Order-Marginalized Scoring for Any-Order Autoregressive Models

Arthur Deng* **Sebastian Thrun**
Department of Computer Science
Stanford University

Abstract

Any-order autoregressive models are trained under an objective that is symmetric over decoding orders, but at sampling time a single order must be committed to. A common task with such models is to evaluate the log-probability of a candidate sequence. We show that this evaluation depends substantially on the decoding order used to compute it: for a model trained on a synthetic grammar, different orders yield different most-likely completions; for a model pre-trained on OpenWebText, different orders rank the same candidates differently on most inputs. We characterize this phenomenon and propose Order-Marginalized Scoring (OMS), a simple Monte Carlo estimator of the per-order log-probability averaged over decoding orders. We use it in two algorithms: a reranker over a fixed candidate pool, and a stochastic best-of-N decoder. OMS helps on two structurally order-agnostic tasks: on Sudoku puzzles with 28 given digits, we show that OMS improves exact-match accuracy from 67% with single-order scoring to 85%; on a zero-shot protein-stability benchmark, OMS improves the Spearman correlation between log-probability scores and experimental folding stabilities for 335 out of 337 protein families. On text infilling, however, OMS does not improve accuracy over reranking with a single random order. These results suggest that OMS is a useful alternative to per-order scoring for order-agnostic tasks where log-likelihood tracks the quantity of interest.[1]

1 Introduction

Any-order autoregressive models (AO-ARMs) generalize the canonical left-to-right factorization of autoregressive sequence models by training a single network on every conditional $p(\cdot \mid x_S)$ for any subset S of revealed positions [1, 2]. This flexibility is useful for tasks like Sudoku completion, fixed-backbone protein sequence design, and text infilling, where no single decoding order is canonical [3, 4]. The any-order training objective averages the per-order log-likelihood $\log p_\sigma(x) = \sum_t \log p(x_{\sigma(t)} \mid x_{\sigma(<t)})$ over a uniform distribution on decoding orders σ. At inference time, the model is queried under a *single* order, and a growing line of work studies how to choose this order adaptively [4–8]. AO-ARMs are in turn generalized by masked diffusion models [MDMs; 2, 9], which can decode several positions per step and reduce to AO-ARMs when they decode one [10]; we study this one-position setting throughout (§3.1).

In this paper, we show that the choice of decoding order matters not only during sampling but also when *scoring* the log-probability of a generated sequence. Scoring a candidate sequence under a learned model is a basic operation in many downstream tasks. For example, the log-probability of a protein sequence under a fixed-backbone inverse-folding model is a standard "zero-shot" predictor for protein thermostability [3, 11, 12]. One might think that the choice of σ used for scoring would

*Correspondence to lxdeng@stanford.edu.

[1] Code is available at https://github.com/LDeng0205/order-marginalized-scoring.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

be unimportant since the training objective of the model incorporates order-invariance. We find that this is not the case in practice for a model trained on a synthetic grammar and a model pre-trained on OpenWebText [2]. Consequently, the sequence with highest log-probability under a decoding order, which we call "per-order argmax", is different across decoding orders. Scoring under a single random σ is a noisy proxy for the model's actual belief about a candidate, and rerankers built on this proxy can be far from optimal.

We propose order-marginalized scoring (OMS), a simple Monte Carlo (MC) estimator of $\tilde{\mu}(x) := \mathbb{E}_\sigma[\log p_\sigma(x)]$, the average per-order log-likelihood. We use OMS as the scoring rule in two algorithms. The first, OMS-rerank, picks the highest-$\tilde{\mu}$ candidate from an externally generated pool. The second, OMS-bon (for best-of-N), generates the pool itself by combining an adaptive position selector from Kim et al. [4] with stochastic token sampling, then applies OMS-rerank.

We find that OMS improves Sudoku completion accuracy and zero-shot protein-stability prediction, two tasks where the model's log-likelihood tracks the quantity of interest. Both of these problems are structurally independent of decoding orders, which makes an order-free scoring target natural. On text infilling, however, OMS does not improve accuracy over reranking with a single random order, even though reranking by log-probability itself helps there. The benefit of averaging over orders is thus problem-dependent; characterizing when it helps remains an important open question beyond the scope of this work.

Contributions.

1. **Phenomenon.** On a synthetic probabilistic context-free grammar (PCFG) dataset and on a published 170M-parameter MDM checkpoint trained on web data, we document that different random decoding orders produce different argmaxes on most prompts, and that finite candidate pools are reranked differently under different orders. Proposition 1 shows that per-order argmaxes can differ in arbitrarily many positions, while the log-probability disagreement is arbitrarily small but positive.

2. **Method.** We characterize $\tilde{\mu}$ as the log-density, up to an additive constant, of the unique reverse Kullback–Leibler (KL) barycenter of the per-order family $\{p_\sigma\}_\sigma$ (Proposition 2), and bound the number of MC samples needed for accurate estimation (Proposition 3). We propose two algorithms, OMS-rerank (Algorithm 1) and OMS-bon (Algorithm 2), that target $\tilde{\mu}$ at decoding time.

3. **Experiments.** On the PCFG, reranking the candidate pool by model log-probability closes most of the gap between the per-order baseline and the oracle upper bound. On Sudoku, OMS-bon substantially improves over the strongest single-order baseline at all hint counts considered. On the Megascale protein-stability benchmark with ProteinMPNN, OMS-rerank with orders shared across a wild type's mutants improves Spearman correlation between log-probability and experimental folding stability over a single shared order. On text infilling with MDLM-OWT, OMS-rerank does not improve F1 over reranking under a single random order, suggesting that the benefit of averaging over orders is problem-dependent.

2 Related work

Any-order autoregressive models and decoding orders. AO-ARMs [13] train one network on the conditionals of every decoding order, and Hoogeboom et al. [1] showed that they share their training objective with absorbing-state masked diffusion [9] under a uniform single-token unmasking schedule. Under a masking noise process, the training objectives and sampling distributions of masked diffusion and flow-matching models are equivalent to those of AO-ARMs [10]. Prior heuristics for choosing which positions to unmask next include MaskGIT-style confidence picking [5], top-prob and its top-prob-margin refinement [4], and quasi-random Halton schedules [6]; learned schedulers instead train reinforcement learning (RL) policies to select positions [7, 8]. Horvitz et al. [14] observe that entropy-based any-order decoding often underperforms left-to-right (L2R) decoding on math and code tasks. Our work targets a different question: rather than asking which order to sample a sequence from, we ask how to score the log-probability of a candidate sequence in a way that does not depend on a single order.

Post-hoc scoring. Several recent papers also study post-hoc scoring for samples from masked diffusion models. Chen et al. [15] score candidate trajectories by a denoising-entropy criterion. Shen et al. [16] jointly search over the decoding order and tokens to maximize a likelihood-based score of the denoising actions along each trajectory. Lee et al. [17] and Yang et al. [18] use sequence-level lookahead and information gain over future masks, respectively. Cao et al. [19] switch between beam search and parallel decoding based on confidence, and Wu and Huang [20] score unmasking decisions by the inter-step KL stability of the model's predictions. These methods score a single trajectory or candidate, by its likelihood or by heuristic measures of uncertainty or stability, to generate better samples. Our scoring rule is different in kind: we score candidates by the order-marginalized log-likelihood $\tilde{\mu}$, whose negative expectation is the any-order training objective, which MDMs share under a masking noise process.

Nie et al. [21] pick multiple-choice answers by Monte Carlo estimates of a sequence's log-likelihood averaged over different orders, an estimator they adopt for its lower variance. OMS-rerank targets the same quantity; we compare it with single-order scoring, analyze why it can do better, and use it inside best-of-N decoding.

3 Phenomenon: per-order argmaxes and rankings disagree

3.1 Setup and notation

Let $x \in \mathcal{V}^T$ be a sequence of length T over a vocabulary $\mathcal{V}$, and let $\mathfrak{S}_T$ denote the set of $T!$ permutations of $[T]$. We write $\sigma \in \mathfrak{S}_T$ for a *decoding order* (also called an *unmasking order*), and $\Delta(\mathcal{V}) := \{p \in \mathbb{R}^{|\mathcal{V}|}_{\geq 0} : \sum_{v\in\mathcal{V}} p(v) = 1\}$ for the probability simplex over the vocabulary. Both AO-ARMs and absorbing-state MDMs parameterize a single shared family of conditionals

$$p_\theta(\cdot \mid x_S) \in \Delta(\mathcal{V}), \qquad S \subseteq [T],\ x_S \in \mathcal{V}^{|S|}, \tag{1}$$

which predict a distribution over the next token given any subset S of revealed positions and their values x_S. We assume these conditionals are strictly positive, as for softmax outputs. Because the same network is queried for every S, both objects induce a family of *order-conditional* joint distributions: for $\sigma \in \mathfrak{S}_T$, writing $\sigma(< t) := \{\sigma(0), \ldots, \sigma(t-1)\}$ for the prefix already revealed at step t (with $\sigma(<0) = \emptyset$),

$$\log p_\sigma(x) \;=\; \sum_{t=0}^{T-1} \log p_\theta\big(x_{\sigma(t)} \mid x_{\sigma(<t)}\big)\,. \tag{2}$$

Define the *order-marginalized log-likelihood* and the *per-example order disagreement*:

$$\tilde{\mu}(x) := \mathbb{E}_{\sigma\sim\mathrm{Unif}(\mathfrak{S}_T)}[\log p_\sigma(x)], \qquad \varepsilon(x) := \sup_{\sigma\in\mathfrak{S}_T} \big|\log p_\sigma(x) - \tilde{\mu}(x)\big|. \tag{3}$$

We also define the global *per-order disagreement* $\varepsilon := \sup_{x\in\mathcal{V}^T} \varepsilon(x)$. The any-order training objective of AO-ARMs [1, 13] is $\mathbb{E}_{x\sim p_{\mathrm{data}}}[-\tilde{\mu}(x)]$, an expectation of the per-order negative log-likelihood under a uniform distribution over decoding orders $\sigma \in \mathfrak{S}_T$. We call the model *well-specified* when the conditionals (1) are those of a single joint distribution; equivalently, $\varepsilon = 0$. Under a masking noise process and for a denoiser that does not condition on the time step, MDM training objectives reduce to $\mathbb{E}_{x\sim p_{\mathrm{data}}}[-\tilde{\mu}(x)]$ for suitable time weightings, and exact MDM sampling is AO-ARM decoding in a uniformly random order [10]. We treat MDMs throughout as samplers that unmask one position per step; under this convention an MDM run with a fixed order σ is exactly the AO-ARM defined by Equation (2). In Appendix B, we provide initial sketches that generalize this formulation to block decoding.

3.2 PCFG: per-order modes can disagree

We first study the phenomenon in a synthetic setting in which the data distribution is known in closed form. We define a topic-mixture Probabilistic Context-Free Grammar (PCFG) chosen so that the data mode is computable in closed form, and train a small MDM on it to convergence (Appendix E). To diagnose order dependence, we search for the most likely sequence for a random unmasking order on a given prompt (prefix). For a fixed prompt, we compare the "per-order argmax" sequence under different decoding orders. We identify the per-order argmax under the model using beam search

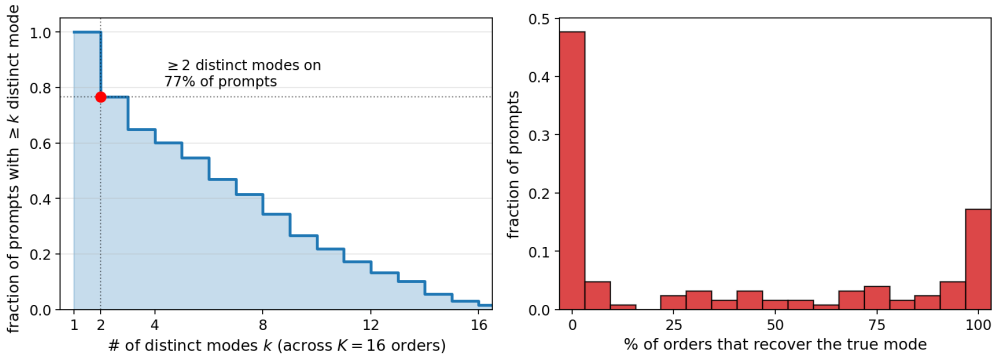


Figure 1: On a synthetic PCFG dataset for which the true data mode is computable in closed form, we compare beam-search argmaxes across $K{=}16$ random decoding orders on 128 random length-4 prefixes. **Left:** Complementary CDF of the number of distinct per-order argmaxes per prompt. A majority of prompts (77%) produce more than one distinct argmax, with mean 5.78 distinct argmaxes per prompt. **Right:** Distribution over prompts of the fraction of orders that recover the closed-form data mode. The distribution is bimodal: on 48% of prompts no order finds the data mode, and on 17% every order does.

(Appendix D). We ask two questions: (1) how many distinct per-order argmaxes are produced by different decoding orders, and (2) how often the per-order argmaxes recover the closed-form data mode.

We diagnose order dependence on 128 random length-4 prefixes, evaluating $K{=}16$ unmasking orders per prefix. The left panel of Figure 1 shows the distribution of the number of distinct per-order argmaxes per prompt: $98/128$ prompts produce two or more distinct argmaxes under 16 unmasking orders; the mean number of distinct argmaxes per prompt is 5.78. Two prompts produce a different argmax under every decoding order. These results show that per-order argmaxes disagree on most prompts, even in a synthetic setting where the model fits the data distribution closely. The right panel of Figure 1 shows the distribution over prompts of the fraction of orders that recover the closed-form data mode. This distribution is strongly bimodal: on 48% of prompts no order finds the data mode, and on 17% every order does. This demonstrates that on a simple synthetic grammar, the per-order argmaxes of a model can disagree substantially, and, consequently, the recovery of the true data mode becomes challenging.

A natural question is whether the per-order argmaxes of a near-perfect model can disagree by an arbitrary amount, or whether the disagreement must shrink with the per-order log-probability gap. The following proposition shows that the disagreement can be arbitrarily large.

Proposition 1 (Argmax pathology)**.** *Let $T \geq 2$ be the sequence length, $|\mathcal{V}| \geq 2$ the vocabulary size, and let $d \in \{2, 4, \ldots, T\}$ be any even Hamming distance. For every $\varepsilon > 0$ there exists an MDM with $\sup_{x,\sigma} |\log p_\sigma(x) - \tilde{\mu}(x)| \leq \varepsilon$ and two orderings $\sigma, \sigma' \in \mathfrak{S}_T$ such that the per-order modes $x_\sigma := \arg\max_x \log p_\sigma(x)$ and $x_{\sigma'}$ satisfy $H(x_\sigma, x_{\sigma'}) = d$. Sending $\varepsilon \to 0$ does* not *force $d \to 0$: scalar log-prob agreement is consistent with arbitrary Hamming disagreement of the modes.*

Intuitively, if two sequences that differ in every position tie under $\tilde{\mu}$, an order-dependent log-probability difference of only ε/T per position is enough for two orders to prefer opposite ones, so their argmaxes can disagree everywhere. The proof (Appendix A) is constructive. We build an explicit AO-ARM as a product of $d/2$ two-position "tilt gadgets", each of which contributes log-probability disagreement of $2\varepsilon/d$ but flips two positions of the per-order argmax. As $\varepsilon \to 0$, the per-gadget log-probability disagreement shrinks but the Hamming distance d between the two argmaxes is untouched. The proposition is therefore a worst-case witness that scalar log-probability agreement, the diagnostic most often reported as evidence of order invariance, does not imply argmax agreement: a model can be arbitrarily close to order-invariant in log-likelihood and still have per-order argmaxes that disagree on every position (take $d = T$ for even T).

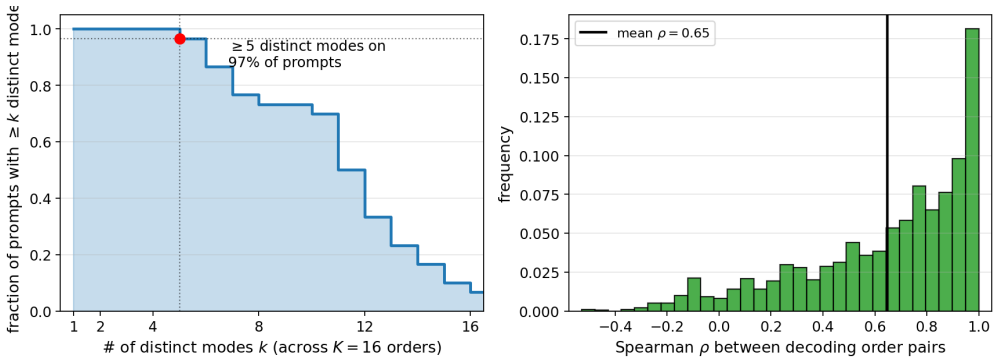


Figure 2: We compare the per-order argmaxes and rankings on a published Masked Diffusion Language Model checkpoint. **Left:** Complementary CDF of the number of distinct per-order argmaxes per prompt. On average 10.2 of the 16 per-order argmaxes of a prompt are distinct; all 16 differ on 2 of the 30 prompts. Note that most of these argmaxes are degenerate, with more than half repeating a single token (§3.3). **Right:** Distribution of pairwise Spearman correlations between rankings of a fixed candidate pool induced by random scoring orders. A perfectly order-invariant model would place all mass at one. The empirical distribution is broadly spread and well below one for the vast majority of order pairs.

3.3 Masked Diffusion Language Model on OpenWebText (MDLM-OWT): per-order rankings can disagree

To check that the phenomenon is not a small-model artifact, we run the same diagnostic on the publicly released 170M-parameter MDM of Sahoo et al. [2] trained on OpenWebText (MDLM-OWT), using its released weights without further training. The model does not take the time step as input, and we always decode and score one position at a time in an explicit order, so it behaves exactly as an AO-ARM (§3.1). Since we cannot compute the data mode in closed form, we instead study the disagreement of per-order argmaxes and rankings on fixed candidate pools for a set of prompts.

As prompts we take the first 8 tokens of 30 random WikiText-103 validation passages, and the model fills the next 56 positions. For each prompt, we run a beam search for each of 16 decoding orders to approximate that order's argmax. On average 10.2 of the 16 per-order argmaxes of a prompt are distinct, and all 16 differ on 2 of the 30 prompts (Figure 2). Most of these argmaxes are degenerate: 98.1% contain at most 11 distinct tokens among their 56 filled positions, and 54.0% are the prompt followed by a single token repeated 56 times.

Then, we ask whether the ranking of these sequences is consistent across orders. We score each pool under 12 new random orders, rank the candidates by log-probability under each order, and compare the rankings of every pair of orders by their Spearman correlation ρ; a perfectly order-invariant model would give $\rho = 1$ for every pair. Across the 1,980 pairs of scoring orders (66 per prompt), the mean ρ is 0.65 (Figure 2, right), suggesting that rankings also vary significantly across orders. Reproduction details are in Appendix E.

4 Order-marginalized scoring

The phenomenon of §3 motivates a scoring rule that is invariant to the choice of decoding order. The natural such rule is the order-marginalized log-likelihood $\tilde{\mu}(x) = \mathbb{E}_{\sigma \sim \mathrm{Unif}(\mathfrak{S}_T)}[\log p_\sigma(x)]$, whose negative expectation is the any-order training objective (§3.1). We propose two algorithms that target $\tilde{\mu}$ at decoding time. The first, OMS-rerank (Algorithm 1), reranks a generated candidate pool. The second, OMS-bon (Algorithm 2), generates the candidate pool itself by combining an adaptive position selector with stochastic token sampling and then applies OMS-rerank.

Algorithm 1: OMS-rerank. Given a finite pool C of candidate sequences, we draw M orders i.i.d. from $\mathrm{Unif}(\mathfrak{S}_T)$, score every candidate under the same M orders, and return the ranking by $\hat{\tilde{\mu}}_M$.

Algorithm 1 OMS-rerank: order-marginalized log-likelihood scoring.

Require: model p_θ, candidate pool C, number of MC samples M

Sample orders $\sigma_1, \dots, \sigma_M \overset{\text{iid}}{\sim} \mathrm{Unif}(\mathfrak{S}_T)$ ▷ shared by all candidates
for $c \in C$ **do**
  $\hat{\tilde{\mu}}(c) \leftarrow \frac{1}{M} \sum_{m=1}^{M} \log p_{\sigma_m}(c)$
end for
return $\{\hat{\tilde{\mu}}(c)\}$

Algorithm 2: OMS-bon. OMS-bon pairs the top-probability adaptive position selector of Kim et al. [4] with stochastic token sampling. At each step we choose the masked position with the largest single-token marginal probability, and sample a token at that position from the temperature-τ softmax of its logits. Repeating this for N independent rollouts produces a candidate pool, which we then rerank with OMS-rerank. The decoder is therefore a best-of-N procedure with a top-prob proposal and a $\tilde{\mu}$-reranker.

Algorithm 2 OMS-bon: stochastic best-of-N decoding.

Require: model p_θ, prompt x_P, particles N, token temperature τ, MC samples M

$C \leftarrow \emptyset$ ▷ set of distinct completions
for $n = 1, \dots, N$ **do**
  Initialize x from prompt x_P with all non-prompt positions masked
  while x has any masked position **do**
    Compute logits at every masked position from $p_\theta(\cdot \mid x)$
    $j^\star \leftarrow \arg\max_j \max_v \log p_\theta(x_j{=}v \mid x)$ over masked j ▷ top-prob
    Sample $v^\star \sim \mathrm{Categorical}(\mathrm{softmax}(\log p_\theta(\cdot \mid x)_{j^\star}/\tau))$
    Set $x_{j^\star} \leftarrow v^\star$
  end while
  $C \leftarrow C \cup \{x\}$ ▷ set union; identical completions merged
end for
return $\arg\max_{c \in C} \hat{\tilde{\mu}}_M(c)$ ▷ Algorithm 1

Beyond matching the training objective of models trained on the any-order (continuous-time NELBO) objective, the order-marginalized log-likelihood $\tilde{\mu}$ has a clean characterization: up to an additive constant, it is the log-density of the distribution closest to the per-order family in expected reverse KL, $\mathbb{E}_\sigma D_{\mathrm{KL}}(q \| p_\sigma)$.

Proposition 2 (KL-barycenter)**.** *Let $\Delta(\mathcal{X})$ be the simplex of probability distributions over $\mathcal{X}{=}\mathcal{V}^T$ and let $q \in \Delta(\mathcal{X})$. Define $\bar{p}(x) := Z^{-1} \exp \tilde{\mu}(x)$ with $Z = \sum_{x \in \mathcal{X}} \exp \tilde{\mu}(x) < \infty$. Then $\bar{p}$ is the unique minimizer of the order-averaged reverse KL,*

$$\bar{p} \;=\; \arg \min_{q \in \Delta(\mathcal{X})} \; \mathbb{E}_\sigma \big[D_{\mathrm{KL}}(q \| p_\sigma) \big]. \tag{4}$$

Consequently $\arg\max_x \tilde{\mu}(x) = \arg\max_x \log \bar{p}(x)$ *is the mode of the logarithmic opinion pool of* $\{p_\sigma\}_{\sigma \in \mathfrak{S}_T}$.

Proposition 2 (proof in Appendix A.2) says that the distribution closest to the per-order family $\{p_\sigma\}$ in expected reverse KL is $\bar{p} \propto \exp \tilde{\mu}$. Its mode and the per-candidate ranking induced by $\tilde{\mu}$ are therefore natural targets for decoding and scoring, respectively, that do not depend on the choice of any single σ. When there is no reason to prefer a particular order over another, $\tilde{\mu}$ is the natural order-free scoring target.

MC sample complexity. The estimator $\hat{\tilde{\mu}}_M$ averages the per-order log-likelihood over M random orders. Recall from (3) that $\varepsilon(x) = \sup_\sigma |\log p_\sigma(x) - \tilde{\mu}(x)|$ is the per-example deviation, and write $\varepsilon_C := \max_{c \in C} \varepsilon(c) \le \varepsilon$ for a pool C. Hoeffding's inequality applied to log-probabilities in an interval of width $2\varepsilon(x)$ gives a sample-complexity bound for ranking a finite candidate pool with high probability.

Proposition 3 (MC sample complexity). *For $\sigma_1, \dots, \sigma_M \overset{\text{iid}}{\sim} \mathrm{Unif}(\mathfrak{S}_T)$, the estimator $\hat{\tilde{\mu}}_M(x) := \frac{1}{M}\sum_{m=1}^{M} \log p_{\sigma_m}(x)$ satisfies, for all $t > 0$,*

$$\Pr\big(|\hat{\tilde{\mu}}_M(x) - \tilde{\mu}(x)| \geq t\big) \;\leq\; 2\exp\!\left(-\frac{Mt^2}{2\varepsilon(x)^2}\right), \tag{5}$$

i.e. $|\hat{\tilde{\mu}}_M(x) - \tilde{\mu}(x)| \leq \varepsilon(x)\sqrt{2\log(2/\delta)/M}$ with probability $\geq 1-\delta$ (and $\hat{\tilde{\mu}}_M(x) = \tilde{\mu}(x)$ if $\varepsilon(x) = 0$). Suppose a pool C is scored with the same orders for all candidates or with independent orders per candidate. If $\tilde{\mu}$ has a unique maximizer on C with margin $\Delta_{\mathrm{top}} > 0$ over the runner-up, then $M \geq \frac{8\varepsilon_C^2}{\Delta_{\mathrm{top}}^2}\log(2|C|/\delta)$ orders suffice for $\arg\max_{c\in C} \hat{\tilde{\mu}}_M(c)$ to be that maximizer with probability $\geq 1-\delta$. If the $\tilde{\mu}(c)$ are distinct with closest-pair gap $\Delta > 0$, the same bound with Δ in place of Δ_{top} suffices to recover the full ranking.

The proof is in Appendix A.3. The required number of orders grows only logarithmically in the pool size $|C|$ but quadratically in $\varepsilon_C/\Delta_{\mathrm{top}}$. The bound is a worst-case sufficient condition and is not tight on the pools we study: far fewer orders suffice in practice, with $M{=}4$ matching $M{=}16$ on Sudoku (Appendix C) and $M{=}16$ bringing the PCFG hit rate within 3% of the oracle (Table 1).

5 Experiments

We evaluate OMS in four settings: data-mode recovery on the closed-form PCFG of §3.2 (§5.1), Sudoku completion (§5.2), zero-shot protein-stability prediction with ProteinMPNN on the Megascale benchmark (§5.3), and text infilling with MDLM-OWT (§5.4). Reproduction details are in Appendix E, and the Sudoku ablations and hyperparameter sweeps are in Appendix C.

5.1 PCFG: data-mode recovery

We use the same PCFG, MDM, and prompts as in §3.2. We ask whether OMS-rerank helps identify the true data mode from a candidate pool of samples for each prompt. The candidate pool is the set of per-order beam-search modes from the diagnostic of §3.2; the closed-form data mode is contained in the pool for 67 of the 128 prompts, yielding an oracle upper bound of 0.523 for the hit rate of the true data mode. We take the top-ranked candidate under OMS-rerank as the prediction of the true data mode, and compare its hit rate with the average hit rate of the candidate pool (the per-order argmax under a random decoding order).

Table 1: PCFG data-mode hit rate. The per-order argmax baseline is reported with a bootstrap standard error over the $K{=}16$ decoding orders (10,000 resamples), reflecting uncertainty due to the choice of orders. OMS-rerank rows are reported as mean $\pm$ std across 5 replicates with different random scoring-order seeds. The oracle upper bound is the fraction of prompts whose pool contains the closed-form data mode.

Method	data-mode hit rate
Per-order argmax, averaged over 16 orders	0.360 ± 0.007
OMS-rerank, $M{=}2$	0.470 ± 0.023
OMS-rerank, $M{=}4$	0.483 ± 0.015
OMS-rerank, $M{=}8$	0.498 ± 0.017
OMS-rerank, $M{=}16$	$\mathbf{0.509 \pm 0.007}$
Oracle upper bound	0.523

Reranking the pool by model log-probability closes most of the recovery gap. OMS-rerank improves the data-mode hit rate over the per-order baseline by a substantial margin and closes most of the gap to the oracle upper bound that selects the data mode whenever it is in the pool (Table 1). The improvement is well outside the cross-replicate noise. At $M{=}16$, the hit rate is within 3% of the oracle.

5.2 Sudoku completion

Next, we evaluate OMS on Sudoku completion. We flatten 9×9 Sudoku grids to sequences of length $T{=}81$ over the $|\mathcal{V}|{=}9$ digits and train an 800K-parameter MDM on solved grids. We evaluate on unique-solution puzzles obtained by greedy hint-reduction at hint counts $h \in \{25, 28, 31\}$, with $n{=}150$ puzzles per hint level.

The evaluation metric is exact-match accuracy: whether the decoded sequence equals the ground-truth solution. Since each puzzle is an independent Bernoulli trial under the evaluation set, we report the puzzle-solving fraction p with binomial standard error $\sqrt{p(1-p)/n}$ over the $n{=}150$ puzzles per hint level. We compare three deterministic single-order decoders. *L2R* fills positions $0, 1, \ldots, T{-}1$ in order with beam search over 16 beams. *Top-prob* is the adaptive position selector of Kim et al. [4], which at each step unmasks the position whose highest single-token marginal probability under the current partial reveal is largest, and emits that argmax token. *Top-prob-margin* is the refinement introduced by Kim et al. [4], which instead picks the position with the largest gap between its top-1 and top-2 token probabilities. These three baselines are deterministic given the prompt and the model.

As OMS-bon uses more forward passes than the baselines, we introduce two additional baselines that use at least as many forward passes as OMS-bon at $N{=}512$: *single-order best-of-N* runs Algorithm 2 with $N{=}1024$ and $M{=}1$, so that one random order scores the pool, and *top-prob beam search* keeps the $B{=}2560$ partial sequences with the highest cumulative log-probability, each beam revealing its own most confident position. We run our OMS-bon decoder at $N \in \{128, 512\}$ with token temperature $\tau = 0.7$ and $M = 4$.

Table 2: Sudoku exact-match accuracy at hint counts h: the fraction of $n{=}150$ puzzles solved exactly, $\pm$ binomial standard error $\sqrt{p(1-p)/n}$. The last column is the number of forward passes per puzzle at $h = 25/28/31$, counted at the model call (one length-81 sequence is one pass).

Method	$h{=}25$	$h{=}28$	$h{=}31$	forward passes
L2R, beam 16	$0.040{\pm}0.016$	$0.033{\pm}0.015$	$0.073{\pm}0.021$	876 / 828 / 780
top-prob [4]	$0.147{\pm}0.029$	$0.340{\pm}0.039$	$0.600{\pm}0.040$	56 / 53 / 50
top-prob-margin [4]	$0.187{\pm}0.032$	$0.313{\pm}0.038$	$0.633{\pm}0.039$	56 / 53 / 50
single-order best-of-N, $N{=}1024$	$0.453{\pm}0.041$	$0.673{\pm}0.038$	$0.880{\pm}0.027$	71.6k / 61.4k / 54.2k
top-prob beam search, $B{=}2560$	$0.187{\pm}0.032$	$0.367{\pm}0.039$	$0.600{\pm}0.040$	134k / 126k / 119k
OMS-bon, $N{=}128$ *(ours)*	$0.533{\pm}0.041$	$0.767{\pm}0.035$	$0.920{\pm}0.022$	21.4k / 14.9k / 10.1k
OMS-bon, $N{=}512$ *(ours)*	$\mathbf{0.647{\pm}0.039}$	$\mathbf{0.853{\pm}0.029}$	$\mathbf{0.940{\pm}0.019}$	64.7k / 45.7k / 33.6k

OMS-bon outperforms single-order baselines at every hint level. At all three hint counts, OMS-bon with $N{=}512$ outperforms every baseline in Table 2, including the two that use more forward passes; its smallest margin is $+0.060$, over single-order best-of-N at $h{=}31$ (paired exact McNemar $p \leq 0.049$ at every hint count; Table 5). Averaging over orders matters: scoring the same candidate pools with $M{=}4$ orders instead of one raises accuracy at $h{=}28$ from 0.65 to 0.77 at $N{=}128$ and from 0.68 to 0.88 at $N{=}1024$. Accuracy improves with the number of rollouts N across the range we test, from 0.65 at $N{=}32$ to 0.88 at $N{=}1024$ ($h{=}28$), with diminishing returns at larger N. The full scaling curve and the component ablations of Algorithm 2 are in Appendix C.

Sudoku at ≤ 31 hints has many locally valid partial completions that are globally inconsistent with the unique solution. We hypothesize that a deterministic single-order top-prob decoder commits to one such completion, and at typical hint counts has no mechanism to escape it. Stochastic top-prob at $\tau = 0.7$ instead produces a diverse pool of candidates that span multiple such completions, and the $\tilde{\mu}$-reranker selects the one to which the order-marginalized model assigns the highest joint probability. OMS-bon thus approximates the mode of $\bar{p} \propto \exp \tilde{\mu}$ over the sampled pool with a tractable best-of-N decoder.

Compute cost. OMS-bon trades compute for accuracy: top-prob costs $T - h$ forward passes per puzzle (one per unfilled position), whereas OMS-bon costs $N(T - h)$ passes for the N rollouts plus UMT for scoring the $U \leq N$ distinct candidates under M orders. At $N{=}512$ and $M{=}4$ this is $1{,}155\times$, $862\times$ and $672\times$ the forward passes of top-prob at $h = 25$, 28 and 31 (Table 6).

5.3 Megascale: zero-shot protein-stability prediction

Prediction of folding stability is important for optimizing protein stabilities in drug discovery [22]. ProteinMPNN [3] is a 1.7M-parameter AO-ARM trained to predict amino-acid identity at every position of a protein conditioned on fixed backbone coordinates. A standard recipe in protein engineering uses $\log p(\text{sequence} \mid \text{backbone})$ as a zero-shot proxy for folding stability [11, 12], following the intuition that sequences that have higher probabilities of fitting a backbone structure are likely more thermostable. The Megascale dataset [22] reports experimental folding stabilities $\Delta\Delta G$ for single-amino-acid *mutants* of reference sequences (*wild types*, henceforth WTs; a mutant is a sequence that differs from its WT in exactly one residue). We use 337 WTs and their 306,501 single mutants (Appendix E).

A common scoring rule, the default of ProteinMPNN's scoring script, draws a fresh random order σ for every sequence [3]. We instead score all mutants of a wild type under the same M random orders and average (OMS-rerank with shared orders, Algorithm 1, as in StaB-ddG [12]), which costs M forward passes per mutant. The performance metric is the *per-wild-type* Spearman correlation ρ, computed within a single WT family by ranking that WT's mutants by their model log-probability score and by their experimental $\Delta\Delta G$, and taking the Spearman correlation of those two rankings. We report the mean of ρ across the 337 WTs with a bootstrap standard error from 10,000 WT-level resamples, and the paired per-WT difference $\Delta\rho$ from a single random order shared by a WT's mutants (averaged over 32 such orders, with a 95% bootstrap CI), with $n_\uparrow$ the number of WTs where $\Delta\rho > 0$ (Table 3). We also report enrichment@$k = (\bar{y}_k - \bar{y})/(\bar{y}^\star_k - \bar{y})$, averaged over WTs, where $\bar{y}$ is the mean $\Delta\Delta G$ over a WT's mutants and $\bar{y}_k$, $\bar{y}^\star_k$ that of its top-k mutants by score and by $\Delta\Delta G$; a random pick scores 0 in expectation and a perfect ranking 1. A positive ρ indicates that higher log-probability is associated with greater folding stability.

Table 3: Zero-shot protein-stability prediction on Megascale (337 WTs). Cost is ProteinMPNN forward passes per mutant. Mean ρ is $\pm$ bootstrap standard error over WTs. $\Delta\rho$ is the paired per-WT difference from a fixed shared random order, and $n_\uparrow$ the number of WTs with $\Delta\rho > 0$. Enrichment@k is 0 in expectation for a random pick and 1 for the k most stable mutants.

Method	Cost	mean ρ	enrichment@5/10/20	$\Delta\rho$ [95% CI]	$n_\uparrow$
Random σ per mutant	1	$+0.463 \pm 0.008$	.417 / .430 / .445	-0.0706 [-0.0744, -0.0667]	4
Fixed L2R	1	$+0.532 \pm 0.009$	.447 / .458 / .476	-0.0017 [-0.0036, $+0.0002$]	167
Fixed shared random order	1	$+0.534 \pm 0.009$	.451 / .463 / .476	0	—
Shared OMS, M=4	4	$+0.542 \pm 0.009$	.457 / .468 / .483	$+0.0083$ [$+0.0074$, $+0.0092$]	278
Shared OMS, M=32	32	$\mathbf{+0.545 \pm 0.009}$	.463 / .474 / .485	$\mathbf{+0.0111}$ [$+0.0106$, $+0.0116$]	335

Sharing the scoring orders across a wild type's mutants is what helps most (Table 3). A single random order shared by all mutants of a WT improves ρ over a fresh order per mutant by $+0.0706$ [$+0.0667$, $+0.0744$] (better on 333 of 337 WTs). Averaging over shared orders adds a smaller but consistent gain, $+0.0083$ [$+0.0074$, $+0.0092$] at M=4 and $+0.0111$ [$+0.0106$, $+0.0116$] at M=32 (better on 335 of 337 WTs), while the fixed L2R order is no better than a random shared one (-0.0017 [-0.0036, $+0.0002$]).

5.4 Text infilling with MDLM-OWT

We test OMS-rerank on a pool of model samples: infilling a 4-token span in 300 WikiText-103 validation passages of 52 tokens, where the span follows the first 24 tokens, with MDLM-OWT. For each passage the pool is 64 samples of the span, each drawn under its own uniformly random span order at temperature 1, and deduplicated (53.0 distinct candidates on average). We compare ways of choosing the span (Table 4). Single-order reranking picks the candidate with the highest log-probability of its span given the context, computed under one random order of the span positions; OMS-rerank averages this score over M orders. As references, a uniform pick takes a random candidate from the pool, and the pool oracle picks the candidate closest to the true span, an upper bound for any reranker.

Reranking by model log-probability helps: single-order reranking improves F1 over a uniform pick by $+0.0936$ [$+0.0773$, $+0.1106$] (paired 95% bootstrap CI over passages). Order marginalization does not help further: OMS-rerank with M=4 differs from single-order reranking by $+0.0028$ [-0.0047, $+0.0105$] F1 (Table 4).

Table 4: Span infilling with MDLM-OWT on 300 WikiText-103 validation passages. Cost is sequence forward passes per passage. F1 is $\pm$ standard error over passages. The OMS-rerank rows average over random M-subsets of the S=16 scoring orders; single-order rerank averages over the S orders used alone.

Arm	cost	exact match	token acc.	F1
uniform pick from the pool	256	0.005	0.164	0.204 ± 0.009
single-order rerank	468	0.054	0.258	0.297 ± 0.015
OMS-rerank, M=4	1104	0.054	0.264	0.300 ± 0.015
OMS-rerank, M=16	3649	0.057	0.260	0.295 ± 0.016
pool oracle	256	0.150	0.530	0.575 ± 0.015

6 Discussion

In this paper we identified a phenomenon in any-order autoregressive models and masked diffusion models: scoring a candidate sequence under a single decoding order is a noisy proxy for the model's actual belief about the candidate. Per-order argmaxes and per-order rankings of fixed candidate pools disagree on most inputs. We characterized this disagreement both empirically and theoretically. Building on this observation, we proposed OMS, a simple Monte Carlo estimator of the order-marginalized log-likelihood $\tilde{\mu}(x) = \mathbb{E}_\sigma[\log p_\sigma(x)]$. Used as a reranker, OMS closes a substantial fraction of the per-order baseline's gap to the oracle on a closed-form PCFG, and improves zero-shot protein-stability prediction on Megascale by $+0.011$ in Spearman correlation over a single shared order. Used inside a stochastic best-of-N decoder, it raises Sudoku exact-match accuracy by 6 to 19 percentage points over single-order best-of-N, which uses at least as many forward passes.

This work has several limitations and natural directions for future work. The empirical wins reported here concentrate on tasks where no decoding order is structurally privileged, including the closed-form PCFG, Sudoku completion, and protein-stability ranking. The gain from OMS depends on the disagreement ε between per-order log-likelihoods and on whether the candidate pool contains the $\bar{p}$-mode. On text infilling with MDLM-OWT (§5.4), OMS-rerank matches but does not improve on reranking under a single random order; whether order marginalization helps on code and other text tasks remains to be validated. We also restrict our experiments to the position-by-position decoding regime; an empirical study of the block-parallel extension in Appendix B is future work, as are choosing the number of orders M adaptively and comparing OMS with the post-hoc scorers of §2.

More broadly, our results suggest that order marginalization is a natural alternative for per-order log-probability scoring. We hope the analysis developed here serves both as a recipe for building order-invariant scoring rules and as a starting point for future work that connects per-order disagreement to downstream performance gaps in AO-ARMs more generally.

Acknowledgments and Disclosure of Funding

We thank Nathaniel Diamant for helpful discussions. Arthur Deng is supported by the Stanford HAI Data Science Fellowship and the Stanford Artificial Intelligence Laboratory (SAIL) Stanley Fellowship.

References

[1] Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022.

[2] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024.

[3] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning–based protein sequence design using ProteinMPNN. *Science*, 378(6615), 2022.

[4] Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *International Conference on Machine Learning*, 2025.

[5] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[6] Victor Besnier, Mickael Chen, David Hurych, Eduardo Valle, and Matthieu Cord. Halton scheduler for masked generative image transformer. In *International Conference on Learning Representations*, 2025.

[7] Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models. In *International Conference on Machine Learning*, 2026.

[8] Chunsan Hong, Seonho An, Min-Soo Kim, and Jong Chul Ye. Improving discrete diffusion unmasking policies beyond explicit reference policies. In *International Conference on Learning Representations*, 2026.

[9] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

[10] Junhao Xiong, Ishan Gaur, Maria Lukarska, Hunter Nisonoff, Luke M. Oltrogge, David F. Savage, and Jennifer Listgarten. Property guidance for protein sequence generative models with ProteinGuide. *Nature Biotechnology*, 2026. doi: 10.1038/s41587-026-03207-z.

[11] Henry Dieckhaus, Michael Brocidiacono, Nicholas Z. Randolph, and Brian Kuhlman. Transfer learning to leverage larger datasets for improved prediction of protein stability changes. *Proceedings of the National Academy of Sciences*, 121(6), 2024.

[12] Arthur Deng, Karsten Householder, Fang Wu, K. Christopher Garcia, and Brian Trippe. Predicting mutational effects on protein binding from folding energy. In *International Conference on Machine Learning*, 2025.

[13] Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *International Conference on Machine Learning*, 2014.

[14] Zachary Horvitz, Raghav Singhal, Hao Zou, Carles Domingo-Enrich, Zhou Yu, Rajesh Ranganath, and Kathleen McKeown. Rethinking reasoning with MDLMs: Early exits, post-hoc reasoning, and beyond. In *Conference on Language Modeling*, 2026.

[15] Ziyu Chen, Xinbei Jiang, Peng Sun, and Tao Lin. Quantifying and optimizing path uncertainty in masked diffusion models. In *Advances in Neural Information Processing Systems*, 2026.

[16] Yangyi Shen, Tianjian Feng, Jiaqi Han, Wen Wang, Tianlang Chen, Chunhua Shen, Jure Leskovec, and Stefano Ermon. Improving diffusion language model decoding through joint search in generation order and token space. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026.

[17] Sanghyun Lee, Seungryong Kim, Jongho Park, and Dongmin Park. Lookahead unmasking elicits accurate decoding in diffusion language models. In *International Conference on Machine Learning*, 2026.

[18] Kaisen Yang, Jayden Teoh, Kaicheng Yang, Yitong Zhang, and Alex Lamb. Improving sampling for masked diffusion models via information gain. In *International Conference on Machine Learning*, 2026.

[19] Mingyu Cao, Alvaro H. C. Correia, Christos Louizos, Shiwei Liu, and Lu Yin. Search or accelerate: Confidence-switched position beam search for diffusion language models. In *International Conference on Machine Learning*, 2026.

[20] Yue Wu and Jian Huang. Stability-weighted decoding for diffusion language models. *arXiv preprint arXiv:2604.17068*, 2026.

[21] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, 2025.

[22] Kotaro Tsuboyama, Justas Dauparas, Jonathan Chen, Elodie Laine, Yasser Mohseni Behbahani, Jonathan J. Weinstein, Niall M. Mangan, Sergey Ovchinnikov, and Gabriel J. Rocklin. Mega-scale experimental analysis of protein folding stability in biology and design. *Nature*, 620(7973), 2023.

A Proofs

Throughout this appendix $\sigma \sim \mathrm{Unif}(\mathfrak{S}_T)$, $\mathcal{V}$ is the output vocabulary (`[MASK]` excluded), and all conditionals are strictly positive on $\mathcal{V}$. For a finite pool $C \subset \mathcal{X}$ we write $\varepsilon_C := \max_{c \in C} \varepsilon(c) \le \varepsilon$ for the disagreement restricted to the pool. Appendix B sketches an extension to parallel decoding.

A.1 Proof of Proposition 1 (argmax pathology)

Construction. Fix $\varepsilon > 0$ and an even d with $2 \le d \le T$, and set $\alpha := 4\varepsilon/d$. Fix two distinct tokens $0, 1 \in \mathcal{V}$. Choose $d/2$ disjoint pairs of positions (i_g, j_g) with $i_g < j_g$, $g = 1, \dots, d/2$ (the *tilt gadgets*); the remaining $T - d$ positions are *constant*. For a gadget position $k \in \{i_g, j_g\}$ with partner $\bar{k}$, every revealed set $S \not\ni k$, every x_S and every $b \in \mathcal{V}$, let

$$p_\theta(x_k{=}b \mid x_S) = Z^{-1} \exp\big(\alpha\, \mathbf{1}[b = \tau_k(x_S)]\big), \qquad Z := e^\alpha + |\mathcal{V}| - 1,$$
$$\tau_k(x_S) = x_{\bar{k}} \text{ if } \bar{k} \in S; \qquad \tau_{i_g}(x_S) = 0 \text{ if } j_g \notin S; \qquad \tau_{j_g}(x_S) = 1 \text{ if } i_g \notin S.$$

Every constant position c has the context-free conditional $p_\theta(x_c{=}b \mid x_S) = q(b) \propto \exp(\mathbf{1}[b = 0])$. All conditionals are strictly positive, and all gadget conditionals share the normalizer Z.

Factorization. A gadget conditional depends on x_S only through the partner, so for every σ and x

$$\log p_\sigma(x) = \sum_{g=1}^{d/2} c_g^\sigma(x_{i_g}, x_{j_g}) + \sum_c \log q(x_c), \qquad c_g^\sigma = \begin{cases} \mathsf{L}, & \sigma \text{ reveals } i_g \text{ before } j_g, \\ \mathsf{R}, & \text{otherwise,} \end{cases}$$

where $\mathsf{L}(a, b) := \alpha(\mathbf{1}[a = 0] + \mathbf{1}[b = a]) - 2\log Z$ and $\mathsf{R}(a, b) := \alpha(\mathbf{1}[b = 1] + \mathbf{1}[a = b]) - 2\log Z$.

Modes. Let σ reveal i_g before j_g for every g (e.g. L→R), and let σ' reveal j_g before i_g for every g (e.g. R→L). Each of $\log p_\sigma$ and $\log p_{\sigma'}$ is a sum of functions of disjoint coordinate blocks, so its joint maximizer over $\mathcal{V}^T$ is found block by block. We have $\mathsf{L}(a, b) \le 2\alpha - 2\log Z$ with equality iff $a = b = 0$, $\mathsf{R}(a, b) \le 2\alpha - 2\log Z$ with equality iff $a = b = 1$, and q is uniquely maximized at 0. Hence x_σ is unique and equals 0 everywhere, while $x_{\sigma'}$ is unique and equals 1 on the gadgets and 0 on the constant positions; so $H(x_\sigma, x_{\sigma'}) = d$.

Disagreement. Under $\sigma \sim \mathrm{Unif}(\mathfrak{S}_T)$ each i_g precedes j_g with probability $1/2$, and the constant terms do not depend on σ. Hence $\tilde{\mu}(x) = \sum_g \frac{1}{2}(\mathsf{L} + \mathsf{R})(x_{i_g}, x_{j_g}) + \sum_c \log q(x_c)$ and

$$\log p_\sigma(x) - \tilde{\mu}(x) = \sum_{g=1}^{d/2} \pm\tfrac{1}{2}(\mathsf{L} - \mathsf{R})(x_{i_g}, x_{j_g}),$$

with sign $+$ if σ reveals i_g first and $-$ otherwise, and $(\mathsf{L} - \mathsf{R})(a, b) = \alpha\big(\mathbf{1}[a = 0] - \mathbf{1}[b = 1]\big) \in \{-\alpha, 0, \alpha\}$. Thus $|\log p_\sigma(x) - \tilde{\mu}(x)| \le \frac{d}{2} \cdot \frac{\alpha}{2} = \varepsilon$ for all x and σ, with equality at $x = 0^T$ under σ; each gadget contributes at most $\alpha/2 = 2\varepsilon/d$.

Limit. The modes depend on α only through the sign $\alpha > 0$, so letting $\varepsilon \to 0$ leaves x_σ, $x_{\sigma'}$ and $H(x_\sigma, x_{\sigma'}) = d$ unchanged. □

A.2 Proof of Proposition 2 (KL-barycenter)

The divergence in (4) is $D_{\mathrm{KL}}(q \| p_\sigma)$, with the free distribution q in the first argument; this is the *reverse* (exclusive) KL. For every $q \in \Delta(\mathcal{X})$,

$$\begin{aligned} \mathbb{E}_\sigma[D_{\mathrm{KL}}(q \| p_\sigma)] &= \sum_x q(x) \log q(x) - \sum_x q(x)\, \mathbb{E}_\sigma[\log p_\sigma(x)] \\ &= \sum_x q(x)\big(\log q(x) - \tilde{\mu}(x)\big) = D_{\mathrm{KL}}(q \| \bar{p}) - \log Z, \end{aligned}$$

using $\tilde{\mu} = \log \bar{p} + \log Z$. Since $D_{\mathrm{KL}}(q \| \bar{p}) \ge 0$ with equality iff $q = \bar{p}$, the minimizer is unique and equals $\bar{p}$ ($\bar{p} > 0$ because all conditionals are positive). The mode statement holds because $\log \bar{p}$ and $\tilde{\mu}$ differ by a constant. □

A.3 Proof of Proposition 3 (MC sample complexity)

Concentration. The variables $Y_m := \log p_{\sigma_m}(x)$ are i.i.d. with mean $\tilde{\mu}(x)$ and values in an interval of width $2\varepsilon(x)$. Hoeffding's inequality gives $\Pr(|\hat{\tilde{\mu}}_M(x) - \tilde{\mu}(x)| \geq t) \leq 2\exp(-2Mt^2/(2\varepsilon(x))^2) = 2\exp(-Mt^2/(2\varepsilon(x)^2))$, and setting the bound equal to δ gives the interval. If $\varepsilon(x) = 0$, every Y_m equals $\tilde{\mu}(x)$.

Top candidate. Let $c^\star$ be the maximizer of $\tilde{\mu}$ on C, and let $E := \{|\hat{\tilde{\mu}}_M(c) - \tilde{\mu}(c)| < \Delta_{\text{top}}/2 \text{ for all } c \in C\}$. On E, for every $c \neq c^\star$, $\hat{\tilde{\mu}}_M(c^\star) > \tilde{\mu}(c^\star) - \Delta_{\text{top}}/2 \geq \tilde{\mu}(c) + \Delta_{\text{top}}/2 > \hat{\tilde{\mu}}_M(c)$. Applying the concentration bound to each candidate and taking a union bound, which needs no independence across candidates and therefore holds for shared orders, independent orders, or any other coupling, gives $\Pr(E^c) \leq \sum_{c\in C} 2\exp(-M\Delta_{\text{top}}^2/(8\varepsilon(c)^2)) \leq 2|C|\exp(-M\Delta_{\text{top}}^2/(8\varepsilon_C^2)) \leq \delta$ under the stated M (if $|C| = 1$ or $\varepsilon_C = 0$ the claim is trivial).

Full ranking. Repeat the argument with Δ in place of Δ_{top}: on $\{|\hat{\tilde{\mu}}_M(c) - \tilde{\mu}(c)| < \Delta/2 \text{ for all } c\}$ every pair keeps its strict order. □

B Extension to parallel decoding

This appendix outlines an extension of the formulation of §3.1 to schedules that unmask several positions per step. These schedules are not used in our experiments, and complete proofs are left to future work. Throughout, the conditionals (1) are fixed and strictly positive.

Block schedules. A block schedule is an ordered partition $\pi = (B_1, \ldots, B_K)$ of $[T]$ into nonempty blocks. It reveals B_k at step k, sampling the positions of a block independently given the earlier blocks:

$$\log p_\pi(x) = \sum_{k=1}^{K} \sum_{j\in B_k} \log p_\theta\big(x_j \mid x_{B_1\cup\cdots\cup B_{k-1}}\big).$$

A permutation is the schedule with T singleton blocks. For a law ν over schedules, let $\tilde{\mu}_\nu(x) := \mathbb{E}_{\pi\sim\nu}[\log p_\pi(x)]$ and $\varepsilon_\nu := \max_x \max_{\pi\in\operatorname{supp}\nu} |\log p_\pi(x) - \tilde{\mu}_\nu(x)|$; for $\nu = \text{Unif}(\mathfrak{S}_T)$ these are $\tilde{\mu}$ and ε. For a denoiser without time input, the R-step MDLM sampler [2] reveals each position at step k independently with a probability determined by the noise schedule; it therefore follows a random schedule with at most R blocks, drawn from a law $\nu_{\alpha,R}$ whose mass depends only on the block sizes.

Propositions 2 and 3 for block schedules. The proofs of Propositions 2 and 3 require only that the per-order scores be bounded and that each p_σ be a normalized distribution, and both properties hold for block schedules. Consequently, for any law ν, $\bar{p}_\nu \propto \exp\tilde{\mu}_\nu$ is the unique minimizer of $\mathbb{E}_{\pi\sim\nu} D_{\text{KL}}(q\|p_\pi)$, and the average of $\log p_\pi$ over M schedules drawn from ν estimates $\tilde{\mu}_\nu$ with the sample complexity of Proposition 3, with ε_ν in place of ε.

Total-correlation decomposition. Let $\text{TC}_\pi(x)$ denote the average of $\log p_\sigma(x)$ over the orders σ that reveal the blocks of π in sequence, minus $\log p_\pi(x)$; it vanishes for permutations. For laws whose mass depends only on the block sizes, such as $\nu_{\alpha,R}$,

$$\tilde{\mu}_\nu(x) = \tilde{\mu}(x) - \mathbb{E}_{\pi\sim\nu}\text{TC}_\pi(x).$$

In a well-specified model with joint P, $\text{TC}_\pi(x)$ is the pointwise total correlation of each block conditioned on the preceding blocks, and its expectation under P is nonnegative; block decoding therefore underestimates $\log P$ on average by the within-block dependence it neglects. The disagreement ε_ν is controlled by the spread of this term: $\varepsilon_\nu \leq \varepsilon + \tau_\nu$ for such laws, where τ_ν bounds $|\text{TC}_\pi - \mathbb{E}_\nu\text{TC}|$ over the support of ν, and $\varepsilon_\nu \leq 2\varepsilon + \tau_\nu$ for any law. Whether parallel decoding increases or decreases the disagreement of trained models remains an empirical question.

C Sudoku: compute scaling and ablations

Table 6 reports the wall-clock time per puzzle; the timed runs reproduce the per-puzzle outcomes of Table 2 exactly. The time ratio is far below the forward-pass ratio because OMS-bon batches its

Table 5: Paired exact two-sided McNemar tests for Table 2: OMS-bon (N=512, M=4) against each other row. Each cell is $+a/-b$, where a puzzles are solved only by OMS-bon and b only by the other row (n=150 per hint count), and the two-sided p-value.

Row	h=25	h=28	h=31
L2R, beam 16	$+92/-1, 1.9\times10^{-26}$	$+123/-0, 1.9\times10^{-37}$	$+132/-2, 8.3\times10^{-37}$
top-prob	$+75/-0, 5.3\times10^{-23}$	$+77/-0, 1.3\times10^{-23}$	$+51/-0, 8.9\times10^{-16}$
top-prob-margin	$+71/-2, 5.7\times10^{-19}$	$+81/-0, 8.3\times10^{-25}$	$+46/-0, 2.8\times10^{-14}$
single-order best-of-N, N=1024	$+33/-4, 1.1\times10^{-6}$	$+31/-4, 3.5\times10^{-6}$	$+13/-4, 0.049$
top-prob beam search, B=2560	$+69/-0, 3.4\times10^{-21}$	$+73/-0, 2.1\times10^{-22}$	$+51/-0, 8.9\times10^{-16}$
OMS-bon, N=128	$+17/-0, 1.5\times10^{-5}$	$+13/-0, 2.4\times10^{-4}$	$+3/-0, 0.25$

N rollouts and its UMT scoring passes, whereas top-prob decodes a single sequence and leaves the GPU largely idle. The comparison therefore measures latency; when the GPU is saturated, for example by decoding many puzzles at once, the time ratio should approach the forward-pass ratio.

Table 6: Wall-clock seconds per Sudoku puzzle, mean over the n=150 puzzles per hint count, each decoded alone after an untimed warm-up on one NVIDIA RTX 5090. OMS-bon uses N=512 and M=4. The last two columns give its cost relative to top-prob in time and in forward passes (Table 2).

Hints	top-prob (s)	OMS-bon (s)	time ratio	forward-pass ratio
h=25	0.091	0.46	5.0×	1,155×
h=28	0.085	0.34	4.0×	862×
h=31	0.059	0.28	4.8×	672×

The following table reports the accuracy of OMS-bon as a function of the number of rollouts N at h=28 (n=100 puzzles), together with single-order scoring (M=1) of the same candidate pools.

N	32	64	128	256	512	1024
OMS-bon (M=4)	0.65	0.72	0.77	0.82	0.86	0.88
single-order scoring (M=1)	0.59	0.62	0.65	0.69	0.70	0.68

Table 7 ablates each component of Algorithm 2 at h=28 with N=128 on n=100 puzzles. Both the top-prob proposal and the $\tilde{\mu}$-reranker are essential, and accuracy is highest at moderate τ. Rescoring the same candidate pools with M=4 orders instead of one solves 12 additional puzzles and none fewer (exact McNemar $p = 4.9\times10^{-4}$); larger M does not change the accuracy.

Table 7: Sudoku ablations of OMS-bon at h=28 with N=128 on n=100 puzzles. Errors are binomial standard errors $\sqrt{p(1-p)/n}$. The default uses τ=0.7 and M=4; the M rows rescore the default's candidate pools.

Configuration	exact-match accuracy
OMS-bon, τ=0.7, M=4 *(default)*	0.77±0.042
with random-σ instead of top-prob (proposal ablation)	0.00±0.000
with $\arg\max$ cumulative log-probability instead of $\tilde{\mu}$-rerank	0.37±0.048
with τ=0.3 (token sampling)	0.43±0.050
with τ=0.5	0.68±0.047
with τ=1.0	0.80±0.040
with τ=1.5	0.43±0.050
with M=1 (single-order scoring)	0.65±0.048
with M=2	0.72±0.045
with M=8	0.77±0.042
with M=16	0.77±0.042
with M=32	0.77±0.042

D Per-order beam search

Algorithm 3 approximates the per-order argmax $\arg\max_x \log p_\sigma(x)$ used in §3.2 and §3.3 by beam search of width B along a fixed order σ.

Algorithm 3 Per-order beam search at order σ.

Require: model p_θ, prompt x_P, unmasking order $\sigma \in \mathfrak{S}_T$, beam width B
Initialize a single beam b_0: x_P at the prompt positions, `[MASK]` elsewhere; set $\log w(b_0) \leftarrow 0$
$\mathcal{B} \leftarrow \{b_0\}$
for $t = 0, 1, \ldots, T-1$ **do**
  $j \leftarrow \sigma(t)$
  if j is a prompt position **then continue**
  end if
  $\mathcal{C} \leftarrow \emptyset$
  for each beam $b \in \mathcal{B}$ **do**
    Compute logits $\ell_b \leftarrow \log p_\theta(\cdot \mid b)_j \in \mathbb{R}^{|\mathcal{V}|}$
    for each of the top-B tokens v under ℓ_b **do**
      $b' \leftarrow b$ with position j set to v; $\log w(b') \leftarrow \log w(b) + \ell_b(v)$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{b'\}$
    end for
  end for
  $\mathcal{B} \leftarrow$ top-B beams in $\mathcal{C}$ by $\log w$
end for
return $\arg\max_{b \in \mathcal{B}} \log w(b)$

Implementation. All B beams are evaluated in one batched model call per step (B sequence forward passes). The PCFG diagnostic uses $B = 48$, the doubled-beam sensitivity check below uses $B = 96$, and the MDLM-OWT diagnostic uses $B = 96$ (doubled to $B = 192$ in the sensitivity check).

Beam-width sensitivity. To verify that the disagreement in §3 is not caused by search error, we repeat both diagnostics with twice the beam width, keeping prompts, orders and seeds fixed.

PCFG diagnostic	beam=48 (paper)	beam=96 (doubled)
fraction of prompts with ≥ 2 distinct argmaxes	0.766	0.773
mean number of distinct argmaxes per prompt	5.78	5.73
fraction of prompts on which all 16 orders agree	0.234	0.227
mean number of orders that recover the data mode	5.77	5.88

MDLM-OWT diagnostic	beam=96 (paper)	beam=192 (doubled)
mean number of distinct argmaxes per prompt	10.17	9.70
fraction of prompts with ≥ 2 distinct argmaxes	1.00	1.00
mean pairwise Spearman ρ across scoring orders	0.65	0.57

In both settings, doubling the beam width leaves every statistic qualitatively unchanged: most per-order argmaxes remain distinct, the per-order rankings continue to disagree, and the distribution of data-mode recovery on the PCFG remains bimodal. When the beam-96 and beam-192 pools are scored together under the same orders, the mean pairwise ρ is 0.705, and only 4.1% of the PCFG beam-48 argmaxes change at beam 96, each to a sequence with a higher score under its order. The disagreement reported in §3 is therefore not an artifact of the beam width.

E Reproduction details

MDLM-OWT checkpoint. The 169.6M-parameter MDM of §3.3 is the official release of Sahoo et al. [2], hosted on HuggingFace at `kuleshov-group/mdlm-owt`. It is a 12-layer Diffusion

Transformer (DiT) trained on OpenWebText with the substitution (SUBS) parameterization of Sahoo et al. [2], vocabulary size $50{,}258$ (GPT-2 byte-pair encoding plus a `[MASK]` token), maximum sequence length 1024, and time-conditioning disabled. We evaluate the released weights without any fine-tuning. The prompts of §3.3 are drawn from WikiText-103 validation lines with more than 64 words, selected by a seeded shuffle; the sequence length is $T{=}64$.

Megascale data and ProteinMPNN. The dataset is the processed release of Tsuboyama et al. [22], with $\Delta\Delta G$ values and AlphaFold models of the wild types. We keep the WTs that have an AlphaFold model and at least 10 single substitutions with a $\Delta\Delta G$ value: 337 of the 479 WTs (181 natural, 146 designed, 10 variant backgrounds), with $306{,}501$ mutants. They include all 298 WTs of the Megascale splits of StaB-ddG [12], whose ProteinMPNN code we use for scoring. The weights are the published soluble-protein checkpoint `v_48_020` of Dauparas et al. [3], with backbone (N, C_α, C, O) featurization, $k{=}48$ neighbors, hidden dimension 128, and 3 encoder and 3 decoder layers ($\sim$1.7M parameters). We disable the inference-time backbone noise so that all stochasticity in the score is due to the choice of decoding order alone.

Scoring with ProteinMPNN. For a decoding order σ of the L residues, one ProteinMPNN forward pass gives $\log p_\sigma(\mathrm{seq} \mid X)$, Equation (2) with every conditional also given the backbone X, because its decoder masks every position not yet revealed by σ. The per-order baseline scores each mutant under its own random order, the default of ProteinMPNN's scoring script [3]. OMS-rerank uses M orders $\{\sigma_m\}_{m=1}^{M}$ for each wild type, shared by all of its mutants (Algorithm 1, as in StaB-ddG [12]), and reports $\hat{\tilde{\mu}}_M(\mathrm{seq} \mid X) = \frac{1}{M}\sum_m \log p_{\sigma_m}(\mathrm{seq} \mid X)$. All methods use the same weights and backbone.

PCFG data. The PCFG of §3.2 is a mixture of 16 topics over sequences of length $T{=}24$ with vocabulary size $|\mathcal{V}|{=}60$. The vocabulary is partitioned into four part-of-speech-like slot types: 8 determiners (DET), 16 adjectives (ADJ), 16 nouns (N), and 20 verbs (V). Each sequence is the slot pattern DET ADJ N V repeated six times.

Generation proceeds in two steps. First, a topic $z \in \{1, \ldots, 16\}$ is drawn from a near-uniform prior $\propto [1.05, \ldots, 0.95]$. Second, given the topic, each position is filled independently from a per-(slot, topic) categorical distribution $p(x_t \mid z, \mathrm{slot}(t))$ obtained as a tempered softmax over slot-specific Gaussian logits. The temperatures are DET$=1.50$, ADJ$=0.85$, N$=0.55$ and V$=0.40$, so that verbs carry most of the topic signal while adjectives admit several competitive choices per topic; the logits are drawn with seed 2027. Under this construction, $\log p_{\mathrm{data}}(x \mid \text{prompt})$ is a log-sum over the 16 topics; we compute its exact mode by branch-and-bound, certified optimal on all 128 prompts, and use it as the recovery target in Table 1.

PCFG MDM training. The PCFG MDM is a $201{,}344$-parameter Transformer with hidden dimension $d{=}88$, 4 attention heads, 2 layers, feed-forward dimension 352 and learned positional embeddings, with `[MASK]` as an additional vocabulary entry. We train it on the NELBO of Sahoo et al. [2]: each sequence draws a masking rate $t \sim \mathrm{Unif}(t_{\min}, 1 - t_{\min})$ with $t_{\min}{=}10^{-3}$, masks each position independently with probability t, and weights the cross-entropy of the masked positions by $1/t$. The optimizer is AdamW with weight decay 10^{-4} and learning rate 3×10^{-4}, batch size 512, and 200 epochs over $40{,}000$ training and $4{,}000$ validation sequences. We evaluate the validation NELBO every 5 epochs with $n_{\mathrm{mc}}{=}8$ masking draws per sequence and use the checkpoint with the lowest value (46.72 nats per sequence, at epoch 175). Training takes about 4 minutes on a single NVIDIA RTX 5090.

Sudoku data. The Sudoku data of §5.2 are 9×9 solved grids encoded as length-81 sequences (row-major) over the $|\mathcal{V}|{=}9$ digits. Each training grid is generated by composing the following symmetry-preserving permutations on a single fixed valid base grid: a uniformly random bijection of digits $\{1, \ldots, 9\}$, a uniformly random permutation of the three row-bands together with a uniformly random permutation of rows within each band, the analogous column-band and within-band permutations, and a 0.5-probability transpose. Each permutation preserves validity, so every generated grid is a valid solution. The training set is $10{,}000$ such grids and the validation set is an independently generated $2{,}000$ grids.

Evaluation puzzles are produced by a greedy hint-reduction procedure that starts from a freshly generated solved grid and removes hint cells one at a time in random order, retaining a removal only

if the resulting puzzle still has a unique solution (verified by a complete backtracking search that stops once it finds 2 solutions). The procedure halts when the target hint count is reached, producing a unique-solution puzzle whose solution is the original grid. Hint-reduction can fail at low hint counts; we try up to three random removal orders per grid before discarding it.

Sudoku MDM training. The Sudoku MDM is an $805{,}120$-parameter Transformer with hidden dimension $d{=}128$, 4 attention heads, 4 layers, feed-forward dimension 512 and no dropout. We train it on the same NELBO as the PCFG MDM, divided by $T{=}81$. The optimizer is AdamW with learning rate 3×10^{-4}, weight decay 10^{-2}, betas $(0.9, 0.95)$, gradient clipping at norm 1, batch size 128, and 30 epochs with a 118-step linear warmup, a constant phase, and a cosine decay to $0.1\times$ the learning rate over the last 20% of steps. We evaluate the validation NELBO each epoch on a fixed set of 32 masking draws per validation grid, which removes Monte Carlo noise from the curve in Figure 3, and use the final-epoch checkpoint. Training takes under a minute on a single NVIDIA RTX 5090.

Training curves. Figure 3 shows the validation NELBO (weight $1/t$) of both MDMs during training, per sequence for the PCFG and per token for Sudoku; for a denoiser without time input, its expectation is $\mathbb{E}_x[-\tilde{\mu}(x)]$ up to the first and last steps (§3.1). Neither model shows divergence or collapse.

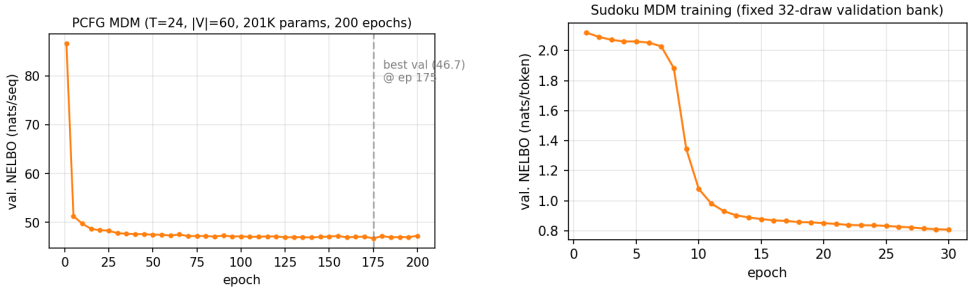


Figure 3: Validation NELBO across training epochs. **Left:** PCFG MDM; it drops from 86.6 nats per sequence at epoch 1 to its minimum 46.7 at epoch 175, and stays within 0.6 of the minimum from epoch 70 on. **Right:** Sudoku MDM; it drops from 2.12 to 0.81 nats per token.

Code to reproduce every experiment, including the PCFG generator, the training and evaluation scripts and all random seeds, is available at `https://github.com/LDeng0205/order-marginalized-scoring`.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction list the documented phenomenon (per-order argmax/ranking disagreement on PCFG and MDLM-OWT), the three propositions (§3, §4), the two algorithms, and the four experimental settings, including the negative result on text infilling; each claim corresponds to a specific section in the paper.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: §6 has a dedicated limitations paragraph: the gains concentrate on tasks where no decoding order is privileged; on text infilling, OMS-rerank does not improve on reranking under a single random order; and the experiments use position-by-position decoding, leaving block-parallel decoding, an adaptive choice of M, and comparisons with other post-hoc scorers to future work.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: The three results (Propositions 1, 2, 3) are stated with their full hypotheses in §3–§4 and proved in Appendix A in the same order they appear in the main text. Appendix B sketches an extension to block decoding; its statements are not claimed as results, and their full proofs are left to future work.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Appendix E lists model architectures, training settings and datasets, and the released code fixes all random seeds; Algorithm 1, Algorithm 2, and Algorithm 3 fully specify the proposed scoring rule, decoder, and beam-search procedure used to obtain the reported numbers.

   Guidelines:

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

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Code to reproduce every experiment, including the PCFG generator, the training and evaluation scripts and all random seeds, is available at `https://github.com/LDeng0205/order-marginalized-scoring` (Appendix E). All third-party assets (MDLM-OWT, ProteinMPNN, Megascale) are publicly available and cited.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Optimizer (AdamW), learning rate, batch size, epochs, model dimensions, evaluation set sizes, and per-experiment hyperparameters (M, N, τ, beam width) are documented in §5 and Appendix E, with training-curve evidence of convergence in Figure 3.

   Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Every experimental table reports an error metric: replicate standard deviation across 5 scoring-order seeds for the PCFG, with a bootstrap standard error over orders for the baseline (Table 1); binomial standard errors $\sqrt{p(1-p)/n}$ for Sudoku (Tables 2 and 7); bootstrap standard errors and 95% bootstrap CIs over $10{,}000$ wild-type-level resamples for Megascale (Table 3); and standard errors over passages, with paired 95% bootstrap CIs, for text infilling (Table 4). The choice of error metric is described in the prose of each subsection.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

Answer: [Yes]

Justification: Costs are given in forward passes for each experiment (§5.2, §5.3 and Table 4); model sizes (201K parameters for the PCFG MDM, 800K for the Sudoku MDM, 1.7M for ProteinMPNN, 169.6M for MDLM-OWT) are given in §5 and the appendices, with training times for the PCFG and Sudoku MDMs in Appendix E and Sudoku decoding times in Appendix C; all experiments were run on a single NVIDIA RTX 5090 with 32 GB of memory.

Guidelines:

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: The paper studies a test-time scoring rule for masked diffusion models on existing public datasets and pretrained checkpoints; it involves no human subjects, no private data, and no foreseeable misuse pathway, and all assets are used in accordance with their published licenses.

Guidelines:

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

10. **Broader impacts**

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

Answer: [N/A]

Justification: The contribution is a foundational test-time scoring rule for masked diffusion models; it does not enable new model capabilities and has no direct path to differentiated positive or negative societal impact beyond the general impacts of the underlying generative-modelling literature it cites.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The released artefacts are a synthetic-PCFG generator, two small task-specific MDM checkpoints (201K and 800K parameters trained on synthetic PCFG and Sudoku data), and evaluation scripts; none of these pose a high misuse risk.

Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The MDLM-OWT checkpoint [2], ProteinMPNN [3], the Megascale dataset [22], and the StaB-ddG code and splits [12] are cited at every point of use; we use each in accordance with its published license (MDLM-OWT: Apache-2.0; ProteinMPNN and StaB-ddG: MIT; Megascale: CC BY 4.0) and we have introduced no derived asset that changes the license terms.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The new assets are the PCFG generator, the trained PCFG and Sudoku MDM checkpoints, and the training and evaluation scripts that produced every table and figure, released with documentation at `https://github.com/LDeng0205/order-marginalized-scoring` (Appendix E).

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

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

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing and no human subjects.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [N/A]

    Justification: The paper involves no human subjects, so no IRB approval is applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not a component of the methodology, theoretical results, or experimental design; they were used as writing and coding assistants.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.