Power Sampling, we follow the original implementation and use block size $B = 3072/16 = 192$, $\alpha = 4$, and $N_{\mathrm{MCMC}} = 10$ MCMC steps per stage.

A.5 DATASETS

We evaluate on four datasets that cover complementary forms of reasoning, including formal mathematical problem solving, executable code synthesis, and expert scientific question answering.

- **MATH500.** MATH500 is a standardized subset of the test split from the broader MATH benchmark (Hendrycks et al., 2021), which contains competition-style mathematics problems across categories such as algebra, geometry, number theory, counting and probability, and precalculus. The original MATH dataset contains $12{,}500$ problems, including $7{,}500$ training problems and $5{,}000$ test problems. MATH500 consists of 500 randomly selected high-school-level test problems and is widely used for evaluating multi-step mathematical reasoning. Since it is publicly accessible and increasingly saturated by strong LLMs, benchmark contamination is a potential concern.

- **AMC23.** AMC23 contains 40 problems from the 2023 American Mathematics Competitions. It focuses on challenging mathematical reasoning topics such as functional equations, algebraic manipulation, and complex analysis, and each problem requires an integer answer in the range of 0 to 999. Due to its relatively small size and public availability, repeated sampling can help reduce evaluation variance, while contamination remains a possible concern.
- **HumanEval.** HumanEval (Chen et al., 2021) is a code-generation benchmark with 164 handwritten programming problems covering algorithms, reasoning, mathematics, and language comprehension. Each problem is paired with unit tests, with an average of 7.7 tests per problem, and a generated solution is considered correct only if it passes all associated tests.
- **GPQA.** GPQA (Rein et al., 2023) is a multiple-choice science question-answering benchmark covering physics, chemistry, and biology. The questions are designed to require advanced domain knowledge and reasoning skills. We use GPQA Diamond, the highest-quality subset of GPQA, which contains 198 questions.

A.6 TEMPERATURE-CONTROLLED DIVERSIFICATION ON GPQA

This experiment asks whether DIVTHINK's gains can be reproduced simply by increasing sampling randomness. A temperature sweep provides a direct control: if generic diversity were sufficient, progressively higher temperatures should reduce redundancy while improving the aggregated answer. Instead, the results reveal a fundamental coverage–consensus trade-off. Higher temperatures substantially reduce collisions and initially increase best-of-8 coverage, but the additional candidates become less individually reliable and do not form a stronger correct majority; best-of-8 peaks at $T = 1.00$, whereas majority voting peaks much earlier at $T = 0.25$ and then deteriorates. The key insight is therefore that candidate-set diversity is useful only when it remains structured around plausible reasoning paths. The matched-temperature comparison isolates this distinction: at the same $T = 0.25$, DIVTHINK reduces collisions while simultaneously improving row accuracy, best-of-8, and majority accuracy by 2.2, 1.0, and 3.0 points, respectively. Unlike global temperature scaling, coordinated repulsion does not merely flatten the sampling distribution; it selectively discourages trajectories from repeating peer-preferred continuations while retaining the base model's local quality signal. This suggests that DIVTHINK improves how the parallel budget is allocated across reasoning modes, rather than obtaining gains from indiscriminate randomization. Because the analysis uses one decoding seed on 198 questions, it should be viewed as mechanism-oriented descriptive evidence rather than a statistical significance claim.

Table 3: Temperature sweep of independent Base@8 sampling on GPQA Diamond (198 questions). Majority-vote accuracy is the primary metric; best-of-8 and collision rate diagnose candidate-set coverage and redundancy.

Method / T	Row Acc.	Best-of-8	Majority	Collision
Base@8, 0.10	**32.7%**	76.3%	32.9%	20.9%
Base@8, 0.25	30.4%	78.8%	**35.4%**	11.2%
Base@8, 0.50	31.4%	79.8%	32.4%	7.2%
Base@8, 0.75	30.0%	86.9%	32.8%	5.1%
Base@8, 1.00	28.0%	**88.4%**	30.3%	3.4%
Base@8, 1.25	23.9%	84.8%	25.3%	**0.2%**

Table 4: Matched-temperature comparison at $T = 0.25$. DIVTHINK uses $\lambda = 0.4$ with K-normalized repulsion.

Method	Row Acc.	Best-of-8	Majority	Collision
Base@8	30.4%	78.8%	35.4%	11.2%
DIVTHINK@8	**32.6%**	**79.8%**	**38.4%**	**10.3%**
Δ	+2.2 pp	+1.0 pp	+3.0 pp	−0.9 pp

A.7 Symmetric Repulsion as the Number of Trajectories Increases

This experiment tests a potential failure mode of coordinated decoding: as the number of interacting trajectories grows, symmetric repulsion might accumulate and suppress a useful high-probability continuation in every trajectory, causing a collective all-miss failure. This concern is particularly relevant to DivThink because each trajectory receives feedback from all active peers. The results show that the token-level intervention does become stronger with K: consensus-token suppression increases by about $2.6\times$ from $K = 2$ to $K = 16$. However, this growth is far smaller than the increase in the number of peers and is consistent with the method's K-normalized averaging, which prevents the repulsion magnitude from scaling linearly with the parallel budget. More importantly, stronger local suppression does not produce the predicted downstream failure. The all-miss change is non-monotonic and returns to approximately zero at $K = 16$, while majority accuracy remains around 81–83% for $K \geq 4$ and best-of-K coverage eventually rises to 93.8%. The key insight is that DivThink primarily redistributes which trajectory retains a plausible continuation rather than removing that continuation from the entire candidate set. As K grows, the larger set can absorb stronger local diversification: individual trajectories separate more, but collectively they preserve consensus quality and expand coverage. The result is therefore qualitatively consistent with the proposed normalized repulsion mechanism and does not indicate an instability at larger K. Nevertheless, because the diagnostic uses only 64 MATH500 examples, it should be interpreted as evidence against this specific failure mode rather than as a general scaling guarantee.

Table 5: Symmetric-repulsion diagnostic on the first 64 MATH500 problems using Qwen2.5-Math-7B, $T = 0.25$, $\lambda = 0.4$, and seed 8. Probability changes are reported in units of 10^{-4}.

K	Best-of-K	Majority	Consensus ↓	Top-5 mass ↓	All-miss Δ
2	78.1%	71.9%	4.89	0.00	+0.66
4	89.1%	**82.8%**	8.86	0.02	+1.63
8	89.1%	81.2%	10.70	0.51	+0.19
16	**93.8%**	**82.8%**	12.81	0.78	−0.03

A.8 Position-Dependent Diversity on HumanEval

This experiment tests two aspects of the proposed mechanism: whether cross-trajectory repulsion changes only the first few decoding decisions or produces sustained trajectory-level diversification, and whether that diversification is obtained at the cost of code correctness. The positional analysis shows that the largest overlap reduction occurs in the first 64 tokens, when trajectories still share similar contexts and therefore exert the strongest redundant preferences on one another. Crucially, the effect persists in every later window even though its magnitude decreases. Together with the shorter shared prefix and lower exact-duplicate rate, this pattern supports an autoregressive amplification view of DivThink: a small number of early, coordinated deviations place trajectories in different contexts, after which those contextual differences continue to separate their subsequent program structures without requiring increasingly strong perturbations. This behavior is more consistent with complementary path exploration than with indiscriminate token-level noise, which would be expected to damage local program quality. Indeed, pass@1 remains unchanged at 31.7%, while pass@8 increases slightly from 67.7% to 68.9%. The accuracy difference corresponds to only two additional solved HumanEval problems and should not be treated as a robust performance gain; rather, the main insight is that DivThink measurably reduces lexical and trajectory-level redundancy without an evident correctness penalty. This is aligned with the method's design goal: preserve the base model's local preferences while using early cross-trajectory repulsion to allocate a fixed candidate budget across more distinct program paths.

Table 6: Pairwise Jaccard overlap by decoding position on HumanEval. Lower values indicate less lexical overlap among trajectories.

Token window	Base@8	DIVTHINK@8	Δ
0–64	0.578	0.552	-0.026
64–128	0.234	0.219	-0.015
128–192	0.185	0.171	-0.014
192–256	0.155	0.147	-0.007
256–320	0.152	0.141	-0.011
320–384	0.148	0.138	-0.010
384–448	0.143	0.140	-0.003
448–512	0.138	0.133	-0.005

Table 7: HumanEval correctness and trajectory-level redundancy for DeepSeek-Math-7B-Instruct with $K = 8$, $T = 0.25$, $\lambda = 0.4$, and seed 8.

Metric	Base@8	DIVTHINK@8	Δ
pass@1	31.7%	31.7%	-
pass@8	67.7%	**68.9%**	+1.2 pp
Exact duplicate pairs	8.71%	**7.99%**	-0.72 pp
Mean shared prefix	17.86	**14.21**	-3.66 tokens

A.9 COMPUTATIONAL OVERHEAD

DIVTHINK does not require additional model forward passes beyond those used to generate the K parallel trajectories. The only additional operation at each decoding step is the aggregation of peer next-token distributions and the corresponding logit subtraction. This introduces a synchronization step across the active trajectories but preserves batched parallel generation.

A.10 LIMITATIONS

Our evaluation is limited by computational resources. We focus primarily on 7B and 14B-scale model families and do not evaluate substantially larger backbones such as 32B or 72B models. Although the observed gains are consistent across general-purpose, math-specialized, and instruction-tuned models, larger models may exhibit different redundancy and diversity patterns.

Our benchmark coverage is also not exhaustive. We evaluate representative tasks in mathematical reasoning, code generation, and scientific question answering, but do not cover all reasoning domains, multilingual settings, or long-context interactive tasks. Future work could test whether coordinated parallel decoding provides similar benefits in broader application scenarios.