

Improving Diffusion Language Model Decoding through Joint Search in Generation Order and Token Space

Yangyi Shen^{1*}, Tianjian Feng^{2*}, Jiaqi Han¹, Wen Wang²,
Tianlang Chen¹, Chunhua Shen^{2,3}, Jure Leskovec^{1†}, Stefano Ermon^{1†}

¹Stanford University ²Zhejiang University ³Ant Group

Correspondence: pyyshen@stanford.edu *Equal contribution. †Equal senior supervision.

Abstract

Diffusion Language Models (DLMs) offer order-agnostic generation that can explore many possible decoding trajectories. However, current decoding methods commit to a single trajectory, limiting exploration in trajectory space. We introduce **Order-Token Search** to explore this space through jointly searching over generation order and token values. Its core is a likelihood estimator that scores denoising actions, enabling stable pruning and efficient exploration of diverse trajectories. Across mathematical reasoning and coding benchmarks, Order-Token Search consistently outperforms baselines on GSM8K, MATH500, Countdown, and HumanEval (3.1%, 3.8%, 7.9%, and 6.8% absolute over backbone), reaching accuracy comparable to diffu-GRPO post-training. Our work establishes joint search as a key component for advancing decoding in DLMs.

1 Introduction

Diffusion Language Models (DLMs) have emerged as a promising alternative to autoregressive models for sequence generation. In particular, Masked Diffusion Models (MDMs) (Sahoo et al., 2024; Shi et al., 2024) train on a core objective: learning to reconstruct clean text by denoising sequences corrupted with random masks at varying rates. At inference, generation starts from a fully masked sequence and proceeds over denoising steps: the model predicts token distributions for all positions, samples tokens for masked positions, and re-masks a subset of tokens to construct the next input. One key opportunity is that MDMs enable order-agnostic decoding: they revise positions in parallel, allowing adaptive generation orders rather than left-to-right. This flexibility makes choosing generation order a central design for decoding.

However, current decoding methods produce only a single trajectory per run. In MDMs, common decoders follow heuristic remasking rules that

effectively determine the trajectory, which positions to update at each step and with what tokens. For instance, **random remasking** mirrors the training corruption by repeatedly masking a random subset of tokens, whereas **low-confidence** remasking greedily fixes high-confidence tokens and only revises uncertain ones (Nie et al., 2025b; Kim et al., 2025a). As illustrated in Figure 1(a)(b), because these heuristics neither branch over alternative generation orders nor maintain competing intermediate states, a single run can easily miss the specific order-token trajectory needed for a correct solution.

Empirically, existing methods occupy opposite ends of an exploration-exploitation trade-off. The *pass@k* metric captures the probability that at least one of *k* samples is correct. **Low-confidence remasking** exploits by committing early to high-confidence tokens and restricting decoding to a narrow set of trajectories, which boosts *pass@1* but leads to slow *pass@k* growth as *k* increases. In contrast, **random remasking** explores a broader set of trajectories and attains higher *pass@k*, but achieves weak *pass@1* because it cannot reliably select good ones. Thus, prior methods either over-exploit or explore blindly, leaving much of the model’s multi-trajectory potential unrealized. We address this with explicit trajectory search and pruning.

Here we propose **Order-Token Search**, a decoding algorithm that performs search in the joint space of generation orders and token values. As shown in Figure 1(c), Order-Token Search maintains beams of partially denoised states and periodically expands each beam by proposing multiple denoising actions, then prunes using a stable likelihood estimator that reliably scores partial trajectories. Across mathematical reasoning and coding benchmarks, Order-Token Search consistently improves *pass@1* over prior best decoding, reaching accuracy comparable to diffu-GRPO post-training (Zhao et al., 2025) (e.g., +3.1/3.8/7.9/6.8% absolute over low-confidence remasking on

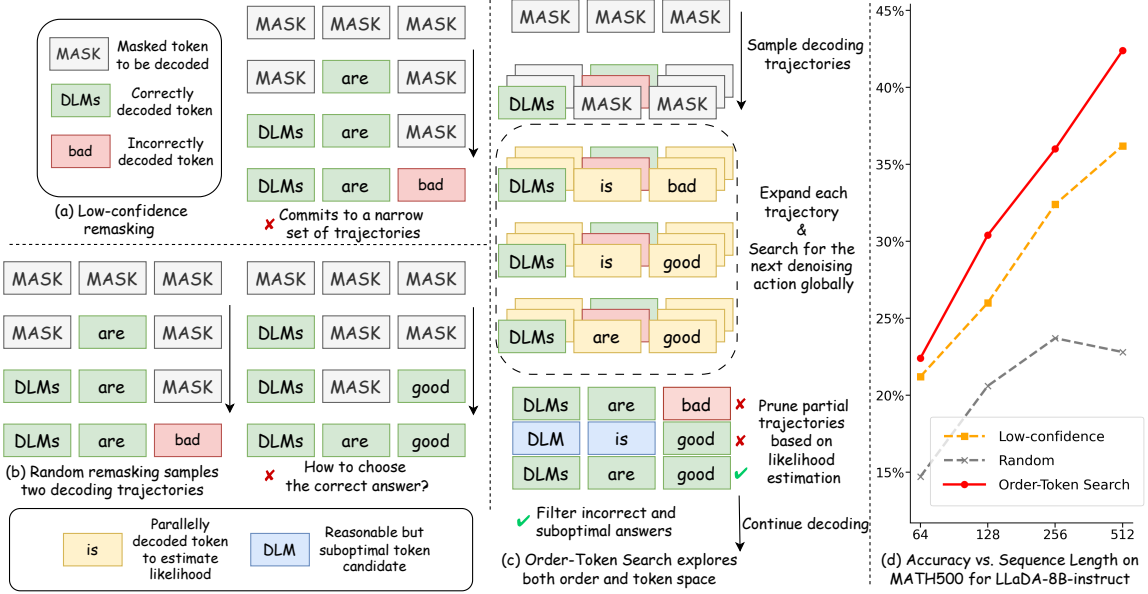


Figure 1: **Why heuristic remasking fails, and how Order-Token Search helps.** (a) **Low-confidence remasking** greedily fixes high-confidence tokens and revises low-confidence positions, which narrows the set of decoding trajectories and can get stuck in an incorrect completion. (b) **Random remasking** uniformly samples different remasking patterns, producing more distinct trajectories (different orders and token assignments), but offers no principles to select the correct trajectory among competing candidates. (c) **Order-Token Search** maintains beams of partial trajectories, periodically expands each trajectory by proposing denoising actions that change both which position to update (order) and what token to write (token), and then prunes partial trajectories using a likelihood estimator to filter incorrect and suboptimal candidates before continuing decoding. (d) **MATH500 accuracy plot** (example dataset) illustrating Order-Token Search’s advantage over low-confidence and random remasking.

GSM8K/MATH500/Countdown/HumanEval). Figure 1(d) illustrates the gains on MATH500: Order-Token Search consistently outperforms both low-confidence and random remasking.

Our contributions are threefold: **(1)** We identify an exploration–exploitation trade-off in MDM decoding through $pass@k$: low-confidence remasking over-exploits, improving $pass@1$ but suppressing $pass@k$, while random remasking blindly explores, boosting $pass@k$ but degrading $pass@1$ (Section 3). **(2)** We propose Order-Token Search, a decoding algorithm that searches in the joint space of generation orders and token values and stably prunes partial trajectories with a likelihood estimator (Section 4). **(3)** We demonstrate consistent gains across mathematical reasoning and coding benchmarks, reaching improvements typically obtained from post-training (Section 5).

2 Background

In this section, we review fundamentals of Masked Diffusion Models, formalize key concepts for remasking strategies, and define our evaluation metrics for the exploration trade-off.

2.1 Discrete Diffusion Models

Diffusion Language Models extend discrete diffusion modeling to text, and recent large-scale training has made them competitive with autoregressive models (Nie et al., 2025b; Ye et al., 2025). Concretely, discrete diffusion models adapt the forward diffusion process and the reverse denoising process (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song and Ermon, 2019; Song et al., 2021) to discrete data by establishing the diffusion process over a discrete domain $\mathbf{x} \in \mathcal{X}$, where $\mathbf{x}$ is a one-hot vector denoting tokens from a vocabulary of size $|\mathcal{X}|$ (Austin et al., 2021a). Given a prior π , the forward process q incrementally corrupts the original data $\mathbf{x}_0$ into a target prior distribution $\text{Cat}(\cdot; \pi)$. Over continuous time $t \in [0, 1]$, it forms a sequence of increasingly noisy latent variables $\mathbf{x}_t$, through the conditional marginal distribution $q(\mathbf{x}_t | \mathbf{x}_0) = \text{Cat}(\mathbf{x}_t; \alpha_t \mathbf{x}_0 + (1 - \alpha_t) \pi)$. Here, α_t is a monotonically decreasing noise schedule that satisfies boundary conditions $\alpha_0 = 1$ and $\alpha_1 = 0$. Furthermore, we can achieve the transition probability between any two intermediate time points $0 < s < t < 1$ through $q(\mathbf{x}_t | \mathbf{x}_s, \mathbf{x}_0) =$

$\text{Cat}(\mathbf{x}_t; \alpha_t/\alpha_s \mathbf{x}_s + (1 - \alpha_t/\alpha_s)\boldsymbol{\pi})$.

Masked Diffusion Model (MDM), a specific instance of this framework, utilizes the prior $\boldsymbol{\pi} = \mathbf{m}$ to achieve absorbing-state diffusion, a particularly suitable setting for language modeling (Sahoo et al., 2024; Shi et al., 2024; Lou et al., 2024). Here, $\mathbf{m}$ is a one-hot vector corresponding to a special MASK token. Defining s as the time step immediately preceding t , the posterior distribution simplifies to:

$$q(\mathbf{x}_s | \mathbf{x}_t, \mathbf{x}_0) = \begin{cases} \text{Cat}(\mathbf{x}_s; \mathbf{x}_t), & \mathbf{x}_t \neq \mathbf{m} \\ \text{Cat}(\mathbf{x}_s; \frac{\alpha_s - \alpha_t}{1 - \alpha_t} \mathbf{x}_0 + \frac{1 - \alpha_s}{1 - \alpha_t} \mathbf{m}), & \mathbf{x}_t = \mathbf{m} \end{cases} \quad (1)$$

The reverse (denoising) process is modeled by $p_\theta(\mathbf{x}_s | \mathbf{x}_t) = q(\mathbf{x}_s | \mathbf{x}_t, \mathbf{x}_\theta(\mathbf{x}_t))$, where p_θ is a parameterized distribution that reverses q , and $\mathbf{x}_\theta(\mathbf{x}_t)$ denotes a neural network trained to predict the original clean data $\mathbf{x}_0$ from its noisy version $\mathbf{x}_t$. This network is optimized by minimizing the negative evidence lower bound, thereby learning to approximate the true posterior distribution.

2.2 Remasking Strategies in MDM

MDM decoding starts from a fully masked sequence, $\mathbf{x}_1 = (\text{MASK}, \dots, \text{MASK})$. The model then iteratively refines this sequence over steps. At each step, the model predicts logits for all currently masked positions. The critical action in this reverse process is the transfer of a prediction—that is, the act of replacing a selected MASK token with its predicted value, thereby committing to that prediction for subsequent steps. The rule that determines which masked token to transfer next is known as the **remasking strategy**, and it defines the decoding order. We focus on three primary strategies:

Random Remasking. The strategy used during training. The position to unmask is chosen uniformly at random from the set of all remaining masked positions. This is a baseline that ensures unbiased, order-agnostic generation. **Low-Confidence Remasking.** A common inference-time strategy. The position with the highest confidence score is unmasked next; the positions with lower scores are remasked. Formally, at each step, the model computes a confidence score for position i as the predicted probability assigned to the currently sampled token at that position $s_i = p_\theta(\cdot | \mathbf{x}_t)_i$. The position with the *maximum* score s_i is unmasked. The intuition is to resolve the position where the model has the greatest certainty first, potentially mitigating error propagation (Nie et al., 2025b; Kim et al., 2025a). **Autoregressive**

(AR). We force the MDM to keep the leftmost predicted token and remask all following tokens. This baseline decouples the effect of generation order and examines the effect of diverse token sampling.

2.3 Metrics for Exploration Trade-off

Evaluating the trade-off between exploration and exploitation requires metrics that capture both deterministic performance and the model’s inherent capability. We use the following metrics established in prior work (Yue et al., 2025): **Accuracy** (*pass@1*), the probability that a single generated sample is correct. This metric evaluates one-trial performance and represents the expected accuracy in a non-deterministic setup. *pass@k*, the probability that at least one sample out of k independent generations is correct. This metric estimates the model’s inherent capability to solve a problem given sufficient sampling budget. For a problem with n generated samples of which c are correct, it is estimated as: $\text{pass}@k \approx 1 - \binom{n-c}{k} / \binom{n}{k}$.

The gap between *pass@1* and *pass@k* measures how much a decoding method benefits from sampling. If *pass@1* is strong but *pass@k* increases slowly with k , the generated samples are highly correlated, indicating conservative decoding that commits early and explores few distinct trajectories. Conversely, if *pass@1* is weak but *pass@k* rises quickly with k , the model can reach correct solutions under sampling, but the decoding method cannot reliably select them. Our goal is to translate this sampling-driven capability into higher *pass@1* by explicitly exploring and pruning in the joint space of generation orders and token choices.

3 Trade-off in Decoding Methods

We study the exploration–exploitation trade-off in MDM decoding by asking how trajectory diversity affects problem-solving performance. We compare three representative methods: low-confidence remasking (low diversity), random remasking (high diversity in generation order), and autoregressive decoding (high diversity in token selections). This comparison is particularly valuable because MDMs offer unique flexibility in generation order compared to autoregressive models, yet optimal strategies for leveraging this flexibility remain unclear.

Low-confidence remasking: stronger *pass@1*, weaker *pass@k* growth. Using LLaDA models (Nie et al., 2025b; Zhu et al., 2026), Figure 2 reports *pass@1* and *pass@k* on math

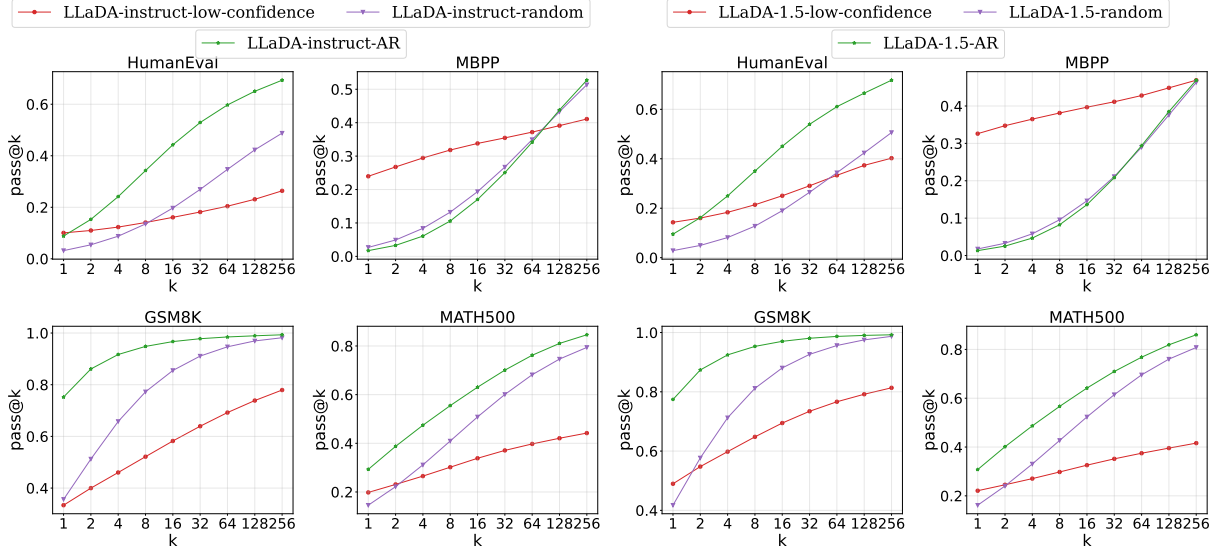


Figure 2: **Standard MDM decoders reveal an exploration-exploitation trade-off.** Empirical $pass@k$ curves for LLaDA-8B-Instruct and LLaDA-1.5 on math and coding benchmarks. Low-confidence remasking often achieves higher $pass@1$ by exploiting local token confidence, but its $pass@k$ grows slowly with k , indicating limited coverage from a narrow set of trajectories. In contrast, random remasking (order diversity) and AR decoding (token diversity) typically start with lower $pass@1$ yet reach higher $pass@k$ by $k = 256$, reflecting broader solution-space coverage through more diverse trajectories. We use $pass@k$ here only as a diagnostic of latent multi-trajectory potential; the goal is to convert this potential into improved $pass@1$ via structured exploration and selection.

(GSM8K, MATH500) and coding (HumanEval, MBPP (Austin et al., 2021b)) benchmarks. Across 8 dataset-model settings, low-confidence remasking achieves the best $pass@1$ in 4 and the second-best in 3. This supports the intuition that filling high-confidence tokens provides stable context within a single trajectory. However, its $pass@k$ grows much more slowly as k increases, suggesting that it over-exploits local confidence and collapses to a narrow set of trajectories.

Exploratory decoding: lower $pass@1$, higher $pass@k$. Random remasking and left-to-right AR decoding usually begin with lower $pass@1$, but their $pass@k$ increases much faster and eventually surpasses low-confidence remasking in most settings. For example, on MATH500 at $pass@256$, random remasking and AR reach about 0.8, while low-confidence remasking remains near 0.4. This indicates that diversity helps uncover correct solutions missed by a single committed trajectory. That diversity can come from either varying generation orders by random remasking or varying token choices under AR. We also find that AR performs best on math, achieving the highest $pass@1$ and $pass@k$ on GSM8K and MATH500, possibly because these tasks align better with left-to-right reasoning. This advantage does not carry over to coding: on HumanEval, AR has the lowest $pass@1$.

Summary and implication. Standard decoding methods expose a clear exploration-exploitation trade-off in MDMs. Low-confidence decoding improves $pass@1$ by exploiting local confidence, but narrows trajectory diversity and misses alternative valid solutions. More exploratory methods reveal broader solution-space coverage, but with lower per-sample reliability. Additional diversity measurements corroborate this trade-off; see Appendix A.11. We use $pass@k$ as a diagnostic of exploratory potential, and the key challenge is how to turn that potential into better $pass@1$.

4 Method: Order-Token Search

Section 3 suggests that improving $pass@1$ requires turning trajectory diversity into a reliable single output. To this end, we introduce Order-Token Search, a novel algorithm inspired by beam search but tailored to the parallel, order-agnostic dynamics of MDMs. Order-Token Search searches jointly over generation order and token choices, and uses the model’s likelihood prediction to score partial trajectories and stably prune unpromising ones. This section describes its two components: search and pruning. Full pseudocode is deferred to Appendix A.1.

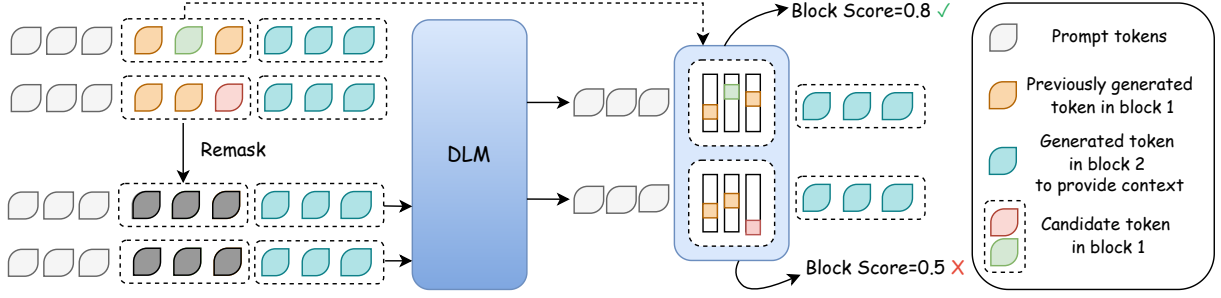


Figure 3: Illustration of pruning in Order-Token Search for DLMs. At one search step, we have two fully denoised candidates (left), where yellow tokens were revealed earlier. **We mask the current block (middle three tokens), feed each masked candidate into the DLM, and obtain token probabilities for the masked positions.** The score is the chain-rule product of these probabilities, which is used to prune the lower-likelihood candidate.

4.1 Search Process

Order-Token Search maintains a beam of K partially denoised candidates, initialized as K copies of $\mathbf{x}_1 = [\mathbf{c}; \text{MASK}^L]$, where $\mathbf{c}$ is the prompt and MASK^L denotes L mask tokens. Unlike single-trajectory decoding, Order-Token Search branches over both uncertainties in DLM generation: *which* tokens to write and *when/where* to write them.

At each user-specified interval (s, t) , every beam is expanded for several denoising steps. The MDM predicts distributions over masked positions, from which the algorithm samples multiple denoising actions, corresponding to different revealed positions and token assignments. Each action produces a new partial candidate, creating multiple trajectories from the same prefix. Pruning then keeps only the top- K candidates according to Section 4.2.

To control cost, we use **block diffusion** (Arriola et al., 2025; Nie et al., 2025b). Instead of branching at every denoising step, we branch only at block boundaries and denoise within each block independently. This reduces the number of search decisions from the number of denoising steps $|T|$ to the number of blocks $|B|$, lowering the overhead from $O(K|T|)$ to $O(K|B|)$ while still maintaining substantial diversity in both token ordering and values. After all blocks are processed, we return the highest-scoring candidate. Computational complexity details are given in Appendix A.3.

4.2 Pruning Process

The effectiveness of Order-Token Search depends on a pruning score that can reliably compare partially denoised candidates with different token choices and generation orders. A natural option is the standard MDM likelihood of reconstructing the full clean sequence from the current noisy state.

In practice, however, this score is high-variance and unstable: when many tokens remain masked, the model is evaluated on a difficult global infilling problem that is mismatched with the small-step denoising behavior it learns best (Kim et al., 2025a).

Scoring incremental denoising actions. Instead of using a monolithic full-sequence likelihood at an intermediate state, we score the denoising actions that produced it. For a transition from a more corrupted state $\mathbf{x}_t$ to a less corrupted state $\mathbf{x}_s$ with $0 \leq s < t \leq 1$, let $\mathbf{x}_0 \sim p_\theta(\cdot | \mathbf{x}_t)$ be a sample from the parametrized posterior. We define

$$\Delta(s, t) = \{i \mid x_{t,i} = \text{MASK} \cap x_{s,i} \neq \text{MASK}\},$$

$$\tilde{\mathbf{x}}_0 = \text{mask}(\mathbf{x}_0, \Delta(s, t)) \quad (2)$$

Here, $\Delta(s, t)$ contains the positions newly revealed during this search interval, and $\tilde{\mathbf{x}}_0$ is obtained by re-masking those positions in $\mathbf{x}_0$ while retaining the model-predicted context elsewhere. We then score only the re-masked positions:

$$s(\mathbf{x}_t; \mathbf{x}_s) = \mathbb{E}_{\mathbf{x}_0} \left[\log p_\theta \left(\mathbf{x}_0^{\Delta(s,t)} \mid \tilde{\mathbf{x}}_0 \right) \right] \quad (3)$$

Intuitively, $s(\mathbf{x}_t; \mathbf{x}_s)$ measures how confidently the model can re-predict the tokens newly revealed, using the surrounding context from its own prediction $\mathbf{x}_0$. Figure 3 illustrates this block-level score.

Trajectory-level score for pruning. We assign each candidate a cumulative score over all branching intervals $\mathcal{I}$: $S(\text{candidate}) = \sum_{(s,t) \in \mathcal{I}} s(\mathbf{x}_t; \mathbf{x}_s)$. Because each term scores only a few denoising steps, it is lower-variance and more comparable across candidates than full-sequence likelihood, which can over-penalize intermediate states for being “globally hard” when many masks remain. Candidates that repeatedly make model-

Table 1: **Model Performance on Math and Coding Benchmarks.** We report accuracy across four benchmarks and multiple generation lengths for two base models (LLaDA and LLaDA-1.5). Bolded values indicate the best performance; underlined values indicate the second best. diffu-GRPO is post-training applied to LLaDA only; we report available lengths. Order-Token Search achieves the highest average accuracy for both base models and attains best dataset-level averages on MATH500, Countdown, and HumanEval, while remaining competitive on GSM8K.

		GSM8K					MATH500					Countdown					HumanEval					Average
	Method / Seq Len	64	128	256	512	Avg	64	128	256	512	Avg	64	128	256	512	Avg	64	128	256	512	Avg	
LLaDA	Low-confidence	44.3	68.7	76.7	78.2	67.0	21.2	26.0	32.4	36.2	29.0	25.8	20.7	19.5	16.0	20.5	10.4	18.9	26.2	26.8	20.6	34.3
	Low-conf + MV	46.4	72.5	80.9	83.1	70.7	20.2	27.4	35.0	36.2	29.7	22.7	23.8	18.4	18.0	20.7	11.6	18.9	26.2	26.8	20.9	35.5
	Random remasking	33.4	56.5	63.6	63.1	54.2	14.7	20.6	23.7	22.8	20.5	8.0	13.3	10.9	11.7	11.0	12.3	19.3	27.0	25.9	21.1	26.7
	Random + MV	43.1	70.7	80.2	80.3	68.6	17.2	26.2	31.8	31.8	26.8	6.3	15.2	14.1	15.2	12.7	15.9	17.1	29.9	28.1	22.7	32.7
	AR	34.0	62.7	75.7	76.9	62.3	18.8	23.4	27.4	34.4	26.0	10.6	12.9	13.3	14.1	12.7	10.4	20.1	29.9	31.7	23.0	31.0
	AR + MV	41.0	69.1	81.7	86.4	69.6	17.4	23.0	32.2	39.9	28.1	10.2	13.3	11.3	13.7	12.1	9.8	20.1	29.9	31.7	22.9	33.2
	AR + beam-search	40.3	70.4	81.1	82.9	68.7	22.2	26.6	35.4	39.8	<u>31.0</u>	18.4	23.1	21.5	21.9	<u>21.2</u>	15.9	22.6	36.0	34.8	<u>27.3</u>	<u>37.1</u>
	diffu-GRPO	-	72.6	79.8	81.9	-	-	33.2	37.2	39.2	-	-	33.2	31.3	37.1	-	-	29.3	39.0	34.8	-	-
	Order-Token Search	45.6	71.6	79.8	83.3	<u>70.1</u>	22.4	30.4	36.0	42.4	32.8	27.7	34.4	26.2	25.4	28.4	15.2	23.2	34.2	37.2	27.4	39.7
LLaDA-1.5	Low-confidence	44.2	69.6	77.5	79.4	67.7	20.2	26.4	32.5	36.2	28.8	19.8	19.7	17.9	21.8	19.8	11.0	15.9	31.1	28.7	21.7	34.5
	Low-conf + MV	49.1	74.4	84.1	84.2	73.0	21.2	30.0	34.8	39.3	<u>31.3</u>	20.7	23.8	20.3	25.4	<u>22.5</u>	11.6	16.5	31.1	28.7	22.0	<u>37.2</u>
	Random remasking	45.0	60.6	65.6	66.9	59.5	19.6	21.1	27.9	28.8	24.4	8.5	11.8	10.1	11.4	10.5	11.8	17.6	26.8	26.5	20.7	28.8
	Random + MV	47.7	74.8	81.2	82.8	71.6	22.4	26.2	31.0	30.4	27.5	5.5	16.4	9.0	14.5	11.4	13.4	15.9	29.9	27.4	21.7	33.1
	AR	37.5	67.7	77.3	79.1	65.4	17.2	23.5	31.2	35.0	26.7	12.3	15.2	14.5	15.7	14.4	13.4	22.7	30.5	30.5	24.2	32.7
	AR + MV	40.7	72.9	83.8	84.2	70.4	18.4	26.4	33.6	37.9	29.1	12.5	17.2	14.8	16.0	15.1	12.8	22.6	30.5	30.5	24.1	34.7
	AR + beam-search	45.1	73.2	82.0	84.9	71.3	19.0	26.4	35.2	38.8	29.9	14.8	21.1	16.0	20.3	18.1	12.2	20.7	32.8	33.8	<u>24.9</u>	36.1
	Order-Token Search	48.4	74.5	81.7	84.0	<u>72.2</u>	24.4	30.8	37.4	42.4	33.8	27.7	31.3	23.8	29.3	28.0	15.2	22.6	36.0	36.6	27.6	40.4

supported decisions accumulate higher scores, while unlikely trajectories are pruned early.

Order-Token Search alternates search and pruning to turn trajectory diversity into improved *pass@1*. Search explores diverse order-token trajectories, and pruning keeps the most model-consistent candidates using a trajectory-level score over incremental denoising actions. Under a fixed inference-time budget, this enables broader exploration and more reliable selection.

5 Experiments

We evaluate Order-Token Search (OTS) on three questions: (1) Does it improve DLM decoding accuracy over strong baselines across diverse tasks? (2) Is the dedicated OTS likelihood estimator important for pruning? (3) How does OTS scale with test-time compute (NFE)?

5.1 Experimental Setup

Baselines. We compare Order-Token Search (OTS) against several strong baselines: **Low-confidence remasking**, a greedy decoding method adopted as an optimal base model configuration in Zhao et al. (2025). **Random remasking**, an exploratory decoding method that re-masks uniformly at random. **Random remasking with majority voting**, which generates a compute-equivalent set of diverse samples via random remasking and selects answers using a consistency heuristic (Wang et al., 2023). **Low-confidence with majority voting**, which combines the greedy decoding with the consistency heuristic mentioned above. **AR**, which

follows the left-to-right autoregressive order in generation. **AR with majority voting** and **AR with beam search**, which strengthen the AR baseline with, respectively, a consistency heuristic and a search based on OTS likelihood estimator. For pruning ablations, we also compare alternative likelihood estimators: **OTS All Blocks**, which scores each expansion by the joint log-likelihood of all blocks revealed so far; **OTS Future Blocks**, which scores the newly-revealed blocks while masking all unrevealed future blocks (see Appendix A.2).

Model and Tasks. Our primary testbed is **LLaDA-8B-Instruct** (Nie et al., 2025b), a state-of-the-art open-source diffusion language model. Since it has not undergone post-training with methods like diffu-GRPO (Zhao et al., 2025), it offers a clean baseline for isolating the performance improvements attributable to our inference-time algorithm. We additionally evaluate **LLaDA-1.5** (Zhu et al., 2026), an RL post-trained variant of LLaDA, to verify that our conclusions hold even after reinforcement-learning-based post-training. We further evaluate on **Dream7B** (Ye et al., 2025) to test whether our conclusions hold beyond the LLaDA family; results are reported in Appendix A.12. For tasks, we evaluate on three mathematical reasoning and one coding benchmarks. **GSM8K** (Cobbe et al., 2021) contains $\sim 1.32k$ grade school math problems requiring multi-step reasoning. **MATH500** (Lightman et al., 2024) is a challenging subset of 500 high-school competition-level problems from the MATH (Hendrycks et al., 2021) dataset. **Countdown** (Pan et al., 2025) is a combinatorial arithmetic game where the goal is to

reach a target number using basic operations on a given set. **HumanEval** (Chen et al., 2021) is a code generation benchmark of 164 hand-written Python programming tasks, where models must synthesize a function from a docstring and pass unit tests.

5.2 Order-Token Search Improves Decoding

Overall performance: Order-Token Search is the strongest decoding method across benchmarks. Table 1 shows that, on both LLaDA and LLaDA-1.5, Order-Token Search achieves the highest overall average accuracy over the strongest baseline from 37.1% to 39.7% on LLaDA and from 37.2% to 40.4% on LLaDA-1.5. It also attains the best dataset-level averages on MATH500, Countdown, and HumanEval. On GSM8K, multi-trajectory baselines such as majority voting and AR beam search outperform Order-Token Search at some sequence lengths, likely reflecting limited headroom: these methods already perform strongly there, and the pass@ k results show AR is particularly effective on GSM8K. Notably, Order-Token Search also outperforms expensive post-training approaches such as diffu-GRPO (Zhao et al., 2025); for example, at sequence length 512 it achieves higher accuracy on GSM8K (83.3% vs. 81.9%), MATH500 (42.4% vs. 39.2%), and HumanEval (37.2% vs. 34.8%).

Diffusion baselines: Order-Token Search improves over remasking-based decoding, especially on harder tasks. Among diffusion-style baselines, low-confidence remasking is already strong, and adding majority voting further improves GSM8K for both models. Replacing confidence-based remasking with random remasking can remain competitive on GSM8K, but hurts more structured reasoning tasks. For example, Random+MV achieves only 12.7% and 11.4% average accuracy on Countdown for LLaDA and LLaDA-1.5, far below the 20.7% and 22.5% of Low-conf+MV. In contrast, Order-Token Search consistently outperforms these diffusion baselines on the harder benchmarks MATH500, Countdown, and HumanEval. Relative to Low-conf+MV, it improves MATH500 from 29.7% to 32.8% on LLaDA and from 31.3% to 33.8% on LLaDA-1.5, raises Countdown from 20.7% and 22.5% to 28.4% and 28.0%, and improves HumanEval from 20.9% and 22.0% to 27.4% and 27.6%.

Autoregressive baselines: Order-Token Search consistently outperforms AR-based decoding. AR and AR + Majority-Voting are

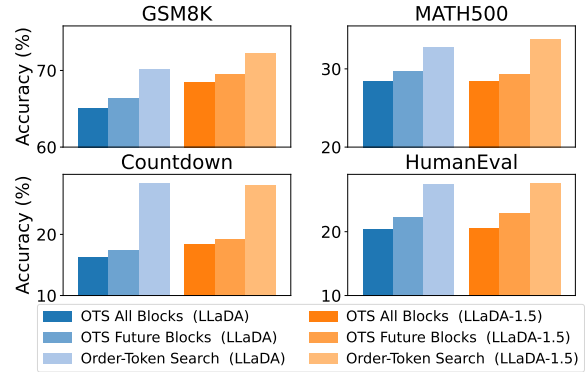


Figure 4: **Per-dataset average accuracy under different likelihood estimators for pruning.** Bars compare methods for LLaDA (blue), LLaDA-1.5 (orange), showing consistent gains of Order-Token Search on GSM8K, MATH500, Countdown, and HumanEval. See Table S6 for sequence-length-specific results.

strong baselines, particularly on MATH500, Countdown, and HumanEval, and AR + beam-search even achieves the second- or third-best overall average for both backbone models. However, Order-Token Search still yields consistent gains. Averaged over the two LLaDA variants, Order-Token Search achieves 71.2/33.3/28.2/27.5% on GSM8K/MATH500/Countdown/HumanEval, compared with 70.0/28.6/13.6/23.5% for AR + MV and 70.0/30.5/19.7/26.1% for AR + beam-search. Notably, AR + beam-search is the block_size = 1 special case of our framework, yet it remains worse than full Order-Token Search. This result highlights the benefit of searching jointly over generation order and token choices, rather than restricting search to token choices under a fixed left-to-right order.

5.3 Necessity of a Good Likelihood Estimator

Robust incremental scoring is essential for pruning. Figure 4 shows that even with the *same* OTS branching and pruning procedure, the choice of likelihood estimator has a large impact. Both OTS All Blocks and OTS Future Blocks underperform Order-Token Search. For LLaDA, Order-Token Search achieves 39.7% overall accuracy, whereas OTS All Blocks and OTS Future Blocks drop to 32.6% and 33.9%, with the largest degradation on Countdown (28.4% → 16.2% / 17.4%). The same trend holds for LLaDA-1.5: Order-Token Search reaches 40.4% overall, versus 34.0% / 35.3% for the ablations, again with a large Countdown gap (28.0% → 18.4% / 19.2%). Together, these results validate our estimator in Eq. 3: scoring in-

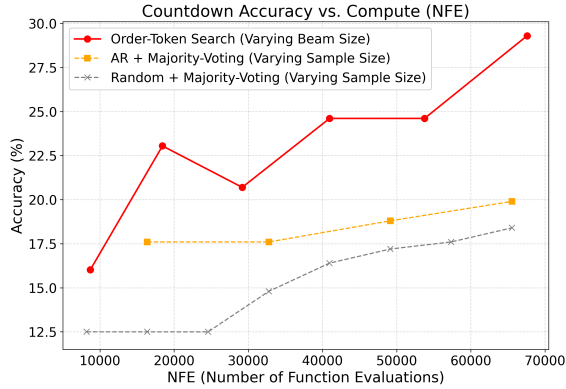


Figure 5: **Countdown accuracy versus test-time compute (NFE)** for Order-Token Search and majority-voting baselines. For each method, we sweep beam size or the number of samples, and set the largest configuration so the right-most points have approximately matched NFE. At this matched-compute point, OTS (beam size 6) achieves 29.3% accuracy, versus 19.9% for AR + MV and 18.4% for Random + MV, demonstrating more efficient returns from additional test-time compute than drawing more independent samples.

cremental denoising actions while conditioning on the model’s full-sequence predictive distribution yields a more stable and discriminative pruning signal than either joint history scoring within one forward pass or scoring without predicted future context. Appendix A.2.6 analyzes the pruning scores directly: OTS yields higher common token-level likelihoods than Future Blocks and All Blocks ($-30.28 / -34.75 / -38.81$ on average), matching their ordering in task accuracy.

5.4 Scaling with Test-Time Compute

Order-Token Search scales more efficiently with test-time compute than majority voting. In Figure 5, we study compute scaling on Countdown by varying the beam size of Order-Token Search and the sample count of majority-voting baselines under roughly matched budgets of Number of Function Evaluations (NFE). At the matched-compute frontier, Order-Token Search with beam size 6 reaches 29.3% accuracy, compared with 18.4% for Random + MV and 19.9% for AR + MV. Moreover, Order-Token Search improves steadily as beam size increases, from 16.0% at beam size 1 to 29.3% at beam size 6, whereas both majority-voting baselines show much smaller gains as more samples are added. These results indicate that Order-Token Search converts additional NFE into accuracy more effectively by searching jointly over generation or-

der and token choices, rather than aggregating independent trajectories with a voting heuristic.

6 Related Work

Diffusion Language Models. Initial developments in discrete diffusion models were established by D3PM (Austin et al., 2021a) and further progressed using masked token approaches (Sahoo et al., 2024; Nie et al., 2025a). Efficient versions such as Plaid (Gulrajani and Hashimoto, 2023) and SEDD (Lou et al., 2024) achieve performance comparable to GPT-2 (Radford et al., 2019), but their scalability still falls short of autoregressive models. The most recent scaling efforts include Dream (Ye et al., 2025), which adapts pre-trained autoregressive models into discrete diffusion models, and LLaDA (Nie et al., 2025b), which trains powerful masked diffusion language models from scratch.

Test-Time Strategies. A primary method to enhance diffusion models is to increase test-time compute, typically by using more denoising steps. Recent work has shown that expanding the inference-time sample space can guide generation toward high-reward outputs (Singhal et al., 2025; Kim et al., 2025b), with techniques like re-masking being introduced to scale the denoising process for masked diffusion models specifically (Wang et al., 2025). More broadly, language models have long benefited from test-time search methods such as beam search, speculative decoding (Leviathan et al., 2023; Xia et al., 2023), and contrastive decoding (Li et al., 2023), which improve on greedy decoding. However, these methods are tailored for the autoregressive paradigm, where search is limited to token choices under a fixed left-to-right order.

Our work addresses this limitation. The parallel, order-agnostic decoding of MDMs creates a joint search space over both token values and their generation order, which is inaccessible to autoregressive methods. Our algorithm, Order-Token Search, is designed for this new paradigm, leveraging parallel decoding to explore multiple generation trajectories and select outputs based on overall likelihood.

7 Conclusion

In this work, we revisited decoding for Diffusion Language Models (DLMs) through the lens of order-token trajectories: the intertwined decisions of which positions to denoise and which tokens to assign. We show that standard decoding methods exhibit a clear exploration-exploitation trade-

off. Low-confidence remasking exploits local confidence well, but collapses to a narrow set of greedy trajectories. On the contrary, random remasking increases trajectory diversity, but lacks a reliable mechanism for selecting good candidates.

Motivated by this observation, we introduce Order-Token Search, which performs search in the joint space of generation orders and token values. Order-Token Search maintains a beam of partially denoised candidates, expands them with diverse denoising actions, and prunes them using a likelihood estimator that scores incremental denoising decisions. Across math reasoning and coding benchmarks, this joint search yields consistent gains, matching or exceeding improvements typically associated with post-training methods such as *diffu*-GRPO. Overall, our results suggest that joint order-token search is a promising way to turn the multi-trajectory capacity of DLMs into stronger single-sample decoding performance.

Limitations

Our study is comprehensive, but has several limitations that we leave to future work. First, Order-Token Search improves decoding by using additional test-time compute for search and pruning, so its benefits come with higher inference cost. Second, our pruning strategy uses a fixed model-based likelihood estimator. While effective in our experiments, more adaptive or learned scoring functions could further improve candidate selection.

Acknowledgments

We thank Hao Chen, Meihua Dang, Charilaos Kanatsoulis, and Linhao Zhong for discussions and for providing feedback on our manuscript. We thank the anonymous reviewers for their feedback on improving the manuscript. We also gratefully acknowledge the support of NSF under Nos. CCF-1918940 (Expeditions), DMS-2327709 (IHBEM), IIS-2403318 (III); NIH under No. 1U24NS146314-01, Gates Foundation, Stanford Data Applications Initiative, Wu Tsai Neurosciences Institute, Stanford Institute for Human-Centered AI, Allen Institute, Genentech, SAP, SCBX, ARO (W911NF-21-1-0125), ONR (NO0014-23-1-2159), and the CZ Biohub. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding entities.

Use of LLMs Large language models (LLMs) were used solely to assist with grammar refinement

and writing clarity during the manuscript preparation stage. All technical ideas, experimental designs, model implementations, and analyses were conceived and executed by the authors without reliance on LLMs. The use of LLMs did not influence research outcomes, data interpretation, or reported results. We carefully reviewed and edited all text to ensure accuracy, originality, and compliance with ethical and academic standards.

References

- Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. 2025. [Block diffusion: Interpolating between autoregressive and diffusion language models](#). In *International Conference on Learning Representations*, volume 2025, pages 50726–50753.
- Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021a. [Structured denoising diffusion models in discrete state-spaces](#). In *Advances in Neural Information Processing Systems*, volume 34, pages 17981–17993. Curran Associates, Inc.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021b. [Program synthesis with large language models](#). *Preprint*, arXiv:2108.07732.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. [Evaluating large language models trained on code](#). *Preprint*, arXiv:2107.03374.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- Ishaan Gulrajani and Tatsunori B Hashimoto. 2023. [Likelihood-based diffusion language models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 16693–16715. Curran Associates, Inc.
- 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*, volume 1.

- Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. [Denoising diffusion probabilistic models](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.
- Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. 2025a. [Train for the worst, plan for the best: Understanding token ordering in masked diffusions](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 30749–30768. PMLR.
- Jaihoon Kim, Taehoon Yoon, Jisung Hwang, and Minhyuk Sung. 2025b. [Inference-time scaling for flow models via stochastic generation and rollover budget forcing](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 30830–30864. Curran Associates, Inc.
- Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. [Fast inference from transformers via speculative decoding](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.
- Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. [Contrastive decoding: Open-ended text generation as optimization](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let's verify step by step](#). In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.
- Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. [Discrete diffusion modeling by estimating the ratios of the data distribution](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32819–32848. PMLR.
- Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025a. [Scaling up masked diffusion models on text](#). In *International Conference on Learning Representations*, volume 2025, pages 82974–82997.
- Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan LI. 2025b. [Large language diffusion models](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 50608–50646. Curran Associates, Inc.
- Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. [Tinyzero](#). <https://github.com/Jiayi-Pan/TinyZero>. Accessed: 2025-01-24.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.
- Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 130136–130184. Curran Associates, Inc.
- Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. [Simplified and generalized masked diffusion for discrete data](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 103131–103167. Curran Associates, Inc.
- Raghav Singhal, Zachary Horvitz, Ryan Teehan, Mengye Ren, Zhou Yu, Kathleen Mckeown, and Rajesh Ranganath. 2025. [A general framework for inference-time scaling and steering of diffusion models](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 55810–55827. PMLR.
- Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. [Deep unsupervised learning using nonequilibrium thermodynamics](#). In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France. PMLR.
- Yang Song and Stefano Ermon. 2019. [Generative modeling by estimating gradients of the data distribution](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. [Score-based generative modeling through stochastic differential equations](#). In *International Conference on Learning Representations*.
- Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. [Musr: Testing the limits of chain-of-thought with multistep soft reasoning](#). In *International Conference on Learning Representations*, volume 2024, pages 14670–14728.
- Guanghan Wang, Yair Schiff, Subham Sahoo, and Volodymyr Kuleshov. 2025. [Remasking discrete diffusion models with inference-time scaling](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 147282–147339. Curran Associates, Inc.
- 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*.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. [Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925, Singapore. Association for Computational Linguistics.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. [Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?](#) In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 57654–57689. Curran Associates, Inc.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. [d1: Scaling reasoning in diffusion large language models via reinforcement learning](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 56729–56762. Curran Associates, Inc.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2026. [LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11425–11460, San Diego, California, United States. Association for Computational Linguistics.

A Appendix

A.1 Order-Token Search Algorithm

To make our proposed decoding strategy more concrete, we present the pseudocode of our [Order-Token Search](#), which illustrates how partially masked sequences are expanded, scored and pruned, exploring both the token space and order space.

A.2 Alternative Likelihood Estimators for OTS

This appendix details two scoring ablations of Order-Token Search (OTS) that isolate design choices in our robust likelihood estimator (Eq. 3). Both baselines keep the *same* OTS search/expansion procedure, and differ *only* in the pruning score used to rank candidates.

Notation. Let $\mathbf{x}_t$ and $\mathbf{x}_s$ ($0 \leq s < t \leq 1$) be the candidate states before/after an expansion, and let $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0 | \mathbf{x}_t)$ be the model’s full-sequence prediction used as context. We define three position sets:

- $\Delta(s, t) := \{i | \mathbf{x}_{t,i} = \text{MASK} \wedge \mathbf{x}_{s,i} \neq \text{MASK}\}$
(newly-revealed positions)
- $R(s) := \{i | \mathbf{x}_{s,i} \neq \text{MASK}\} \setminus \text{prompt positions}$
(revealed-so-far)
- $F(s) := \{i | \mathbf{x}_{s,i} = \text{MASK}\}$
(unrevealed future positions)

Let $\text{mask}(\mathbf{x}_0, S)$ denote replacing positions in S with MASK.

A.2.1 OTS All Blocks: Joint Likelihood of All Revealed Blocks

Idea. Instead of scoring only the newly-revealed blocks $\Delta(s, t)$ (Eq. 3), this baseline scores the *entire revealed prefix* $R(s)$ at each expansion, i.e., a harder joint infilling task.

Score. We replace Eq. 3 with:

$$s_{\text{ALL}}(\mathbf{x}_t; \mathbf{x}_s) = \mathbb{E}_{\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0 | \mathbf{x}_t)} \log p\left(\mathbf{x}_0^{R(s)} \mid \text{mask}(\mathbf{x}_0, R(s))\right). \quad (\text{S1})$$

Since $s_{\text{ALL}}(\mathbf{x}_t; \mathbf{x}_s)$ already evaluates *all revealed blocks so far*, we use it directly as the pruning score at each search boundary (i.e., we do not sum over intervals).

A.2.2 OTS Future Blocks: Likelihood Without Conditioning on Predicted Future

Idea. Our main scorer conditions the likelihood of $\Delta(s, t)$ on the *full* predicted $\mathbf{x}_0$, which includes predicted future content. This ablation removes that advantage by masking all unrevealed future positions $F(s)$ when scoring the current update.

Score. We score only the newly-revealed positions, but compute their likelihood using a context where future blocks are masked:

$$s_{\text{FUTURE}}(\mathbf{x}_t; \mathbf{x}_s) = \mathbb{E}_{\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0 | \mathbf{x}_t)} \log p\left(\mathbf{x}_0^{\Delta(s, t)} \mid \text{mask}(\mathbf{x}_0, \Delta(s, t) \cup F(s))\right). \quad (\text{S2})$$

This evaluates how well the model supports the current block using only the prompt and previously revealed blocks, without conditioning on any predicted future tokens. The total candidate score

Algorithm 1 Order-Token Search for Diffusion Language Models

```
1: Input: Prompt  $\mathbf{p}$ , model  $p_\theta$ , beam size  $K$ , generation length  $L$ , total steps  $S$ , search interval  $N$ ,  
   temperature  $\tau$ , number of blocks  $b$ .  
2: Initialize beam set  $\mathcal{B} \leftarrow \{(\mathbf{x}_i, \mathbf{s}[b], \text{score})\}$ , where  $\mathbf{x}_i = [\mathbf{p}; \text{MASK}^L]$ ,  $\mathbf{s}[0 : b - 1] = 0$ ,  $\text{score} = 0$   
    $\{K \text{ identical beams}\}$   
3: for step  $s \leftarrow 1$  to  $S$  do  
4:    $\mathbf{l} \leftarrow p_\theta(\mathcal{B}.\mathbf{x})$  {Get logits for all beams, shape:  $(K, L, V)$ }  
5:   if  $s \bmod N == 0$  then  
6:      $\mathcal{B}_{\text{candidates}} \leftarrow \emptyset$   
7:     for  $(\mathbf{x}, \mathbf{s}, \text{score}) \in \mathcal{B}$  do  
8:        $\text{block\_idx} \leftarrow \text{get\_current\_block\_index}(\mathbf{x})$  {Compatible with semi-AR generation}  
9:       for  $i \leftarrow 1$  to  $K$  do  
10:         $\tilde{\mathbf{l}} \leftarrow \text{add\_gumbel\_noise}(\mathbf{l}_x, \tau)$   
11:         $\mathbf{x}_0 \leftarrow \text{argmax}(\tilde{\mathbf{l}}, \text{dim} = -1)$  {Sample a candidate completion}  
12:         $\mathbf{x}_{\text{candidate}} \leftarrow \text{transfer\_tokens}(\mathbf{x}, \mathbf{x}_0, \mathbf{l}_x)$  {Only apply  $\frac{L}{S}$  predicted tokens}  
13:         $\mathbf{x}_{\text{full\_seq}} \leftarrow \text{transfer\_all\_tokens}(\mathbf{x}, \mathbf{x}_0)$  {Apply all predicted tokens}  
14:         $\mathbf{x}_{\text{masked}} \leftarrow \text{mask\_tokens}(\mathbf{x}_{\text{full\_seq}}, \text{block\_idx})$  {Mask the current block}  
15:         $\text{block\_score} \leftarrow \text{score\_block}(\mathbf{x}_{\text{masked}}, \text{block\_idx})$  {Score the sequence}  
16:         $\mathbf{s}[\text{block\_idx}] = \text{block\_score}$   
17:         $\text{score} = \text{sum}(\mathbf{s}[0 : \text{block\_idx}])$   
18:         $\mathcal{B}_{\text{candidates}} \leftarrow \mathcal{B}_{\text{candidates}} \cup \{(\mathbf{x}_{\text{candidate}}, \mathbf{s}, \text{score})\}$   
19:      end for  
20:    end for  
21:     $\mathcal{B} \leftarrow \text{top}_K(\mathcal{B}_{\text{candidates}})$  {Prune to the  $K$  best candidates}  
22:  else  
23:    for  $(\mathbf{x}, \_, \_) \in \mathcal{B}$  do  
24:       $\tilde{\mathbf{l}} \leftarrow \text{add\_gumbel\_noise}(\mathbf{l}_x, \tau)$   
25:       $\mathbf{x}_0 \leftarrow \text{argmax}(\tilde{\mathbf{l}}, \text{dim} = -1)$   
26:       $\mathbf{x} \leftarrow \text{transfer\_tokens}(\mathbf{x}, \mathbf{x}_0, \mathbf{l}_x)$   
27:    end for  
28:  end if  
29: end for  
30: Return: The sequence from  $\mathcal{B}$  with the highest final score.
```

is then accumulated over search-guided denoising steps, $\sum_{(s,t) \in \mathcal{I}} \mathcal{S}_{\text{FUTURE}}(\mathbf{x}_t; \mathbf{x}_s)$, matching the aggregation rule in Order-Token Search.

A.2.3 AR-style Likelihood: Left-to-Right Token Log-Likelihood

For comparison, we describe the traditional likelihood used in autoregressive (AR) language models. Given a partially generated sequence $\mathbf{x} = (x_1, \dots, x_L)$, an AR model factorizes the likelihood in a left-to-right manner:

$$p_{\text{AR}}(\mathbf{x}) = \prod_{i=1}^L p_\theta(x_i \mid x_{<i}), \quad (\text{S3})$$

where $x_{<i} := (x_1, \dots, x_{i-1})$. Accordingly, the standard log-likelihood score for a candidate se-

quence is

$$s_{\text{AR}}(\mathbf{x}) = \sum_{i=1}^L \log p_\theta(x_i \mid x_{<i}). \quad (\text{S4})$$

In AR decoding (e.g., beam search), candidates are ranked by accumulating token-level log-probabilities along the generation path. In our Order Search and Token Search baselines, we adopt an analogous AR-style score by summing log-likelihoods of the already revealed tokens in a forward pass.

A.2.4 Misalignment of AR-style likelihood in MDMs

Search-based decoding depends on accurate likelihood estimates for pruning, but naive AR-style

Table S1: Wall-clock time (in seconds) comparison on the Countdown dataset, averaged over all problems. This demonstrates that OTS (4 beams) is roughly 2-3x slower than a single heuristic-based remasking run, but about 2x faster than majority voting with 5 samples.

Method / Generation length	64	128	256	512
Heuristic-based remasking	1.55	3.19	6.60	14.52
+ Majority-voting (5 samples)	7.73	15.94	32.99	72.59
Order-Token Search (4 beams)	3.52	7.46	16.64	40.41

Table S2: **Accuracy of search algorithms with OTS or AR-style likelihood estimate (Seq Len 256)**. Bolded values indicate best performance. OTS consistently outperforms both Order Search and Token Search that adopt an AR-style likelihood estimate.

Decoding Method (Compute)	GSM8K	MATH500	Countdown	HumanEval
Token Search (3x)	8.5	3.8	0.0	0.6
Order Search (3x)	79.2	35.8	15.2	32.9
Order-Token Search (1x)	79.8	36.0	26.2	34.2

likelihood is mismatched to MDMs, which are trained to denoise *multiple positions in parallel* (Section 2.1). In Table S2, Order Search and Token Search instantiate this AR-style estimate by scoring partially denoised sequences via forward passes conditioned only on the currently revealed tokens. Both lag far behind Order-Token Search: on Countdown, Order-Token Search outperforms Order Search by 11%, while Token Search collapses to 0% accuracy. This indicates that directly porting AR beam-search heuristics to MDMs yields unreliable pruning signals; in contrast, Order-Token Search succeeds by pairing joint order-token exploration with a likelihood estimation function tailored to incremental denoising.

A.2.5 Ablation with Cumulative Token Probability

We further consider a simple likelihood-based pruning baseline, denoted CUMPROB, which keeps the same OTS branching and search procedure but ranks partial trajectories by the accumulated log-probability the model assigns to each committed token. Both methods thus draw likelihood information from the same underlying model under an identical branching procedure, isolating the effect of the pruning score itself.

As shown in Table S3, OTS attains the better overall average on both models (39.7% vs. 38.0% on LLaDA, 40.4% vs. 38.7% on LLaDA-1.5). The two are close on GSM8K, with the difference concentrated in MATH500, Countdown, and HumanEval. This suggests that how likelihood is es-

timated for partially denoised candidates affects pruning quality.

A.2.6 Likelihood Diagnostics for Pruning

Section 5.3 compares pruning estimators under a fixed branching and search procedure: OTS reaches 39.7% average accuracy on LLaDA against 33.9% for OTS Future Blocks and 32.6% for OTS All Blocks, and 40.4% against 35.3% and 34.0% on LLaDA-1.5. To relate these differences to the pruning signal itself, we record two decoding-time statistics. `sel_logp` is the accumulated selection score used for pruning; `tok_logp` accumulates the log-probabilities of committed tokens and is computed identically across methods, so it is comparable between estimators. Table S4 reports both for all datasets and generation lengths.

The likelihood ordering (OTS > OTS Future Blocks > OTS All Blocks) matches the accuracy ordering in nearly every setting, the sole exception being `tok_logp` on MATH500 at length 64. We do not claim that likelihood is a pointwise predictor of correctness; high-likelihood failures are discussed in Appendix A.9. The more limited reading is that the OTS estimator yields a more discriminative selection signal for partial trajectories, and substituting a weaker estimator lowers final accuracy.

A.3 Computation Complexity

To manage computational complexity, we deliberately structure the search using **block diffusion** (Arriola et al., 2025; Nie et al., 2025b). This avoids the prohibitive cost of a naive search at every step,

Table S3: **Ablation with cumulative token-probability pruning.** All methods use the same OTS branching procedure; only the pruning score differs. We report accuracy per generation length, per-dataset averages, and the overall average.

		GSM8K					MATH500					Countdown					HumanEval					Average
	Method / Seq Len	64	128	256	512	Avg	64	128	256	512	Avg	64	128	256	512	Avg	64	128	256	512	Avg	
LLaDA	CumProb	45.5	71.7	78.8	83.6	69.9	21.4	25.8	35.8	39.4	30.6	27.3	27.3	20.7	26.6	25.5	15.2	24.4	32.9	30.5	25.8	38.0
	OTS	45.6	71.6	79.8	83.3	70.1	22.4	30.4	36.0	42.4	32.8	27.7	34.4	26.2	25.4	28.4	15.2	23.2	34.2	37.2	27.4	39.7
LLaDA-1.5	CumProb	49.7	74.3	81.0	83.6	72.1	20.6	27.4	33.2	38.6	30.0	22.7	30.5	25.0	27.0	26.3	14.6	23.2	34.1	33.5	26.4	38.7
	OTS	48.4	74.5	81.7	84.0	72.2	24.4	30.8	37.4	42.4	33.8	27.7	31.3	23.8	29.3	28.0	15.2	22.6	36.0	36.6	27.6	40.4

Table S4: **Likelihood diagnostics under different pruning estimators.** We report the accumulated selection score (sel_logp) and the common committed-token log-probability diagnostic (tok_logp) for OTS, OTS Future Blocks, and OTS All Blocks. All methods use the same OTS branching/search procedure and inference budget, differing only in the likelihood estimator. Higher values (less negative) indicate higher likelihood.

Length	Dataset	Order-Token Search		OTS Future Blocks		OTS All Blocks	
		sel	tok	sel	tok	sel	tok
64	GSM8K	-107.19	-12.31	-141.19	-14.28	-183.86	-14.71
	MATH500	-96.09	-13.58	-139.45	-16.29	-170.08	-16.04
	Countdown	-111.14	-16.94	-122.63	-18.65	-164.80	-19.71
	HumanEval	-96.79	-12.40	-155.44	-13.88	-215.27	-14.36
	Avg.	-102.80	-13.81	-139.68	-15.78	-183.40	-16.21
128	GSM8K	-105.70	-14.15	-212.27	-17.20	-404.68	-17.89
	MATH500	-161.64	-26.60	-277.73	-27.23	-449.17	-31.98
	Countdown	-234.17	-37.61	-313.96	-42.65	-430.42	-43.11
	HumanEval	-164.44	-20.30	-278.59	-23.54	-469.94	-24.48
	Avg.	-166.49	-24.67	-270.64	-27.66	-438.44	-29.37
256	GSM8K	-164.07	-22.15	-395.92	-26.57	-933.59	-27.20
	MATH500	-206.86	-34.33	-464.81	-40.29	-944.65	-53.11
	Countdown	-350.08	-58.13	-513.98	-60.57	-933.52	-70.02
	HumanEval	-203.71	-25.20	-424.14	-32.48	-974.88	-36.19
	Avg.	-231.18	-34.95	-449.71	-39.98	-946.66	-46.63
512	GSM8K	-176.75	-26.80	-449.25	-29.25	-1268.91	-31.62
	MATH500	-253.13	-50.74	-742.50	-64.33	-1774.39	-74.81
	Countdown	-412.22	-80.27	-697.95	-89.57	-2028.25	-98.39
	HumanEval	-236.95	-32.97	-506.62	-39.20	-1388.61	-47.32
	Avg.	-269.76	-47.70	-599.08	-55.59	-1614.53	-63.04
Overall Avg.		-192.56	-30.28	-364.78	-34.75	-795.94	-38.81

which would incur a complexity of $O(S \cdot K^2 \cdot L)$, where S is the number of diffusion steps. The overhead of our approach is far lower. Let L be the generation length, K the beam size, and B the number of blocks. The total Number of Function Evaluations (NFE) for OTS is the sum of K independent denoising trajectories (costing $S \cdot K \cdot L$) and the likelihood evaluations for search, which are performed B times at block boundaries (costing $B \cdot K^2 \cdot L$).

Thus, the total NFE is $\text{NFE}(\text{OTS}) \approx S \cdot K \cdot L + B \cdot K^2 \cdot L$. In our main experimental setting (where $S = L/2$ and $B = L/32$), with a typical beam size $K \approx 4$, this simplifies to $\text{NFE}(\text{OTS}) \approx (L^2 \cdot K)/2 + (K^2 \cdot L^2)/32 \approx 2.5 \cdot L^2$. This is critically important, as it is directly comparable to the NFE of a standard majority-voting baseline with 5 samples: $\text{NFE}(\text{MV-5}) = S \cdot 5 \cdot L =$

$(L/2) \cdot 5 \cdot L = 2.5 \cdot L^2$. Therefore, OTS provides a structured joint search over the (order $\times$ token) space at roughly the same computational cost as a widely-used unstructured sampling baseline. After processing all blocks, the single best sequence is selected from the final K candidates based on the highest likelihood. To further validate our analysis, we measured wall-clock time on the Countdown dataset, averaging over all problems and comparing heuristic-based remasking (e.g., low-confidence remasking), naive majority voting (5 samples), and OTS with 4 beams. As shown in Table S1, our optimized implementation of OTS runs faster than the majority-voting baseline.

A.4 Comparison Study of Greedy Decoding and Order-Token Search under Identical Temperature

To further validate the effectiveness of our approach, we compare greedy decoding with Order-Token Search under identical temperature settings. This controlled setup rules out confounding factors and highlights the contribution of the search strategy itself. As reported in Table S5, Order-Token Search delivers a remarkable performance boost on the Countdown dataset, confirming that the method provides tangible gains beyond simple greedy decoding.

A.5 Additional Experimental Details

We provide further details on the experimental settings that complement the main results.

A.5.1 Order-Token Search Settings

Our **Order-Token Search** results (as shown in Table 1 and Table S2) is configured with beam sizes of $K \in \{3, 5, 8\}$ and block size of 32 along with a small search for the Gumbel noise temperature $\tau \in [0.2, 1.0]$, keeping in mind its role in balancing diversity and stability. As a general principle, a higher temperature introduces more diversity among the beams, but it can also risk destabilizing the token selection and decoding order. The settings used for our main experiments were chosen to maintain a reasonable balance between these factors.

In the main paper, we adopt the low-confidence remasking strategy together with the setting $\text{gen_len} = 2 \times \text{diffusion_steps}$; $\text{block_size} = 32$ for our baseline experiments. This configuration follows prior work (Zhao et al., 2025) and provides what can be regarded as a form of optimality: while it does not guarantee strict global optimality, it has been shown to yield a reasonably effective and competitive baseline under low-confidence conditions. Random-remasking majority-voting and Order-Token Search both use the same configuration. And we change $\text{block_size} = 1$ to simulate AR decoding on AR, AR majority-voting and AR Beam Search.

For the Order Search and Token Search experiments reported in Table S2, we use the configuration with $K = 3$.

A.5.2 Pass@k Evaluation Settings

Following Yue et al. (2025), we use $\text{pass}@k$ to characterize multi-sample reasoning coverage.

For our DLM experiments, we set the sampling temperature to 0.8 and use $\text{gen_len} = \text{block_size} = 256$. For autoregressive decoding, we use $\text{block_size} = 1$.

A.5.3 Computational Cost Analysis

Order Search generally search only on the decoding order of sequences based on the confidence of each position. This algorithm is designed based on the intuition that decoding order might change the ultimate accuracy. Therefore, at every step, we keep K positions that have the highest probability from model logits independently unmasked. We then perform a look-ahead at the next step to have K^2 candidate sequences each with one more position unmasked. We calculate the confidence score and keep the top- K candidates. In the experiment, we adopt the configuration of $K = 3, T = 0.0, \text{gen_len} = 256$ and the results also witness promising improvement.

However, the algorithm is computationally expensive, for it requires $K^2 \times \text{gen_len}$ forward passes in total. With $K = 3$ and $\text{gen_len} = 256$, this amounts to $3^2 \times 256 = 2304$ forward evaluations. In contrast, our Order-Token Search with $K = 5$ requires only $(128 \times 5) + (\frac{256}{32}) \times 25 = 840$ forward passes, where $256/32$ corresponds to the number of blocks and each block update involves 5×5 expansions.

Token Search is closely related to Order Search, but instead of expanding K positions at each step, it expands the top- K most confident tokens for a single position. Starting from K sequences, this again produces K^2 candidate sequences per step, leading to the same overall complexity of $K^2 \times \text{gen_len}$ forward passes. For instance, with $K = 3$ and $\text{gen_len} = 256$, Token Search also requires $3^2 \times 256 = 2304$ forward evaluations. Although the search space differs (token values vs. decoding order), the computational burden remains quadratic in K , making it substantially more expensive than our Order-Token Search, which scales only linearly with K .

A.6 Limitations on Sudoku Benchmarks

Table S7 indicates that Sudoku behaves qualitatively differently from the other benchmarks. Across both LLaDA and LLaDA-1.5, performance on Sudoku remains uniformly low across a wide range of decoding strategies, including autoregressive decoding, majority voting, beam search, and our proposed Order-Token Search. Notably, these

Table S5: Countdown task performance under different configurations.

Seq Len(L), Diffusion Steps(S), Beam Size(K), Temperature(T)	Accuracy (%)
L=128,S=64,K=1,T=0.0	20.7
L=128,S=64,K=1,T=0.4	22.7
L=128,S=64,K=5,T=0.4	34.4

Table S6: Performance Comparison of OTS Variants and Order-Token Search.

		GSM8K					MATH500					Countdown					HumanEval					Average
Method / Seq Len		64	128	256	512	Avg	64	128	256	512	Avg	64	128	256	512	Avg	64	128	256	512	Avg	
LLaDA	OTS All Blocks	42.8	68.4	74.5	74.8	65.1	21.0	27.2	32.8	33.0	28.5	11.7	16.8	19.9	16.4	16.2	11.6	18.3	23.8	28.1	20.4	32.6
	OTS Future Blocks	43.3	66.8	76.2	78.9	66.3	22.1	28.4	31.6	36.8	29.7	15.2	18.0	20.3	16.0	17.4	12.8	16.5	28.1	31.7	22.3	33.9
	Order-Token Search	45.6	71.6	79.8	83.3	70.1	22.4	30.4	36.0	42.4	32.8	27.7	34.4	26.2	25.4	28.4	15.2	23.2	34.2	37.2	27.4	39.7
LLaDA-1.5	OTS All Blocks	45.6	70.7	78.9	78.3	68.4	20.8	26.4	31.8	34.6	28.4	9.0	21.5	20.7	22.3	18.4	13.4	16.6	25.6	26.8	20.6	34.0
	OTS Future Blocks	45.3	71.3	80.5	80.7	69.5	19.8	27.2	34.8	35.6	29.4	13.7	21.9	21.5	19.5	19.2	12.2	17.1	27.4	34.8	22.9	35.3
	Order-Token Search	48.4	74.5	81.7	84.0	72.2	24.4	30.8	37.4	42.4	33.8	27.7	31.3	23.8	29.3	28.0	15.2	22.6	36.0	36.6	27.6	40.4

methods do not exceed a simple random baseline: for a 4×4 Sudoku, a uniformly random fill attains 25% expected cell accuracy.

Overall, this pattern suggests that Sudoku performance is primarily limited by how well the underlying model internalizes the task’s global, hard constraints, rather than by the choice of inference-time decoding alone. Sudoku requires enforcing row, column, and subgrid consistency, and such constraint structure may not be reliably captured by token likelihoods. When the model’s scores are weakly correlated with constraint satisfaction, search-based decoding has limited leverage: exploring decoding trajectories mainly reorders and recombines signals already present in the model, without introducing new structural knowledge.

Order-Token Search is designed to improve reasoning by exploring and pruning *order-token trajectories* under a fixed model. It is most effective in regimes where the model assigns informative relative likelihoods to intermediate states that reflect partial correctness. On Sudoku, however, the model appears to assign nearly uninformative or occasionally misleading scores to constraint-violating states, which can cause diverse decoding methods (including AR, beam search, and Order-Token Search) to saturate at similarly low accuracy.

We therefore view Sudoku as a boundary case for inference-time structured search: decoding can amplify and select among *existing* reasoning signals, but its gains depend on the presence of reliable constraint-related signal in the model’s scoring. Improving performance on Sudoku likely requires model-level changes, such as explicit constraint modeling or targeted training, rather than more sophisticated decoding alone.

For this reason, we report Sudoku separately and exclude it from aggregate conclusions about reasoning improvements, treating it as a diagnostic case study of model capability on global constraint satisfaction.

A.7 Sensitivity of Order-Token Search to Block Size

Our scoring function $s(x_t; x_s)$ is explicitly designed to be stable across a range of block sizes. Conceptually, the block size controls a bias-variance trade-off in likelihood estimation. When the block is larger, the model must jointly predict more tokens at once, making each scoring step harder but fewer in number. When the block is smaller, each prediction is easier and closer to the MDM training distribution—where the model typically denoises a limited number of masks at a time—but search is invoked more frequently. In all cases, the score of a candidate is the sum of these incremental block-level log-likelihoods over its full generation path (Eq. 3), so changing the block size simply changes how finely this path-wise likelihood is decomposed, not the underlying distribution being estimated. We therefore view the block size primarily as an efficiency and granularity knob rather than a fragile hyperparameter for the scoring rule itself.

In practice, we find that Order-Token Search is not highly sensitive to the exact block size within a reasonable range. On MATH500 with generation length 128, sweeping the block size from 1 to 128 yields accuracies between 23.0% and 28.0%. Across block sizes 2–64, performance stays in a narrow band around 26.5% (approximately 26.5 ± 1.5), and all such settings significantly outperform the degenerate cases of block

Table S7: **4×4 Sudoku result.** We report Sudoku average per-cell accuracy across multiple generation lengths for two base models (LLaDA and LLaDA-1.5). Bolded values indicate the best performance, and underlined values indicate the second best. We treat Sudoku as an outlier relative to the math/coding benchmarks in Table 1. *Reference:* a completely random method would fill in a 4×4 Sudoku with 25% of the cells correct in expectation.

Model	Method / Seq Len	Sudoku				
		64	128	256	512	Avg
LLaDA	Low-confidence	8.5	11.7	6.5	5.5	8.1
	Low-conf + MV	8.9	10.3	8.5	7.2	8.7
	Random + MV	6.4	6.5	8.4	6.9	7.1
	AR	14.7	15.9	13.6	12.0	<u>14.1</u>
	AR + MV	14.4	15.9	12.1	14.3	14.2
	AR + beam-search	13.2	17.7	10.6	8.2	12.4
	OTS All Blocks	6.5	6.9	7.5	6.9	6.9
	OTS Future Blocks	7.0	7.7	7.4	6.9	7.3
	Order-Token Search	10.1	11.7	8.5	7.4	9.4
LLaDA-1.5	Low-confidence	13.5	15.6	11.4	10.3	12.7
	Low-conf + MV	13.7	14.9	10.8	12.6	13.0
	Random + MV	11.1	11.3	9.3	9.2	10.2
	AR	14.5	15.8	14.5	13.6	<u>14.6</u>
	AR + MV	14.6	16.6	13.2	13.1	14.4
	AR + beam-search	17.5	16.8	13.5	11.0	14.7
	OTS All Blocks	9.3	10.5	9.6	9.7	9.8
	OTS Future Blocks	9.6	11.1	8.3	8.5	9.4
	Order-Token Search	13.8	16.1	11.2	9.9	12.8

size 1 and 128, where Order-Token Search either loses the order space entirely (block size 1) or forces the model to effectively denoise the entire sequence in one shot (block size 128). This empirical plateau for intermediate block sizes matches the bias–variance trade-off discussed above and supports the view that block size primarily controls the efficiency and granularity of search rather than acting as a delicate tuning parameter. The full sweep is visualized in Figure S1.

A.8 Case Study: A Search Instance

Figure S2 provides a qualitative analysis of Order-Token Search’s search trajectory on a Countdown task requiring the combination of numbers (87, 28, 35) to reach a target value of 94. This particular problem exemplifies a case where the low-confidence remasking baseline fails, as it greedily commits to the locally plausible but ultimately incorrect path beginning with $87 + 28 = 115$. However, this path cannot yield the target 94 using the remaining number 35, since 115 ± 35 results in values (80 or 150) distant from the solution.

Order-Token Search overcomes this limitation

by maintaining multiple candidate paths simultaneously. While the addition-based path is explored, the algorithm also evaluates the alternative $87 - 28 = 59$ trajectory. The dedicated likelihood estimation correctly identifies the subtraction path as superior when $59 + 35$ precisely yields the target 94. This case demonstrates how Order-Token Search’s joint exploration of generation orders and token space enables escape from local optima that trap greedy methods, systematically identifying globally correct solutions through parallel hypothesis testing.

A.9 Failure Cases: High-Likelihood but Incorrect Trajectories

We examine OTS-selected trajectories that attain high likelihood yet are incorrect. For each case we report its sequence log-likelihood $\log L$ and the fraction of *correct* trajectories from the same configuration that the likelihood exceeds. When this is high, likelihood alone could not have separated the failure from the successes.

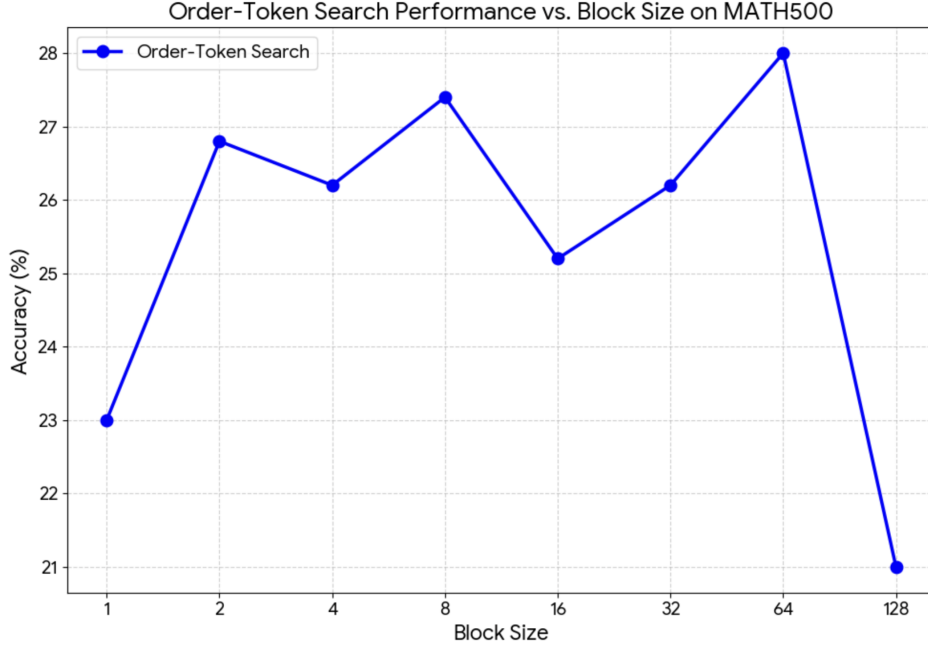


Figure S1: **Effect of block size on OTS accuracy** on MATH500 with generation length 128. Accuracy remains stable for block sizes 2–64, while the degenerate settings of block size 1 and 128 significantly underperform, confirming that block size mainly acts as an efficiency and granularity knob.

A.9.1 Math: fluent derivations with a local arithmetic slip

Example 1. $\log L = -32.60$; exceeds 33% of correct trajectories. With $f(n) = \lfloor n \rfloor$ if $n \geq 4$, $\lceil n \rceil$ otherwise, find $f(\frac{\pi}{3}) + f(\sqrt{45}) + f(8^{2/3})$. Ground truth 12; model 14.

1. For $\frac{\pi}{3} < 4$, $f(\frac{\pi}{3}) = \lceil \frac{\pi}{3} \rceil = 2$. 2. For $\sqrt{45} \geq 4$, $f(\sqrt{45}) = \lfloor \sqrt{45} \rfloor = 6$. 3. For $8^{2/3} \geq 4$, $f(8^{2/3}) = \lfloor 8^{2/3} \rfloor = 6$.
Adding these together: $2 + 6 + 6 = 14$.

Everything is correct except the single slip $8^{2/3} = 4$, not 6.

Example 2. $\log L = -14.79$; exceeds 78% of correct trajectories. Express $5^5 \div 5^4 - 5^3 + 5^2 \cdot 5^1$ as an integer. Ground truth 5; model -95.

1. $5^5 \div 5^4 = 5^{5-4} = 5$. 2. $5^3 = 125$. 3. $5^2 \cdot 5^1 = 5^2 = 25$.
Substituting back: $5 - 125 + 25 = -120 + 25 = -95$.

The structure and downstream arithmetic are flawless; step 3 drops the exponent addition ($5^2 \cdot 5^1 = 125$).

A.9.2 Countdown: hits the target but violates the constraints

Example 3. $\log L = -25.81$; exceeds 86% of correct trajectories. Numbers $\{49, 44, 7\}$, target 86; model outputs $93 - 7 (= 86)$, but $93 \notin \text{set}$.

First, let's try adding 44 and 49: $44 + 49 = 93$.
Next, let's subtract 7 from 93: $93 - 7 = 86$. Answer: 93 - 7

The reasoning finds the valid solution $(44 + 49) - 7$, but the final expression treats the intermediate 93 as an input.

Example 4. $\log L = -9.58$; exceeds *all* correct Countdown trajectories in its configuration. Numbers $\{28, 54, 99\}$, target 73; model outputs $99 - 26 (= 73)$, but $26 \notin \text{set}$.

We need to find a way to combine the numbers 28, 54, and 99 to get 73 using the operations $+$, $-$, $*$, and $/$. Answer: 99 - 26

The expression invents 26 (implicitly $54 - 28$); the valid form is $99 - 54 + 28$.

A.9.3 Discussion

Likelihood measures model support for a trajectory, not arithmetic correctness or constraint satisfaction:

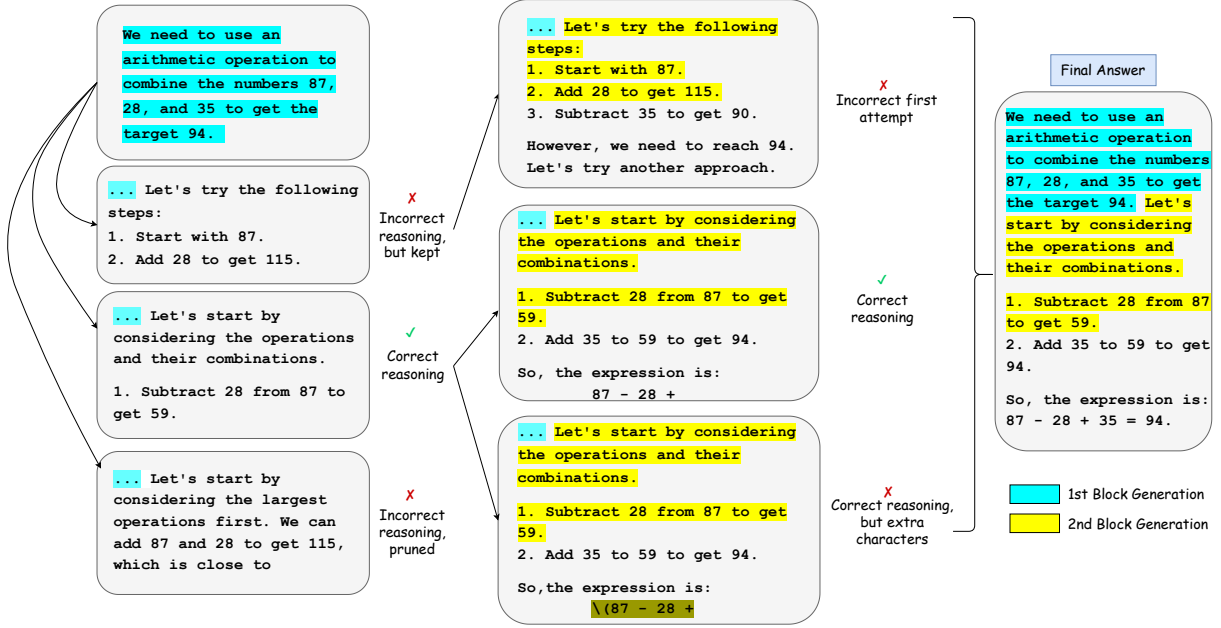


Figure S2: Case study of search trajectories for a sampled Countdown problem. Each box depicts an independently generated candidate sequence with arrows denoting the parent-child relationship in block diffusion. Order-Token Search evaluates each candidate and decides whether to move forward with its prefix sequence. Its likelihood criterion successfully pruned out inferior candidates that contains incorrect reasoning or syntactical errors, ultimately retaining only high-scoring candidates that lead to the correct solution.

Table S8: **Performance on MuSR.** We evaluate LLaDA-8B-Instruct with a generation length of 512. Bolded values indicate the best performance and underlined values indicate the second best.

Method	512
Low-confidence	52.8
Random remasking	41.3
Random + MV	48.3
AR	47.8
AR + MV	<u>53.7</u>
Order-Token Search	56.8

several incorrect trajectories score above 80–100% of correct ones, so no likelihood threshold could separate them. This does not contradict its role in OTS—as the ablation in Figure 4 shows, a stronger likelihood estimator provides a more reliable *aggregate* pruning signal under an identical search procedure and budget, without being a pointwise verifier.

A.10 MuSR: Longer-Context Soft Reasoning

To evaluate whether Order-Token Search generalizes beyond mathematical and code reasoning, we additionally evaluate it on MuSR (Sprague et al., 2024), a benchmark for multi-step soft reasoning over natural-language narratives. We evaluate LLaDA-8B-Instruct with a generation length of

512, providing a generation budget suitable for the longer reasoning chains in MuSR.

Table S8 reports the results. Order-Token Search achieves 56.8% accuracy, outperforming all considered baselines, including AR with majority voting (53.7%). This result provides additional evidence that joint search over generation order and token space is effective beyond mathematical and code reasoning tasks.

A.11 Detailed Diversity Analysis

Section 3 reports diversity metrics averaged over the eight dataset–model settings. Here we provide the complete per-dataset and per-model results in Table S9. We measure four complementary notions of diversity: token-position entropy, average Distinct-3/4, pairwise edit distance, and answer entropy.

The aggregate trend is consistent across the two backbones: AR and random remasking generally produce substantially more diverse trajectories than low-confidence remasking, with random remasking achieving the highest average value for all four metrics. This trend is particularly pronounced for token-position entropy and pairwise edit distance, which directly capture variation in generation order and generated trajectories.

Table S9: **Detailed diversity measurements across datasets and models.** We report four complementary diversity metrics for low-confidence remasking, AR decoding, and random remasking. Higher values indicate greater diversity. Bold values denote the highest value within each model–dataset setting and metric.

Model	Benchmark	Token-position entropy $\uparrow$			Avg. Distinct-3/4 $\uparrow$			Pairwise edit distance $\uparrow$			Answer entropy $\uparrow$		
		Low-conf.	AR	Random	Low-conf.	AR	Random	Low-conf.	AR	Random	Low-conf.	AR	Random
LLaDA	Countdown	0.625	0.902	0.927	0.515	0.447	0.664	0.152	0.856	0.871	0.571	0.915	0.940
	GSM8K	0.611	0.871	0.895	0.060	0.156	0.437	0.346	0.680	0.800	0.298	0.293	0.633
	MATH500	0.674	0.869	0.889	0.079	0.274	0.536	0.423	0.742	0.821	0.320	0.604	0.731
	HumanEval	0.242	0.702	0.869	0.035	0.152	0.417	0.226	0.487	0.704	0.322	0.492	0.557
LLaDA-1.5	Countdown	0.072	0.906	0.929	0.928	0.453	0.675	0.012	0.859	0.873	0.184	0.911	0.937
	GSM8K	0.565	0.876	0.900	0.122	0.164	0.449	0.348	0.691	0.809	0.378	0.314	0.682
	MATH500	0.674	0.875	0.894	0.089	0.291	0.552	0.436	0.757	0.830	0.370	0.621	0.748
	HumanEval	0.184	0.719	0.870	0.079	0.168	0.426	0.337	0.501	0.703	0.298	0.509	0.565
Overall Avg.		0.456	0.840	0.897	0.238	0.263	0.520	0.285	0.697	0.801	0.343	0.582	0.724

Table S10: **Dream7B performance on mathematics and coding benchmarks.** We report accuracy at generation lengths 128 and 256; dataset-level and overall averages are computed over these two lengths. Bolded values indicate the best performance and underlined values the second best. OTS attains the highest overall average accuracy.

Method	GSM8K			MATH500			Countdown			HumanEval			Overall Avg.
	128	256	Avg.	128	256	Avg.	128	256	Avg.	128	256	Avg.	
Low-confidence	65.7	75.7	<u>70.7</u>	30.4	33.2	31.8	23.8	20.3	22.1	32.3	34.2	33.2	39.5
Low-conf + MV	64.3	75.3	69.8	29.2	35.2	32.2	28.5	27.3	27.9	33.6	36.6	35.1	41.3
Random remasking	49.2	52.2	50.7	17.6	18.8	18.2	14.8	10.6	12.7	22.0	22.6	22.3	26.0
Random + MV	59.2	64.9	62.1	20.6	24.2	22.4	14.1	11.7	12.9	17.7	18.9	18.3	28.9
AR	51.7	77.5	64.6	32.4	36.6	34.5	23.4	24.6	24.0	34.8	51.8	43.3	41.6
AR + MV	52.9	79.7	66.3	31.6	40.0	35.8	23.4	25.4	<u>24.4</u>	33.5	51.2	<u>42.4</u>	<u>42.2</u>
Order-Token Search	65.7	76.3	71.0	32.0	37.2	<u>34.6</u>	25.8	30.1	27.9	36.6	37.8	37.2	42.7

A.12 OTS Beyond the LLaDA Family

To test whether OTS extends beyond the LLaDA family, we evaluate it on Dream7B (Ye et al., 2025), a diffusion language model with a different training recipe. We evaluate at generation lengths 128 and 256, and compute all averages over these two lengths. As shown in Table S10, OTS attains the highest overall average, 42.7% against 42.2% for AR + MV, consistent with the LLaDA and LLaDA-1.5 results in Table 1. This supports that the order-token search formulation is not specific to the LLaDA family.