WALK FAST BUT BE CAREFUL: UNDERSTANDING PARALLEL SAMPLING IN MASKED DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we use random walks on graphs as a verifiable sandbox for studying parallel sampling strategies in masked diffusion models (MDMs). We train an MDM on random walk samples from a fixed graph. The graph and transition kernel are never shown to the model and serve as latent structure that is both controllable and enables evaluation. The framework provides a validity check for generated walks and a measure of distributional fidelity through the estimated transition kernel. Using simple graphs, we theoretically prove that parallel unmasking via widely used scores such as lowest entropy is *not* uniformly better than random parallel sampling; even with exact conditional probabilities, performance critically depends on the conditional dependence structure induced by the graph, a phenomenon difficult to isolate in benchmarks like Sudoku. We also develop training-free *bisection samplers* for MDMs, which take logarithmically many steps in the sequence length and are provably exact for random walks if the learned marginals are exact. Experiments on graph-walk tasks confirm that different parallel samplers perform better on different graph structures. Experiments on pretrained MDMs show that bisection-style samplers provide strong speed-quality tradeoffs on OpenWebText generation and reasoning benchmarks including GSM8K, MBPP, and HumanEval. Together, these results use graph walks to uncover conditional dependence as a key principle of parallel MDM sampling and translate this insight into efficient samplers that transfer to language generation and reasoning.

1 INTRODUCTION

Masked diffusion models (MDMs) generate discrete data by iteratively denoising masked tokens (Devlin et al., 2019; Austin et al., 2021a; Campbell et al., 2022; Sahoo et al., 2024; Shi et al., 2024; Lou et al., 2024; Ou et al., 2025). Because the denoiser can be queried on arbitrary partial contexts, inference is inherently any-order: tokens may be revealed left-to-right, adaptively, or in parallel. This flexibility makes parallel sampling particularly attractive: multiple tokens can be revealed from a single denoiser call, substantially reducing the number of function evaluations (NFEs). Recent work has therefore explored multi-token samplers and adaptive policies for deciding which coordinates to reveal together, with token ordering strongly affecting generation quality on structured tasks such as Sudoku (Chang et al., 2022; Shih et al., 2022; Kim et al., 2025; Ben-Hamu et al., 2025; Anari et al., 2026; Wu et al., 2026; Wei et al., 2025).

However, parallel speedup can incur statistical error. With exact conditional marginals, any one-coordinate reveal order is exact by the chain rule, whereas a parallel update replaces the true block conditional by a product of one-coordinate marginals. If blocks of positions $B_1, \dots, B_R$ are revealed over R denoiser calls, the resulting error is bounded by (Ben-Hamu et al., 2025)

$$\mathrm{KL}\left(\nu(X)\middle\|p_\theta(X)\right) \le \underbrace{\mathbb{E}\left[\sum_r \sum_{i\in B_r} \Delta_\theta(i; X_{U_r})\right]}_{\Delta_\theta \text{ (marginal approximation error)}} + \underbrace{\mathbb{E}\left[\sum_r \mathrm{TC}_\nu(X_{B_r} \mid X_{U_r})\right]}_{\mathrm{TC} \text{ (conditional total correlation)}}. \tag{1}$$

Here $X = (X_1, \dots, X_L) \sim \nu$ be a random sequence of length L, where X_i is the token at position i. For $U \subseteq [L] = \{1, \dots, L\}$, write $X_U = (X_i)_{i\in U}$ and $x_U = (x_i)_{i\in U}$ for a realization of X_U. Here U_r denotes the positions revealed before round r, and $p_\theta(X)$ denotes the sampler's output distribution. The expectations are over target samples $X \sim \nu$ and any independent randomness used to select the

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

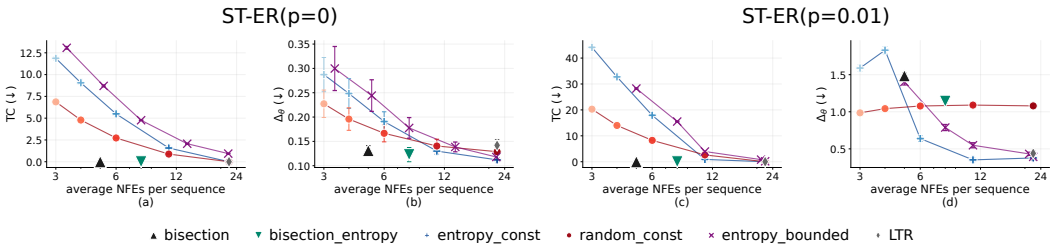

Figure 1: Error decomposition for endpoint-conditioned generation. Models are trained only on unconditional random walks; endpoints are revealed at test time. ST–ER(p) is a spanning tree with non-tree edges added independently with probability p. We plot the two terms in (1), total correlation (TC) and marginal approximation error Δ_θ, against NFEs. TC is substantially larger than Δ_θ, and the relative performance of parallel samplers changes with graph structure. Bisection variants achieve low error with fewer NFEs. The corresponding coherence and TV results are reported in Figure 12.

blocks[1]: at each round, the sampling strategy selects B_r using only previously revealed values, then reveals the target values X_{B_r}. The quantity $\Delta_\theta(i; x_U) := \mathrm{KL}\left(\nu(X_i \mid x_U) \middle|\middle| p_\theta(X_i \mid x_U)\right)$ measures marginal approximation error, while $\mathrm{TC}_\nu(X_B \mid x_U) := \mathrm{KL}\left(\nu(X_B \mid x_U) \middle|\middle| \prod_{i\in B} \nu(X_i \mid x_U)\right)$ is the conditional total correlation, measuring the dependence among tokens revealed together. Thus, parallel sampling depends both on marginal prediction accuracy and on the conditional dependence of the selected block.

The two sources of error are hard to isolate in language, where model error, sampler error, and evaluation noise are intertwined. Sudoku has therefore been used as a controlled benchmark in prior masked-diffusion work. However, commonly used instances with many constraints often have a unique solution (Shah et al., 2024; Kim et al., 2025). For a unique solution $x^\star$, the intrinsic block-dependence error vanishes for any block B: $\nu(X_B \mid x_U) = \prod_{i\in B} \nu(X_i \mid x_U) = \prod_{i\in B} \mathbf{1}\{X_i = x_i^\star\}$. Thus, while such benchmarks can test adaptive unmasking strategies that reduce model approximation error, they do not necessarily test whether a strategy identifies conditionally independent tokens that can be safely unmasked in parallel. In this work we ask,

How do we mechanistically test when parallel unmasking is statistically accurate, and design parallel samplers that respect the conditional dependence structure of the target distribution?

To answer these questions, we turn to graph path problems, which are increasingly used as controlled benchmarks for reasoning in language models (Bachmann & Nagarajan, 2024; Wang et al., 2024; Zhong et al., 2025; Huang et al., 2026; Ye et al., 2025a; Wang et al., 2026). Our goal is complementary to this line of work: we isolate how dependence structure affects parallel sampling. We use MDMs to sample from the distribution over random walks on a graph. This setting gives us direct access to both path validity and distributional fidelity. Exploiting the conditional dependence structure of random walks, we develop an efficient *training-free* parallel sampler, *bisection*. Experiments on unconditional OpenWebText generation and conditional reasoning benchmarks including GSM8K, HumanEval, and MBPP (Cobbe et al., 2021; Chen et al., 2021; Austin et al., 2021b) show that bisection achieves strong speed–quality tradeoffs relative to standard parallel samplers.

We fix a graph $G = (V, E)$ and walk length L; training examples are walks $X_{1:L} \in V^L$, with vertices treated as categorical tokens and vocabulary $V \cup \{\texttt{[MASK]}\}$. The model never sees G or its transition kernel, only samples from the walk distribution, which induces a controllable *latent* dependency structure through graph topology, endpoint constraints, and walk order. Standard random walks are first-order Markov, while higher-order variants allow the next state to depend on the previous k states, capturing memory effects in web navigation, network flows, and complex networks (Chierichetti et al., 2012; Rosvall et al., 2014; Benson et al., 2016; 2017).

Graph walks provide an exact validity check: a generated sequence is coherent iff every consecutive token pair is an edge. Since the ground-truth transition kernel is tractable for both unconditional walks and endpoint-conditioned bridges, they also allow direct measurement of distributional fidelity via TV distance (Section 5). Figure 1 shows that small changes in graph structure can reverse sampler rankings. We further isolate the two terms in (1) (details in Appendix J.1). In both graphs, the intrinsic

[1]Equality holds if the unmasking policy is a deterministic function of the revealed context x_U

dependence error (TC) is roughly an order of magnitude larger than the marginal approximation error, and sampler rankings largely track TC. This suggests that parallel unmasking depends critically on conditional dependence, not uncertainty alone.

Contributions. Our primary contributions are summarized below.

1. A controlled benchmark for parallel masked diffusion. We formulate graph random walks as a controlled benchmark for parallel sampling in MDMs. The model only has sample access to random walks on the graph, but the graph is *available for evaluation*. The benchmark supports unconditional generation, endpoint-conditioned walks, and higher-order random walks, with coherence and TV distance as direct evaluation metrics.

2. Separations between parallel sampling strategies. Simple Directed Acyclic Graph (DAG) constructions show that lowest-entropy two-at-a-time unmasking can outperform random unmasking on some graphs and underperform it on others, demonstrating that no local uncertainty score is uniformly optimal.

3. Bisection samplers for MDMs. We introduce bisection-style samplers that reveal separator positions or separator blocks and then recurse. For order-k Markov walks, the sampler is exact under perfect conditionals for both unconditional walks and endpoint-conditioned walks, with $O(k(1+\log(L/k)))$ NFEs.

4. Transfer beyond graph walks. On pretrained MDMs, bisection improves speed–quality tradeoffs on OpenWebText, GSM8K, MBPP, and HumanEval, showing that its gains extend to language generation, mathematical reasoning, and code generation.

2 Preliminaries on graph-walks and MDMs

We use a masked diffusion model (MDM) (Sahoo et al., 2024) (Appendix A) to generate order-k random-walk samples $X_{1:L} \in [V]^L$, represented as vertex sequences on graph $G = (V, E)$. Given an initial law ρ_k on length-k histories, and k-Markov transition kernel P_k, the random-walk distribution is $\nu_k(x_{1:L}) = \rho_k(x_{1:k}) \prod_{\ell=k}^{L-1} P_k(x_{\ell+1} \mid x_{\ell-k+1:\ell})$.

The model vocabulary is $V \cup \{\texttt{[MASK]}\}$. During standard MDM training, coordinates of $X_{1:L} \sim \nu_k$ are randomly masked, and the model is trained to predict the clean token at each masked position. The model never sees the graph, transition kernel, or graph features; it only observes sampled walks. Given revealed context x_U, the model returns one-coordinate marginals $p_i(\cdot \mid x_U)$ which we interpret as an approximation to $\nu_k(X_i = \cdot \mid X_U = x_U)$, for $i \in [L] \setminus U$.

Generation starts from an initial revealed set U_0. For unconditional generation, $U_0 = \emptyset$. For conditional generation, we reveal a held-out path's endpoints and set $U_0 = \{1, L\}$ with $\widetilde{X}_1 = s$ and $\widetilde{X}_L = t$, targeting the endpoint-conditioned bridge law $\nu_k^{s,t} := \nu_k(\cdot \mid X_1 = s, X_L = t)$. At round r, let $M_r = [L] \setminus U_r$. Given $\widetilde{X}_{U_r} = x_{U_r}$, an unmasking policy chooses a block $B_r \subseteq M_r$, samples independently $\widetilde{X}_i \sim p_i(\cdot \mid x_{U_r}),\ i \in B_r$, and updates $U_{r+1} = U_r \cup B_r$.

The baseline policies differ only in how they choose B_r. Random unmasking chooses either one uniformly random coordinate from M_r, or a uniformly random subset of size $\min\{b, |M_r|\}$ in the b-at-a-time version. Greedy entropy chooses the smallest $H_i = -\sum_{a \in V} p_i(a) \log p_i(a)$; greedy confidence chooses the largest $C_i = \max_{a \in V} p_i(a)$; and greedy margin chooses the largest $\Delta_i = p_i^{(1)} - p_i^{(2)}$, where $p_i^{(1)} \geq p_i^{(2)}$ are the two largest probabilities under p_i.

3 Understanding parallel unmasking via graph structure

Simple graph constructions show that no single uncertainty-based parallel unmasking heuristic may be uniformly optimal. The key issue is not merely which positions are most certain, but which positions can be revealed together without introducing conditional-dependence errors. Adaptive one-at-a-time samplers help reduce approximation error when marginals are not perfectly learned (Kim et al., 2025). For parallel sampling, while such adaptive samplers may help when low-entropy positions act as separators, they can fail catastrophically by clustering updates within a dependent

component and sampling coupled choices independently. Figures 2 and 3 illustrate these behaviors with two simple graph families.

We use the standard perfect conditional assumption to decouple the intrinsic dependence error of the parallel sampler from the model's approximation error.

Assumption 1 (Perfect conditional marginals)**.** *For every partial assignment x_U satisfying $\nu(X_U = x_U) > 0$, the model's conditional marginal for the masked coordinates is exact, i.e, $p_i(\cdot \mid x_U) = \nu(X_i = \cdot \mid X_U = x_U)$ for all $i \in [L] \backslash U$.*

Lemma 5 in Appendix B shows that sequential unmasking is order-invariant under Assumption 1. This follows directly from the probability product rule and is standard in any-order autoregressive modeling (Anari et al., 2026; Kim et al., 2025; Uria et al., 2016; 2014). For simplicity, we use DAGs with a context length such that each level corresponds to a position in the generated walk. We analyze two-at-a-time random (TR) and entropy (TE) samplers, breaking ties uniformly at random.

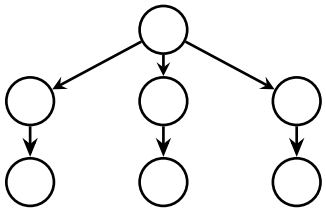

Figure 2: Star-DAG with $d = 3$ chains of length $m = 2$.

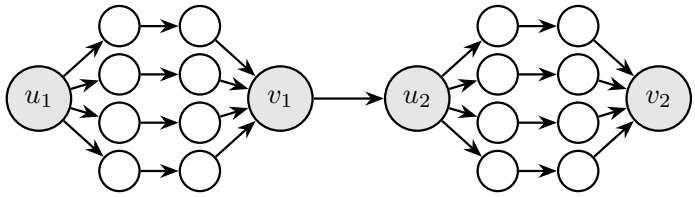

Figure 3: The (K, L) bottleneck DAG, shown with $K = 2$ bottleneck gadgets and $L = 4$ parallel corridors.

3.1 STAR-DAG: PARALLEL ENTROPY BEATS RANDOM

Definition 1. *Fix integers $d \geq 2$ and $m \geq 1$. Let $G(d, m)$ be the directed graph with one root vertex r and d disjoint directed chains of length m emanating from r: $r \to v_{i,1} \to \cdots \to v_{i,m}$. for $i \in [d]$.*

Lemma 1. *Let the context length be $L = m + 1$ and Assumption 1 hold. Denote by coh^{TE} and coh^{TR} the probabilities of generating a coherent directed path in $G(d, m)$ with TE, and TR samplers respectively. Then*

$$\mathrm{coh}^{TE} = 1, \qquad \mathrm{coh}^{TR} = \frac{2}{m+1} + \frac{m-1}{d(m+1)}.$$

The target path is determined by a single hidden chain index $I \in [d]$: the root coordinate is deterministic, and every non-root coordinate reveals the same index I. Lowest-entropy unmasking first selects the deterministic root and one non-root coordinate, which identifies I and makes all remaining coordinates deterministic; random two-at-a-time unmasking succeeds only if its first pair contains the root or if two independently sampled non-root coordinates happen to choose the same chain. The proof is in Appendix C.

3.2 (K, L) BOTTLENECK DAG: PARALLEL RANDOM BEATS ENTROPY

Definition 2. *A (K, L) bottleneck DAG is defined as a graph with $2K$ bottleneck nodes where each pair is connected by a corridor of L parallel directed paths with two nodes each.*

Lemma 2. *Let Assumption 1 hold and coh^{TR}, coh^{TE} denote the probabilities of a coherent path generation in the (K, L) bottleneck DAG with TR and TE respectively. If K is even and $L > 1$, then*

$$\liminf_{K \to \infty} \left(\mathrm{coh}^{TR} - \mathrm{coh}^{TE}\right) \geq \frac{2}{3} + \frac{1}{3L}.$$

In the bottleneck DAG, errors arise only when two positions from the same dangerous corridor are revealed together but the independently sampled tokens belong to different parallel paths. TR spreads its pairs across all masked positions and therefore rarely hits the same corridor, whereas TE first consumes the low-entropy bottleneck nodes and leaves many corridor positions to be paired with each other; Appendix D formalizes this by comparing the resulting success recurrences.

These separations suggest judging a parallel unmasking rule by how each update affects the conditional dependencies among the still-masked positions. The aim is to make current and subsequent parallel updates distributionally correct. In the Star-DAG, entropy achieves this by first identifying which branch the path follows, whereas in the bottleneck DAG, random unmasking works better by spreading its choices out, reducing conflicting choices within a corridor. Marginal uncertainty is therefore only a proxy; the fundamental objective is to construct conditionally valid parallel updates.

A similar principle may apply to creative language tasks: a proof strategy or key lemma, a story twist, or a central thesis can constrain subsequent dependent details. Although structural anchors may divide the remaining generation into weakly coupled subproblems, naive parallel sampling may still produce locally inconsistent updates. For graph walks, the next section proposes parallel samplers that are provably correct for Markov chains.

4 A NEW FAMILY OF BISECTION SAMPLERS



Figure 4: Bisection with $k = 1$, each word is one token. Blue tokens are sampled in parallel at the midpoints of the remaining masked intervals; gray tokens were revealed in earlier rounds.

Bisection sampling uses graph walks' Markov structure to split generation into conditionally independent subproblems. For a first-order walk, $X_{<t} \perp\!\!\!\perp X_{>t} \mid X_t$, so revealing the midpoint allows the two remaining intervals to be sampled recursively in parallel. Figure 4 illustrates this order on sample text. For an order-k walk, a contiguous block of k tokens captures the memory needed to separate the two sides. In each active masked interval of length ℓ, we therefore reveal the middle block of $\min\{k, \ell\}$ tokens sequentially, then recurse on the remaining intervals in parallel. Middle blocks keep the recursion balanced; unrestricted pivots near endpoints may unbalance the subproblems. Algorithm 1 in Appendix E gives the full procedure.

The following lemma formalizes conditional independence between blocks in distinct masked intervals for endpoint-conditioned walks.

Lemma 3 (Conditional independence for order-k Markov bridges)**.** *Let $1 \le k \le L$, and let ν_k be the law of an order-k Markov chain on $X_{1:L}$. For endpoints s, t with $\nu_k(X_1 = s, X_L = t) > 0$, let $\nu_k^{s,t} := \nu_k(\cdot \mid X_1 = s, X_L = t)$ be the bridge law. Let $U \subseteq [L]$ contain the endpoints, and suppose the revealed coordinates in U form contiguous separator chunks $I_0, \dots, I_m$, ordered from left to right, with $|I_j| \ge k$ for every internal chunk $1 \le j \le m-1$. Let B_j be the masked interval between I_{j-1} and I_j, and let $S_j \subseteq B_j$ be any subset. Then, for every x_U with $\nu_k^{s,t}(X_U = x_U) > 0$,*

$$\nu_k^{s,t}(X_{S_1} = x_{S_1}, \dots, X_{S_m} = x_{S_m} \mid X_U = x_U) = \prod_{j=1}^{m} \nu_k^{s,t}\big(X_{S_j} = x_{S_j} \mid X_U = x_U\big)\,.$$

At each bisection level, the algorithm selects one separator block per masked interval. Lemma 3 allows different blocks to be sampled in parallel, while sequential sampling within each block preserves its internal dependencies. Under Assumption 1, each substep therefore samples from the correct joint conditional law. The chain rule over levels yields correctness. Each level uses at most k sequential substeps, and balanced splits give $\mathcal{O}(1 + \log(L/k))$ levels. The next lemma gives the guarantee for unconditional and endpoint-conditioned walks; proofs appear in Appendix F.

Lemma 4 (Exactness of order-k bisection sampling)**.** *Let $1 \le k \le L$, and let ν be an order-k random-walk law, either unconditional or conditioned on fixed reachable endpoints. Let Assumption 1 hold. If $\widetilde{\nu}_k^{\mathrm{bis}}$ denotes the output law of Algorithm 1, initialized with $U = \emptyset$ or with the prescribed endpoints, respectively, then $\widetilde{\nu}_k^{\mathrm{bis}} = \nu$. Moreover, the parallel sampling depth is $\mathcal{O}(k \log(L/k))$.*

Score-guided bisection sampling. In practice, learned marginals need not satisfy Assumption 1. To address model approximation error, we combine adaptive, score-guided selection with separators that control dependence error. In each active interval, we choose the lowest-uncertainty pivot in its middle half, then grow a contiguous separator: given the current active block, we grow the separator

in the direction where uncertainty is the least. At each substep, we compare the scores of the masked tokens immediately adjacent to its left and right ends and reveal the lower-uncertainty token (or the only available neighbor). We continue until the separator contains $\min\{k, \ell\}$ tokens, for interval length ℓ. The middle-half restriction balances the recursion; contiguous growth produces an order-k separator whenever masked tokens remain on both sides. The parallel depth remains $\mathcal{O}(k \log(L/k))$. Each level uses at most k sequential denoiser calls, with all active intervals processed in parallel. Since the pivot lies in the middle half, every remaining interval has length at most $\lfloor 3\ell/4 \rfloor$. Thus there are $\mathcal{O}(\log(L/k))$ levels before intervals have length at most k and are filled in one final level. Algorithm 2 in Appendix E gives the full procedure. Under Assumption 1, score-guided bisection is also exact for Markov chains: conditional on the revealed history, each substep samples at most one coordinate per previously separated interval, so the same argument applies (Appendix F).

Section 6 shows that these samplers also improve speed-quality tradeoffs for language generation. This may be because the MDM conditions on all revealed tokens at each recursive step, including those in distant intervals, using long-range dependencies throughout sampling.

5 Experiments

We use the graph-walk MDM setup from Section 2 to study how latent graph structure affects parallel samplers. In all graph experiments, the vocabulary is the vertex set V (and a [MASK] token) with $|V| = 500$. For each graph, we train an MDM on random-walk samples from the unconditional path law ν. For order $k = 1$, unless otherwise stated, we use lazy random walks on an undirected graph $G = (V, E)$, so that $P_1(v \mid u) = 1/(\deg(u) + 1)$ if $v = u$ or $(v, u) \in E$, and 0 otherwise. All reported numbers in the graph experiments are averaged (with error bars) over five random runs. See Appendix J for details.

Graph families. ST–ER(p) is a spanning tree augmented by independently adding each non-tree edge with probability p, which controls the graph density while preserving connectivity.

Metrics. Graph walks allow exact validation via both path coherence and TV distance, without an external judge model. For an order-k walk with initial law ρ_k and transition kernel P_k, we define the coherence of a sequence $\widetilde{X}_{1:L}$ as

$$\mathrm{coh}_k(\widetilde{X}_{1:L}) = \mathbf{1}\{\rho_k(\widetilde{X}_{1:k}) > 0\} \prod_{\ell=k}^{L-1} \mathbf{1}\Big\{P_k(\widetilde{X}_{\ell+1} \mid \widetilde{X}_{\ell-k+1:\ell}) > 0\Big\}.$$

Average coherence is the fraction of coherent generated samples, a support-validity check. Since coherence alone does not measure agreement with the transition statistics, we also report row-weighted transition total variation. Let $\widehat{P}_k(\cdot \mid h)$ be the empirical next-token distribution after history $h \in V^k$, and let $\widehat{\omega}(h)$ be the empirical frequency of that history. We define

$$\mathrm{TV}_k = \frac{1}{2} \sum_{h \in V^k : \widehat{\omega}(h) > 0} \widehat{\omega}(h) \sum_{v \in V} \Big|\widehat{P}_k(v \mid h) - P_k(v \mid h)\Big|.$$

For endpoint-conditioned generation, we also report conditional TV. It compares empirical next-step transition probabilities with exact endpoint-conditioned probabilities, averaging errors across positions and endpoint pairs. See (8) and Appendix J.2.

5.1 Unconditional generation

In the unconditional task, generation starts from the fully masked sequence and targets the unconditioned walk law ν. We evaluate unmasking policies on ST–ER(p) graphs.

Figure 5 shows both metrics on ST-ER($p = 0$), a spanning tree, and ST-ER($p = 0.01$). As in Figure 1, changing only the edge density substantially changes the gap between random and entropy-based samplers in both metrics.

Moreover, bisection-style samplers substantially reduce the NFEs while preserving coherence and transition fidelity, whereas random or entropy-based parallel unmasking can lose coherence depending on the graph structure.

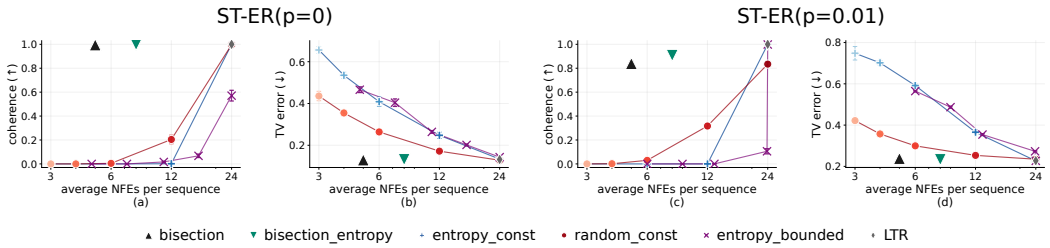

Figure 5: Unconditional random-walk generation on ST-ER graphs. (a,b) report coherence and TV error on ST-ER($p = 0$). (c,d) report the same metrics on ST-ER($p = 0.01$).

We also evaluate parallel samplers on order-k unconditional random walks. Unlike first-order Markov random walks, we choose $X_{\ell+1}$ to depend only on its *distant ancestors* in the trajectory $X_{1:\ell}$ (see Appendix J.3). This more sharply tests preservation of higher-order structure: a locally plausible sequence may still violate the distant-history rule.

Figure 6 shows that coherence improves as the separator size approaches the true Markov order $K = 4$. Increasing the separator size from 1 to 4 raises mean coherence from 0.608 to 0.659 for divide-half bisection and from 0.766 to 0.827 for entropy-guided bisection. The random and entropy one-token baselines achieve 0.714 and 0.896, respectively.

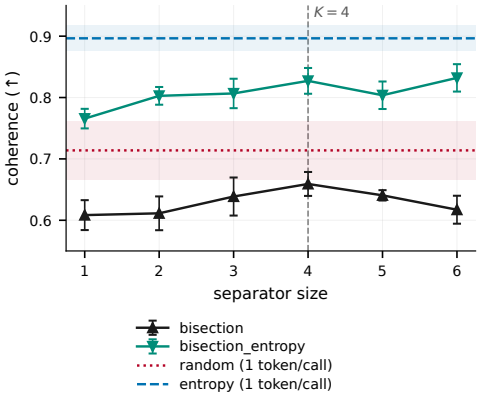

Figure 6: Coherence vs. separator size for order-$K = 4$ walks (dashed line). Bars and shading show SD across four batches of 256 samples.

5.2 Endpoint-conditioned bridge generation

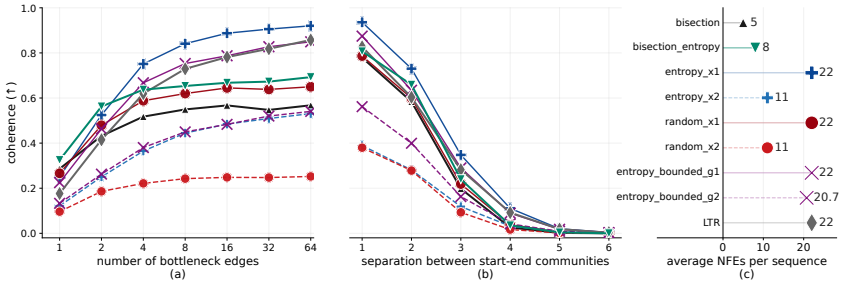

Figure 7: Endpoint-conditioned bridge generation on bottleneck graph families. (a) shows the two-community bottleneck task (b) shows the chain-of-communities task (c) reports the average NFEs per sequence for each sampler.

For endpoint-conditioned generation, we reveal x_1 and x_L from a held-out walk $x_{1:L}$; the sampler fills in the bridge. For ST–ER(p) graphs, Figure 1 reports TC and marginal approximation error. Figure 12 in Appendix J reports the corresponding coherence and conditional TV results. We also use two bottleneck tests that probe compositional bridge generation.

Two-community bottleneck. The first setting has two components $V = V_1 \sqcup V_2$, $|V_1| = |V_2| = 250$, each sampled independently from the ST-ER(p) family. We vary the number b of inter-component edges, training a separate model for each b. At test time, the revealed endpoints lie in different subgraphs. As b decreases, the bridge becomes harder since the sampler must discover a rarer crossing. This tests whether the sampler can compose local motion within communities with the required inter-community crossing.

Figure 7(a) shows sampler performance, with average NFEs in Figure 7(c). With few bottleneck edges, one-at-a-time random and entropy samplers perform similarly; their ordering flips as more bottleneck edges are added. Entropy-guided bisection outperforms all baselines with few bottlenecks while using far fewer NFEs.

Chain of communities. We use ten communities $V = V_1 \sqcup \cdots \sqcup V_{10}$, each with $|V_j| = 50$ and sampled from the same ST–ER(p) family. Consecutive communities are connected by a single edge (r_j, r_{j+1}), with $r_j \in V_j$ and $r_{j+1} \in V_{j+1}$, with no edges between non-consecutive communities.

We test endpoint-conditioned generation with endpoints in increasingly distant communities. For separation h, we construct a walk $x_{1:L}$ with $x_1 \in V_j$ and $x_L \in V_{j+h}$ by joining the sequence $r_j, r_{j+1}, \ldots, r_{j+h}$ to two random walks confined to the endpoint communities. Both local walks start at the designated vertices; we reverse the left walk and choose their lengths so that the combined walk has L vertices. At test time, we reveal only x_1 and x_L and mask all interior vertices. This construction guarantees a valid bridge of the requested length.

A coherent bridge must traverse all h inter-community edges, so increasing h requires the sampler to recover a longer sequence of hidden crossings. Figure 7(b) shows that one-at-a-time greedy entropy achieves the highest coherence, whereas two-at-a-time greedy entropy and random unmasking have comparable coherence. Bisection and entropy-guided bisection achieve coherence close to one-at-a-time random unmasking, using approximately 5 and 8 NFEs per sequence, respectively, compared with 22 for the one-at-a-time baselines (Figure 7(c)).

6 Implications beyond graph random walks

We next test whether the bisection schedules motivated by graph random walks improve speed–quality tradeoffs for pretrained masked diffusion models on mathematical reasoning, code generation, and unconditional text generation (see dataset details in Appendix G).

6.1 Reasoning benchmarks

Setup. We evaluate parallel samplers on GSM8K (Cobbe et al., 2021), MBPP (Austin et al., 2021b) and HumanEval (Chen et al., 2021) with Dream-v0-Instruct-7B (Ye et al., 2025b) and LLaDA-1.5 (Zhu et al., 2025), as shown in Figure 8. All three benchmarks are prompted zero-shot and decoded with 512 output tokens at temperature 0.1 with top-$p = 0.9$. We report pass@1 with one sample per problem versus NFE. The DUS (dilated) sampler (Luxembourg et al., 2025) uses semi-AR decoding.

Results. Figure 8 shows that bisection offers a strong trade-off between accuracy and NFEs on both models. Bisection performs well across all three tasks at roughly 30 NFEs, and attains the highest observed accuracy on the code-generation benchmarks, MBPP and HumanEval. On GSM8K, bisection performs best among the non-autoregressive samplers, while dilated semi-autoregressive decoding achieves higher accuracy at larger sampling budgets. This may reflect the largely left-to-right structure of step-by-step mathematical solutions. These results suggest unmasking motivated by graph walks can benefit reasoning and code generation with diffusion language models.

6.2 Unconditional language generation

Setup. We evaluate samplers on OpenWebText (OWT) (Gokaslan et al., 2019) using 1024 generated samples per method with top-$p = 0.9$ sampling. We use the MDM checkpoint trained on OWT (Sahoo et al., 2024; Wang et al., 2025) and compare the samplers in Figure 9. The x-axis reports average NFEs per sequence. We report MAUVE (Pillutla et al., 2021) with 2048 reference validation samples, token entropy and uni-gram repetition scores. MAUVE compares generated and reference text representations in a quantized embedding space; higher scores indicate closer agreement with the reference distribution. Generative perplexity and bi-/tri-gram repetition scores appear in Appendix G. Higher is better for MAUVE and entropy, while lower is better for repetition.

Results. Figure 9 shows a clear compute–quality tradeoff. The autoregressive baseline performs best overall, but uses roughly 1023 NFEs per sequence. Among non-autoregressive methods, bisection-based samplers perform best: entropy-guided bisection reaches MAUVE ≈ 0.16 at about 145 NFEs,

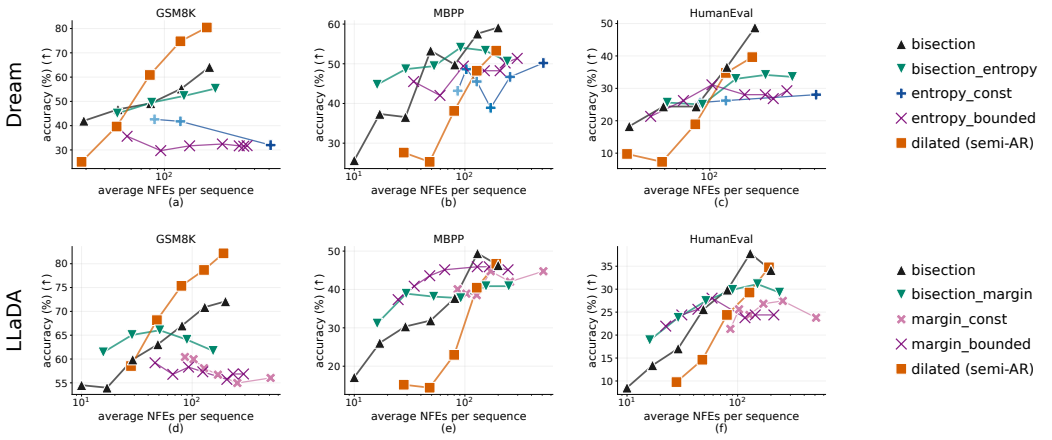

Figure 8: Reasoning benchmarks: pass@1(↑) for Dream-v0-Instruct-7B (top) and LLaDA-1.5 (bottom) on GSM8K, MBPP and HumanEval, all prompted zero-shot.

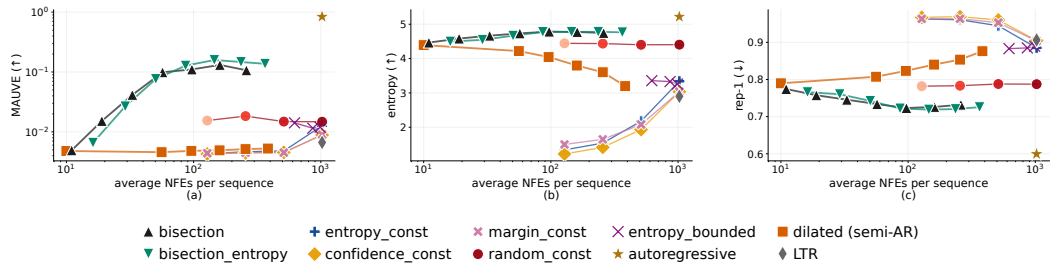

Figure 9: Speed-quality tradeoffs for language generation with a pretrained OpenWebText MDM across samplers: (a) MAUVE (↑); (b) entropy (↑); (c) token repetition rate (↓). The x-axis reports average NFEs per sequence.

while standard bisection reaches MAUVE ≈ 0.13 at about 161 NFEs. Both match or exceed the diversity of all other non-autoregressive baselines.

Low generative perplexity alone does not predict quality: Figure 10 in Appendix G shows that constant-entropy and entropy-bounded samplers achieve low perplexity, but have poor MAUVE, consistent with prior observations (Wang et al., 2025). Overall, bisection, especially entropy-guided bisection, gives the strongest non-autoregressive speed-quality tradeoff in the main-paper metrics, suggesting that coarse-to-fine unmasking based on exact conditional separation in graph walks also improves language generation. We show unconditional OpenWebText samples from the two best-performing bisection settings in Figure 11 in Appendix G.

7 Conclusion and Future Directions

We introduced graph random walks as a controlled benchmark for parallel MDM sampling. The hidden graph lets us evaluate support validity through coherence and distributional fidelity through transition TV. We showed that parallel unmasking depends on conditional independence rather than local uncertainty alone, and proposed bisection-style samplers using Markov separators for logarithmic parallel depth under perfect conditional marginals.

Future work includes quantifying denoiser estimation error propagation through sequential, parallel, and bisection samplers to extend our theory beyond exact conditional marginals. Further graph-walk stress tests include directed graphs, weighted kernels, nonreversible walks, time-inhomogeneous Markov chains, and larger community hierarchies. Our real-world experiments suggest graph walks can inform practical decoding for reasoning tasks and unconditional language generation. Future work includes scaling experiments and combining bisection with remasking or verifier-guided correction.

ETHICS STATEMENT

Our experiments use synthetic graph-walk data, publicly released language models, and existing text datasets and reasoning benchmarks; we collect no new human-subject data. Licenses for the OpenWebText experiments are discussed in Appendix I. Pretrained language models and web-text data may contain societal biases or harmful content. Our evaluation measures sampling quality and computational cost in controlled experiments. Deployment requires separate validation and safeguards appropriate to the intended use.

REPRODUCIBILITY STATEMENT

Appendices B–D and F contain the proofs, and Appendix E gives the bisection algorithms. Sections 5 and 6, together with Appendices G and J, describe the experimental procedures and additional results. Appendix H reports compute resources. We will release the code upon acceptance.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing and revising proofs and to support experiment implementation. We also used these tools for manuscript editing, literature searches and citation checking, code development, and figure preparation. The authors take responsibility for the final manuscript, mathematical claims, code, figures, and experimental results, including content produced with the assistance of generative AI.

REFERENCES

Nima Anari, Carlo Baronio, CJ Chen, Alireza Haqi, Frederic Koehler, Anqi Li, and Thuy-Duong Vuong. Parallel sampling via autospeculation. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*, pp. 1168–1179, 2026.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tyEyYT267x.

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*, volume 34, pp. 17981–17993, 2021a.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b.

Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 2296–2318. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/bachmann24a.html.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=WBcBhT1NKO.

Austin R. Benson, David F. Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016. doi: 10.1126/science.aad9029.

Austin R Benson, David F Gleich, and Lek-Heng Lim. The spacey random walk: A stochastic process for higher-order data. *SIAM Review*, 59(2):321–345, 2017.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 28266–28279, 2022.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5453–5512. PMLR, 2024. URL https://proceedings.mlr.press/v235/campbell24a.html.

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*, pp. 11315–11325, 2022.

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

Flavio Chierichetti, Ravi Kumar, Prabhakar Raghavan, and Tamas Sarlos. Are web users really markovian? In *Proceedings of the 21st international conference on World Wide Web*, pp. 609–618, 2012.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4239. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f0d629a734b56a642701bba7bc8bb3ed-Abstract-Conference.html.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jQj-_rLVXsj.

Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. DiffusionBERT: Improving generative masked language models with diffusion models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 4521–4534, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.248. URL https://aclanthology.org/2023.acl-long.248/.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/67d96d458abdef21792e6d8e590244e7-Abstract.html.

Jianhao Huang, Zhanpeng Zhou, Renqiu Xia, Baharan Mirzasoleiman, Weijie Su, and Wei Huang. How transformers learn to plan via multi-token prediction, 2026. URL https://arxiv.org/abs/2604.11912.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham M. Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. Outstanding Paper Award.

Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences*, 110(52):20935–20940, 2013.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343, 2022. URL https://openreview.net/forum?id=3s9IrEsjLyk.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 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*. PMLR, 2024. URL https://openreview.net/forum?id=CNicRIVIPA.

Omer Luxembourg, Haim Permuter, and Eliya Nachmani. Plan for speed: Dilated scheduling for masked diffusion language models. *arXiv preprint arXiv:2506.19037*, 2025.

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. *arXiv preprint arXiv:2502.09992*, 2025. URL https://arxiv.org/abs/2502.09992.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sMyXP8Tanm.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/260c2432a0eecc28ce03c10dadc078a4-Abstract.html.

Martin Rosvall, Alcides V Esquivel, Andrea Lancichinetti, Jevin D West, and Renaud Lambiotte. Memory in network flows and its effects on spreading dynamics and community detection. *Nature communications*, 5(1):4630, 2014.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Mariano Marroquin, Justin T. Chiu, Alexander M. Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=L4uaAR4ArM.

Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 56674–56702. Curran Associates, Inc., 2024. doi: 10.52202/079017-1805. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/67b31ca159553d5593e62d7b998d63ea-Paper-Conference.pdf.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=xcqSOfHt4g.

Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive models the right way. In *Advances in Neural Information Processing Systems*, volume 35, pp. 2762–2775, 2022.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=BYWWwSY2G5s.

Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 467–475. PMLR, 2014.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2503.00307. arXiv:2503.00307v4, revised 7 February 2026.

Siwei Wang, Yifei Shen, Shi Feng, Haoran Sun, Shang-Hua Teng, and Wei Chen. Alpine: Unveiling the planning capability of autoregressive learning in language models, 2024. URL https://arxiv.org/abs/2405.09220.

Yuxiang Wang, Yu Xiang, Baojian Zhou, Qifang Zhao, Keyue Jiang, Yanghua Xiao, and Xiaoxiao Xu. On the trainability of masked diffusion language models via blockwise locality, 2026. URL https://arxiv.org/abs/2604.24832.

Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with slowfast sampling: The three golden principles. *arXiv preprint arXiv:2506.10848*, 2025. URL https://arxiv.org/abs/2506.10848.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3Z3Is6hnOT.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning, 2025a. URL https://arxiv.org/abs/2410.14157.

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

Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Quanquan Gu. Diffusion language models can perform many tasks with scaling and instruction-finetuning. *arXiv preprint arXiv:2308.12219*, 2023. URL https://arxiv.org/abs/2308.12219.

Qimin Zhong, Hao Liao, Siwei Wang, Mingyang Zhou, Xiaoqun Wu, Rui Mao, and Wei Chen. Understanding and enhancing the planning capability of language models via multi-token prediction, 2025. URL https://arxiv.org/abs/2509.23186.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025. URL https://arxiv.org/abs/2505.19223.

Appendix organization. The appendix is organized as follows.

- **Appendix A**: Provides additional preliminaries and related work on masked diffusion models, any-order unmasking, and higher-order random walks on graphs.
- **Appendix B**: Gives the deferred proof that exact one-token-at-a-time unmasking is order-invariant under perfect conditional marginals.
- **Appendix C**: Proves the Star-DAG separation, where entropy-based two-at-a-time unmasking succeeds while random two-at-a-time unmasking can fail.
- **Appendix D**: Proves the bottleneck-DAG separation, where random two-at-a-time unmasking outperforms entropy-based two-at-a-time unmasking.
- **Appendix E**: Gives the full bisection algorithms.
- **Appendix F**: Contains the formal conditional-independence and exactness proofs for order-k bisection sampling.
- **Appendix G**: Provides dataset descriptions and additional language experiments on the OpenWebText dataset
- **Appendix H**: Reports compute resources and runtime details for the graph-walk training and sampler-evaluation experiments.
- **Appendix I**: Provides licenses for external codebase and dataset used.
- **Appendix J**: Provides additional coherence tables for graph random-walk samplers across graph families, conditioning regimes, and unmasking policies.

A Additional preliminaries and related work

The main text uses masked diffusion models through a black-box conditional marginal view: given a partially revealed sequence x_U, the denoiser returns one-coordinate distributions $p_i(\cdot \mid x_U)$ for masked coordinates $i \notin U$. This appendix records the standard MDM details behind this view and expands the related-work discussion.

A.1 Additional details on masked diffusion models

Absorbing corruption. Masked diffusion models for discrete data commonly use an absorbing-state corruption process (Austin et al., 2021a; Campbell et al., 2022; Sahoo et al., 2024; Shi et al., 2024). Let $\mathcal{V}$ be the clean vocabulary and let $[\texttt{MASK}] \notin \mathcal{V}$ be the absorbing mask token. For a decreasing schedule $\alpha_t \in [0,1]$ with $\alpha_0 = 1$ and $\alpha_1 = 0$, the forward process masks coordinates independently:

$$q_t(z \mid x) = \prod_{i=1}^{L} \left(\alpha_t \delta_{x_i}(z_i) + (1-\alpha_t)\delta_{[\texttt{MASK}]}(z_i)\right).$$

Thus Z_t is exactly a partially revealed version of X, with revealed set $U_t = \{i : Z_{t,i} \neq [\texttt{MASK}]\}$.

Reverse update. For $0 \le s < t \le 1$, if $Z_{t,i} \neq [\texttt{MASK}]$, the absorbing coordinate is copied unchanged. If $Z_{t,i} = [\texttt{MASK}]$, then the exact reverse posterior, conditioned on the clean token, unmasks with probability $\rho_{s,t} := (\alpha_s - \alpha_t)/(1-\alpha_t)$ and otherwise remains masked:

$$\Pr(Z_{s,i} = \cdot \mid Z_{t,i} = [\texttt{MASK}], X_i) = \rho_{s,t}\delta_{X_i}(\cdot) + (1-\rho_{s,t})\delta_{[\texttt{MASK}]}(\cdot).$$

Since X_i is unknown at generation time, the learned sampler replaces δ_{X_i} by a denoiser $p_{\theta,i}(\cdot \mid Z_t)$. Equivalently, a finite reverse step first chooses a subset of currently masked coordinates and then samples each chosen coordinate independently from its one-coordinate denoising distribution. In the infinitesimal limit $s = t - dt$, $\rho_{s,t} = O(dt)$, so for fixed L the probability of revealing two or more coordinates in one infinitesimal interval is $O(dt^2)$. This motivates viewing continuous-time MDM sampling as a one-coordinate-at-a-time jump process, while finite-step or accelerated samplers may reveal multiple coordinates from the same denoiser call.

Training objective. The usual MDM objective is a schedule-weighted denoising cross-entropy over masked coordinates (Sahoo et al., 2024; Shi et al., 2024). In the notation of the main text, a typical loss has the form

$$\mathcal{L}(\theta) = \mathbb{E}_{X\sim\nu,\, t,\, Z_t\sim q_t(\cdot|X)} \sum_{i:Z_{t,i}=[\texttt{MASK}]} w(t)\ \left(-\log p_{\theta,i}(X_i \mid Z_t)\right),$$

for a nonnegative schedule-dependent weight $w(t)$. Our theoretical results do not depend on the exact choice of $w(t)$; they only use the induced conditional-marginal interpretation $p_i(\cdot \mid x_U) \approx \nu(X_i = \cdot \mid X_U = x_U)$, formalized in Assumption 1.

A.2 Related work

Discrete and masked diffusion models. Diffusion models for discrete data have been developed through several closely related formalisms. Early work introduced structured categorical corruptions and multinomial diffusion processes (Austin et al., 2021a; Hoogeboom et al., 2021), while continuous-time formulations model the forward and reverse dynamics as jump processes or continuous-time Markov chains (Campbell et al., 2022; Sun et al., 2023). More recent approaches improve the learning objective or parameterization for discrete denoising, including score-entropy ratio estimation and simplified masked objectives (Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024). Discrete flow models and discrete flow matching give another view in which probability paths on finite state spaces are learned through posterior or denoising objectives (Campbell et al., 2024; Gat et al., 2024). Our work is complementary to these modeling advances: rather than proposing a new training objective, we study how a trained masked denoiser should be queried at inference time.

Diffusion language models. Several works adapt diffusion or denoising ideas to text generation. Continuous-diffusion language models map tokens through continuous embeddings and enable controllable generation (Li et al., 2022; Gong et al., 2023), while masked or absorbing-state language diffusion models operate directly over token vocabularies (He et al., 2023; Sahoo et al., 2024; Shi et al., 2024). Recent large-scale diffusion language models and block diffusion models suggest that diffusion-style generation can scale to stronger language modeling regimes and interpolate between autoregressive and non-autoregressive generation (Ye et al., 2023; Arriola et al., 2025; Nie et al., 2025). These models motivate our focus on sampling schedules: if MDMs are to be competitive at scale, one needs fast parallel samplers that do not destroy global sequence coherence.

Any-order generation and token ordering. Masked language modeling began as a representation-learning objective in models such as BERT (Devlin et al., 2019), but the same ability to condition on arbitrary visible tokens also enables any-order generation. Any-order-autoregressive and masked generative models exploit this flexibility by choosing which coordinate to reveal next (Uria et al., 2014; 2016; Shih et al., 2022). In the exact one-token setting, every reveal order is valid by the chain rule, but practical orderings can substantially affect sample quality. MaskGIT popularized confidence-based iterative decoding (Chang et al., 2022), and recent work shows that token ordering matters sharply in masked diffusion models, including structured tasks such as Sudoku (Kim et al., 2025). Our graph benchmark isolates the same phenomenon in a setting where the latent constraint structure is controllable and directly checkable.

Parallel and accelerated masked decoding. The main computational appeal of masked diffusion is that multiple tokens can be revealed per denoiser call. Recent samplers accelerate generation by choosing larger blocks, adaptive block sizes, or by adding inference-time mechanisms such as KV caching, slow-fast schedules, auto speculation, and remasking (Ben-Hamu et al., 2025; Wu et al., 2026; Wei et al., 2025; Anari et al., 2026; Wang et al., 2025). These methods improve the speed-quality tradeoff, but a parallel update implicitly replaces a true block conditional by a product of one-coordinate marginals. Our contribution is to make this approximation explicit: on graph walks, whether a parallel update is safe is determined by conditional independence in the underlying sequence distribution.

Random walks and higher-order network dynamics. Random walks and higher-order Markov models are standard tools for modeling sequential structure on graphs. Higher-order variants capture memory effects in web navigation, network flows, community detection, and complex networks

(Chierichetti et al., 2012; Rosvall et al., 2014; Krzakala et al., 2013; Benson et al., 2016; 2017). We use these models in a different role: the walk distribution is a controlled sequence distribution for training and testing masked diffusion samplers. The graph and transition rule are hidden from the model but available to the evaluator, giving exact checks of support validity through coherence and, for first-order unconditional walks, distributional fidelity through transition TV.

Structured evaluation for generative models. Open-ended text generation is difficult to evaluate with a single scalar metric; distributional metrics such as MAUVE compare generated and human text in an embedding space (Pillutla et al., 2021), and our language experiments use OpenWebText as the evaluation domain (Gokaslan et al., 2019). Graph walks serve a different purpose: they are not intended as a replacement for language evaluation, but as a mechanistic sandbox where failures can be attributed to support violations or transition-statistic errors. This makes them useful for diagnosing sampler behavior before transferring the resulting schedules back to language generation.

B Deferred Proofs from Section 3

Lemma 5 (Exactness of sequential unmasking)**.** *Let ν be a target distribution on paths $X_{1:L} = (X_1, \dots, X_L)$, and let $\sigma = (\sigma_1, \dots, \sigma_L)$ be an ordering of $\{1, \dots, L\}$. Suppose Assumption 1 holds for this order σ. Consider the sequential sampler $\widetilde{X}_{\sigma_r} \sim p_{\sigma_r}(\cdot \mid \widetilde{X}_{U_{r-1}})$, where $U_{r-1} := \{\sigma_1, \dots, \sigma_{r-1}\}$, for $r = 1, \dots, L$. Let $\widetilde{\nu}_\sigma$ denote the law of $\widetilde{X}_{1:L}$. Then $\widetilde{\nu}_\sigma = \nu$.*

Proof. Fix $x_{1:L} \in \mathcal{V}^L$. If $\nu(x_{1:L}) > 0$, then every prefix $x_{U_{k-1}}$ has positive ν-probability, so Assumption 1 and the probability chain rule give

$$\widetilde{\nu}_\sigma(x_{1:L}) = \prod_{k=1}^{L} p_{\sigma_k}(x_{\sigma_k} \mid x_{U_{k-1}}) = \prod_{k=1}^{L} \nu(X_{\sigma_k} = x_{\sigma_k} \mid X_{U_{k-1}} = x_{U_{k-1}}) = \nu(x_{1:L}).$$

If $\nu(x_{1:L}) = 0$, let k be the first index such that $\nu(X_{U_k} = x_{U_k}) = 0$. Then $\nu(X_{U_{k-1}} = x_{U_{k-1}}) > 0$, while $\nu(X_{\sigma_k} = x_{\sigma_k} \mid X_{U_{k-1}} = x_{U_{k-1}}) = 0$. By Assumption 1, the corresponding multiplicative term in the sampler probability is also zero: $p_{\sigma_r}(x_{\sigma_r} \mid x_{U_{r-1}}) = 0$. Hence $\widetilde{\nu}_\sigma(x_{1:L}) = 0$. □

C A toy graph where entropy-two-at-a-time wins

Lemma 1. *Let the context length be $L = m + 1$ and Assumption 1 hold. Denote by coh^{TE} and coh^{TR} the probabilities of generating a coherent directed path in $G(d, m)$ with TE, and TR samplers respectively. Then*

$$\mathrm{coh}^{TE} = 1, \qquad \mathrm{coh}^{TR} = \frac{2}{m+1} + \frac{m-1}{d(m+1)}.$$

Proof of Lemma 1. The target distribution is supported on the d paths indexed by $I \in [d]$. The first coordinate is deterministic: $X_1 = r$. Every non-root coordinate reveals the same hidden chain index I:

$$X_{t+1} = v_{I,t}, \qquad t = 1, \dots, m.$$

We first analyze entropy unmasking. Initially, $H(X_1) = 0$, since X_1 is deterministic. For every non-root coordinate $t + 1$,

$$X_{t+1} \sim \mathrm{Unif}\{v_{1,t}, \dots, v_{d,t}\},$$

so $H(X_{t+1}) = \log d$. Therefore the greedy lowest-entropy two-at-a-time rule first selects the root coordinate X_1 and one non-root coordinate X_j, $j \geq 2$.

The sampler draws $X_1 = r$ deterministically. It also draws $X_j = v_{i,j-1}$ for some $i \in [d]$. This reveals the hidden chain index $I = i$. Once $I = i$ is known, every remaining coordinate is deterministic:

$$X_{t+1} = v_{i,t}, \qquad t = 1, \dots, m.$$

Thus all later conditional marginals are point masses, and no inconsistency can ever be introduced. Hence $\mathrm{coh}^{\mathrm{TE}}_{d,m} = 1$.

Now consider random two-at-a-time unmasking. The first random block is a uniformly random two-element subset of the $L = m + 1$ positions. The probability that this block contains the root coordinate is

$$\frac{L-1}{\binom{L}{2}} = \frac{2}{L} = \frac{2}{m+1}.$$

If the first block contains the root, then the other selected coordinate reveals the chain index I, after which all remaining coordinates are deterministic. Thus this case succeeds with probability 1.

On the complementary event, the first block contains two non-root coordinates. Suppose the two selected positions are $a, b \geq 2$. Under the product-marginal parallel update, the sampler draws

$$X_a = v_{\widehat{I}_a, a-1}, \qquad X_b = v_{\widehat{I}_b, b-1},$$

where $\widehat{I}_a, \widehat{I}_b \overset{\text{i.i.d.}}{\sim} \mathrm{Unif}([d])$. These two sampled coordinates are jointly extendable to a valid path if and only if they come from the same chain: $\widehat{I}_a = \widehat{I}_b$. This happens with probability $1/d$. If they disagree, no valid path in $G_{d,m}$ contains both sampled vertices, so the sampler has already failed. If they agree, then the common value fixes I, and all remaining coordinates are deterministic, so the sampler succeeds. Therefore

$$\mathrm{coh}^{\mathrm{TR}}_{d,m} = \frac{2}{m+1} + \left(1 - \frac{2}{m+1}\right)\frac{1}{d} = \frac{2}{m+1} + \frac{m-1}{d(m+1)}.$$

□

D A TOY GRAPH WHERE RANDOM-TWO-AT-A-TIME WINS

Lemma 2. *Let Assumption 1 hold and* coh^{TR}, coh^{TE} *denote the probabilities of a coherent path generation in the* (K, L) *bottleneck DAG with TR and TE respectively. If* K *is even and* $L > 1$*, then*

$$\liminf_{K\to\infty} \left(\mathrm{coh}^{TR} - \mathrm{coh}^{TE}\right) \geq \frac{2}{3} + \frac{1}{3L}.$$

Proof of Lemma 2. Let n denote the number of dangerous corridors remaining. Each dangerous corridor contains a pair of internal positions. If these two positions are unmasked in the same parallel round, then the independently sampled channel choices agree with probability $1/L$ and disagree with probability $1 - 1/L$. All other pairs of simultaneously unmasked positions are safe, because they either lie in different corridors or contain a bottleneck position.

We first lower bound the success probability of TR. Since the coordinate choices of TR are independent of the sampled values, the sequence of two-at-a-time choices induces a uniformly random matching M on the initially masked positions. Let m be the number of initially masked positions. For each dangerous pair D_i, the probability that D_i appears as an edge of M is $1/(m-1)$. Therefore, by a union bound,

$$\begin{aligned} 1 - \mathrm{coh}^{\mathrm{TR}}_K &= \Pr\left(\text{TR fails}\right) \\ &\leq \sum_{i=1}^{K} \Pr\left(D_i \in M\right) \Pr\left(\text{channel mismatch} \mid D_i \in M\right) \\ &= \frac{K}{m-1}\left(1 - \frac{1}{L}\right). \end{aligned}$$

The (K, L) bottleneck DAG has at least $3K$ initially masked positions. Hence

$$1 - \mathrm{coh}^{\mathrm{TR}}_K \leq \frac{K}{3K-1}\left(1 - \frac{1}{L}\right),$$

and consequently

$$\liminf_{K\to\infty} \mathrm{coh}^{\mathrm{TR}}_K \geq 1 - \frac{1}{3}\left(1 - \frac{1}{L}\right) = \frac{2}{3} + \frac{1}{3L}.$$

We now upper bound the success probability of TE. The lowest-entropy coordinates are the bottleneck coordinates, so the first K two-at-a-time rounds reveal all $2K$ bottleneck nodes. After this, the only remaining possible errors are the K dangerous internal pairs. Let $\mathrm{coh}_n^{\mathrm{TE}}$ denote the probability of generating a coherent sequence with entropy-based unmasking when n dangerous pairs remain. If the next entropy round selects the two positions from the same dangerous pair, which occurs with probability $1/(2n-1)$, then it succeeds with probability $1/L$ and leaves $n-1$ dangerous pairs. Otherwise it selects positions from two different dangerous pairs, which occurs with probability $(2n-2)/(2n-1)$, and leaves $n-2$ dangerous pairs. Therefore

$$\mathrm{coh}_n^{\mathrm{TE}} = \frac{1}{2n-1}\cdot\frac{1}{L}\mathrm{coh}_{n-1}^{\mathrm{TE}} + \frac{2n-2}{2n-1}\mathrm{coh}_{n-2}^{\mathrm{TE}}.$$

Set $\mathrm{coh}_0^{\mathrm{TE}} = 1$ and define

$$H_n := \max\{\mathrm{coh}_n^{\mathrm{TE}}, \mathrm{coh}_{n-1}^{\mathrm{TE}}\}.$$

Then

$$\mathrm{coh}_{n-1}^{\mathrm{TE}} \le \left(\frac{1}{2n-3}\cdot\frac{1}{L} + \frac{2n-4}{2n-3}\right) H_{n-2},$$
$$\mathrm{coh}_n^{\mathrm{TE}} \le \left(\frac{1}{2n-1}\cdot\frac{1}{L} + \frac{2n-2}{2n-1}\right) H_{n-2} = \left(1 - \frac{1-1/L}{2n-1}\right) H_{n-2}.$$

Thus

$$H_n \le \left(1 - \frac{1-1/L}{2n-1}\right) H_{n-2}.$$

For $n = 2r$, using $H_1 \le 1$ and $H_2 \le 1$,

$$H_{2r} \le \prod_{i=2}^{r}\left(1 - \frac{1-1/L}{4i-1}\right)$$
$$\le \exp\left(-\frac{1-1/L}{4}\sum_{i=2}^{r}\frac{1}{i}\right) \le \left(\frac{2}{r+1}\right)^{(1-1/L)/4}.$$

In particular, $\mathrm{coh}_K^{\mathrm{TE}} \to 0$ along even K. Combining this with the lower bound for TR gives

$$\liminf_{K\to\infty}\left(\mathrm{coh}_K^{\mathrm{TR}} - \mathrm{coh}_K^{\mathrm{TE}}\right) \ge \frac{2}{3} + \frac{1}{3L}.$$

□

E Bisection sampling algorithms

Algorithms 1 and 2 give the basic and score-guided bisection samplers, respectively. In both, $1 \le k \le L$. Initialize $U = \emptyset$ for unconditional sampling, or $U = \{1, L\}$ with fixed reachable endpoint values for bridge sampling. Scores depend only on the revealed context; ties are resolved by a fixed rule or fresh randomness independent of the token draws.

F Deferred Proofs from Section 4

Lemma 3 (Conditional independence for order-k Markov bridges)**.** *Let $1 \le k \le L$, and let ν_k be the law of an order-k Markov chain on $X_{1:L}$. For endpoints s, t with $\nu_k(X_1 = s, X_L = t) > 0$, let $\nu_k^{s,t} := \nu_k(\cdot \mid X_1 = s, X_L = t)$ be the bridge law. Let $U \subseteq [L]$ contain the endpoints, and suppose the revealed coordinates in U form contiguous separator chunks $I_0, \ldots, I_m$, ordered from left to right, with $|I_j| \ge k$ for every internal chunk $1 \le j \le m-1$. Let B_j be the masked interval between I_{j-1} and I_j, and let $S_j \subseteq B_j$ be any subset. Then, for every x_U with $\nu_k^{s,t}(X_U = x_U) > 0$,*

$$\nu_k^{s,t}(X_{S_1} = x_{S_1}, \ldots, X_{S_m} = x_{S_m} \mid X_U = x_U) = \prod_{j=1}^{m}\nu_k^{s,t}\left(X_{S_j} = x_{S_j} \mid X_U = x_U\right).$$

Algorithm 1 Order-k bisection sampling

Require: Length L, order k, conditionals $p_i(\cdot \mid \widetilde{X}_U)$, initial unmasked set U

while $U \neq \{1, \dots, L\}$ **do**
  Let $B_1, \dots, B_m$ be the maximal contiguous intervals in $[L] \setminus U$
  for all $B_j = \{a_j, \dots, b_j\}$ in parallel **do**
    $\ell_j \leftarrow |B_j|$ and $r_j \leftarrow \min\{\ell_j, k\}$
    Choose the middle contiguous block $S_j = \{\tau_j, \tau_j + 1, \dots, \tau_j + r_j - 1\} \subseteq B_j$
  end for
  for $h = 1, \dots, k$ **do**
    for all j such that $h \leq |S_j|$ in parallel **do**
      Let i be the hth position in S_j
      Sample $\widetilde{X}_i \sim p_i(\cdot \mid \widetilde{X}_U)$
    end for
    Add all positions sampled in this substep to U
  end for
end while
return $\widetilde{X}_{1:L}$

Algorithm 2 Order-k score-guided bisection sampling

Require: Length L, order k, conditionals p_i, scores s_i, initial unmasked set U

procedure ParallelSample($\mathcal{C}, U$)
  ▷ `C is a set of disjoint candidate index sets`
  $I \leftarrow \emptyset$
  for all $C \in \mathcal{C}$ *in parallel* **do**
    Choose $i \in \arg\min_{j \in C} s_j(\widetilde{X}_U)$
    Sample $\widetilde{X}_i \sim p_i(\cdot \mid \widetilde{X}_U)$
    $I \leftarrow I \cup \{i\}$
  end for
  $U \leftarrow U \cup I$ ▷ `Synchronous state update`
  return I, U
end procedure

procedure BisectionSample(L, k, U)
  while $U \neq \{1, \dots, L\}$ **do**
    $\mathcal{B} \leftarrow \{B \mid B$ is a maximal contiguous masked chunk in $\{1, \dots, L\} \setminus U\}$
    ▷ `Phase 1: Unmask centers`
    $\mathcal{C}_{\text{center}} \leftarrow \{C \subseteq B \mid B \in \mathcal{B}, C$ is the centered sub-interval of size $\lceil |B|/2 \rceil\}$
    $I, U \leftarrow$ ParallelSample($\mathcal{C}_{\text{center}}, U$)
    Initialize active separators $A_B \leftarrow B \cap I$ for all $B \in \mathcal{B}$
    ▷ `Phase 2: Grow separators outwards`
    for $h = 2, \dots, k$ **do**
      $\mathcal{B}_{\text{active}} \leftarrow \{B \in \mathcal{B} \mid |A_B| < |B|\}$
      $\mathcal{C}_{\text{grow}} \leftarrow \{\{\min A_B - 1, \max A_B + 1\} \cap B \mid B \in \mathcal{B}_{\text{active}}\}$
      $I, U \leftarrow$ ParallelSample($\mathcal{C}_{\text{grow}}, U$) ▷ `Grow blocks in parallel`
      for all $B \in \mathcal{B}_{\text{active}}$ **do**
        $A_B \leftarrow A_B \cup (B \cap I)$ ▷ `Update active separators`
      end for
    end for
  end while
  return $\widetilde{X}_{1:L}$
end procedure

Proof. The order-k factorization is

$$\nu_k(x_{1:L}) = \rho_k(x_{1:k}) \prod_{\ell=k}^{L-1} \nu_k(x_{\ell+1} \mid x_{\ell-k+1:\ell}).$$

Each factor involves at most $k+1$ consecutive coordinates. Since every internal revealed chunk has length at least k, no factor involves masked coordinates from two distinct intervals B_j. Fixing $X_U = x_U$ therefore leaves a product of nonnegative functions, one for each x_{B_j}. Its normalizing constant factors over the intervals, giving

$$\nu_k^{s,t}(X_{[L]\setminus U} = x_{[L]\setminus U} \mid X_U = x_U) = \prod_{j=1}^{m} \nu_k^{s,t}(X_{B_j} = x_{B_j} \mid X_U = x_U).$$

The endpoint conditioning is already included in x_U, since $1, L \in U$. Marginalizing each factor to S_j proves the claim. This argument also applies to the unconditional law, including masked intervals at the sequence boundaries. With only one masked interval, the factorization is immediate. □

Lemma 4 (Exactness of order-k bisection sampling)**.** *Let $1 \le k \le L$, and let ν be an order-k random-walk law, either unconditional or conditioned on fixed reachable endpoints. Let Assumption 1 hold. If $\widetilde{\nu}_k^{\mathrm{bis}}$ denotes the output law of Algorithm 1, initialized with $U = \emptyset$ or with the prescribed endpoints, respectively, then $\widetilde{\nu}_k^{\mathrm{bis}} = \nu$. Moreover, the parallel sampling depth is $\mathcal{O}(k \log(L/k))$.*

Proof. Let U_r be the unmasked set at the beginning of bisection level r. Let $B_1, \dots, B_{m_r}$ be the maximal contiguous masked intervals. At initialization there is at most one such interval; at later levels, every internal revealed separator has length at least k. For each masked chunk B_j, Algorithm 1 chooses a contiguous block $S_j \subseteq B_j$, $k_j := |S_j| = \min\{|B_j|, k\}$. The positions inside each S_j are then unmasked sequentially, while different chunks are processed in parallel.

By Lemma 3 and its unconditional version proved above, conditional on the current unmasked variables $X_{U_r} = x_{U_r}$, the selected blocks from different masked chunks are conditionally independent:

$$\nu\left(X_{S_1} = x_{S_1}, \dots, X_{S_{m_r}} = x_{S_{m_r}} \mid X_{U_r} = x_{U_r}\right) = \prod_{j=1}^{m_r} \nu\left(X_{S_j} = x_{S_j} \mid X_{U_r} = x_{U_r}\right).$$

Now fix a realization $x_{1:L}$ in the support of ν. For each block S_j, write its positions in the order used by the algorithm as $S_j = \{s_{j,1}, \dots, s_{j,k_j}\}$. Because the blocks S_j are conditionally independent given X_{U_r}, conditioning additionally on already unmasked prefixes of these blocks preserves independence across chunks. We will denote by a substep h the sequential unmasking step of each S_j. Therefore, at each substep h, the next positions $\{s_{j,h} : h \le k_j\}$ are conditionally independent given the current unmasked variables.

Let U denote

$$U := U_r \cup \bigcup_{j=1}^{m_r} \{s_{j,t} : 1 \le t \le \min\{h-1, k_j\}\}.$$

Hence, conditional on the current generated values $\widetilde{X}_U = x_U$, the algorithm samples the substep-h positions with probability

$$\prod_{j:\, h \le k_j} p_{s_{j,h}}(x_{s_{j,h}} \mid x_U).$$

By the perfect conditional marginal assumption (Assumption 1),

$$p_{s_{j,h}}(x_{s_{j,h}} \mid x_U) = \nu(X_{s_{j,h}} = x_{s_{j,h}} \mid X_U = x_U).$$

Using the conditional independence from Lemma 3, this product is exactly the true conditional law of the positions sampled at substep h.

Thus, over all $h = 1, \dots, k$, the whole level samples exactly from the true conditional law of the newly unmasked blocks:

$$\begin{aligned}\Pr\left(\widetilde{X}_{S_1} = x_{S_1}, \dots, \widetilde{X}_{S_{m_r}} = x_{S_{m_r}} \mid \widetilde{X}_{U_r} = x_{U_r}\right) \\ = \nu\left(X_{S_1} = x_{S_1}, \dots, X_{S_{m_r}} = x_{S_{m_r}} \mid X_{U_r} = x_{U_r}\right).\end{aligned}$$

After the level is completed, update

$$U_{r+1} = U_r \cup S_1 \cup \cdots \cup S_{m_r}.$$

Applying the blockwise chain rule over all bisection levels gives

$$\widetilde{\nu}_k^{\mathrm{bis}}(x_{1:L}) = \prod_r \nu\left(X_{S_1^{(r)}} = x_{S_1^{(r)}}, \ldots, X_{S_{m_r}^{(r)}} = x_{S_{m_r}^{(r)}} \mid X_{U_r} = x_{U_r}\right) = \nu(x_{1:L}).$$

Therefore $\widetilde{\nu}_k^{\mathrm{bis}} = \nu$.

It remains to bound the number of rounds. At each bisection level, the algorithm unmasks a contiguous block of at most k positions in each active masked chunk. The selected block lies in the middle of the chunk, so the largest remaining masked chunk decreases by a constant factor until its length is at most k. Thus the number of bisection levels is $\mathcal{O}(1 + \log(L/k))$. Each level requires at most k sequential substeps to unmask the selected block, while different masked chunks are processed in parallel. Hence the total parallel sampling depth is $\mathcal{O}(k(1 + \log(L/k)))$. □

Score-guided bisection. The same argument proves exactness of Algorithm 2. Fix a level and condition on the revealed history before a substep, including the index choices. These choices depend only on revealed values and independent tie-breaking randomness, so they reveal no additional information about masked values. The masked intervals present at the start of the level remain conditionally independent after conditioning on the values subsequently revealed within them. Each substep chooses at most one coordinate in each of these intervals, so its product of exact marginals equals the true joint conditional law. At the end of the level, each new separator is contiguous and has length k, unless its interval has been filled completely. The separator condition therefore holds at the next level, and the chain-rule argument above gives the target law. A pivot in the centered interval of size $\lceil \ell/2 \rceil$ leaves at most $\lfloor 3\ell/4 \rfloor$ positions on either side; growing the separator can only shorten these intervals. Thus there are $\mathcal{O}(1 + \log(L/k))$ levels, each with at most k sequential substeps.

G Additional details for real world examples

Datasets. OpenWebText (OWT) is a large corpus of web text collected from URLs shared on Reddit (Gokaslan et al., 2019). GSM8K contains grade-school math word problems requiring multi-step reasoning (Cobbe et al., 2021). MBPP consists of short Python programming problems specified in natural language (Austin et al., 2021b), while HumanEval contains hand-written programming problems used to evaluate functional correctness of generated code (Chen et al., 2021).

Additional metrics for language experiment Figure 10 reports generative perplexity under GPT2-Large, bi-gram and tri-gram repetition rates for the same OpenWebText experiment. All bisection-based samplers, especially entropy-guided bisection, maintain low repetition rates, while maintaining generative perplexity comparable to the AR model. Constant-entropy and entropy-bounded samplers achieve low perplexity, but have poor MAUVE, and high repetition rates consistent with prior observations (Wang et al., 2025). These metrics support the same conclusion as the main-paper results: bisection-based samplers provide the strongest non-autoregressive speed-quality tradeoff.

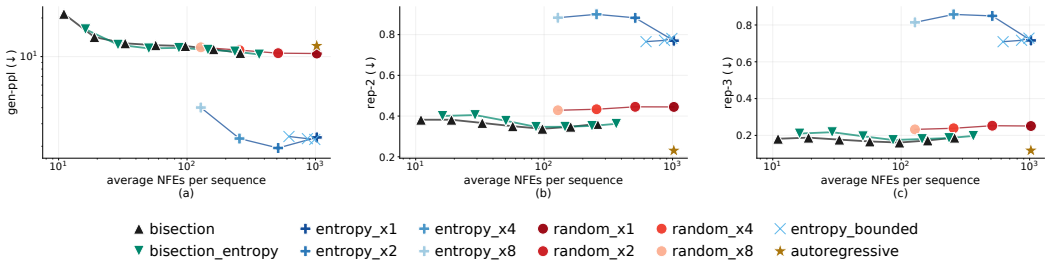

Figure 10: Additional language-generation metrics for the pretrained OpenWebText MDM. We report (a) generative perplexity under GPT2-Large (↓), and (b) bi-gram repetition rate (↓), (c) tri-gram repetition rate (↓) The x-axis is the average NFEs per sequence.

Bisection, k=32 (161 NFEs, MAUVE 0.131)

```
smartphones. As of the second quarter of 2017, nearly 6 million devices
have been connected to an Internet of Things (IoT) network, known as IoT
devices. Currently, it is estimated that as many as 20% of IoT devices
are connected to a home and will be affected by both the home automation
and command and control industries. Experts estimate that up to 40% of
home-connected devices are using telepresence, as well as telepresence
technology to remotely control sensors or other connected devices. It is
estimated that connected sensors account for a third of all IoT devices,
but that doesn't account for the other 80% of connected devices. The IoT
can also be used for a host of other products, such as medical devices
and gaming controllers. Many smart devices rely
```

Entropy-guided bisection, k=16 (146 NFEs, MAUVE 0.159)

```
that test was found to meet safety standards. The results were completed
over two weeks at the department's technology center where the test was
led by assistant director Jay Somentere. The test at KSTO was created in
the 1970s for the University of Oregon's Department of Applied Science.
"It's a safe and sound environment for both pedestrians and bicyclists,"
Somentere said. "Our goal is to be safe in the state of mind of every
pedestrian or bicyclist in Oregon. We're committed to it." Somentere
said his test had been using a three-axis sensor, but had to be updated,
at some point, for the use of an advanced digital imaging camera that
can detect the presence of multiple forms of movement in the roadway.
"There are two things that the test
```

Figure 11: Unconditional OpenWebText samples from the two best-performing bisection settings, truncated to the first 150 words.

H Compute Resources

All model training and sampler evaluation runs were performed on the TACC Vista cluster. GPU experiments used Vista Grace-Hopper nodes, each with one NVIDIA H200 GPU with 96GB HBM3 memory, a 72-core NVIDIA Grace CPU, and approximately 120GB of host memory. CPU-only preprocessing, aggregation, and plotting were run on Vista CPU/login nodes; the node used for these auxiliary jobs exposed 144 ARM Neoverse-V2 cores and 237GiB of system memory. These CPU-only jobs were negligible compared with model training and sampler evaluation.

Each 100k-sample graph model trained in roughly 30 minutes on a single Vista GPU node, i.e., well under one GPU-hour per model. Sampler evaluation time varied with the conditioning task. All-sampler unconditional sweeps were the shortest, taking roughly 10 minutes per model to generate all samples for each sampler. Endpoint-conditional sweeps took roughly 25 minutes for each ST–ER graph and about two hours for the bottleneck-family sweep, for roughly three GPU-hours total across the final table settings. Bottleneck-conditional sweeps were the most expensive: the final bottleneck-family sweeps took between about 30 minutes and 1.5 hours each, totaling roughly five GPU-hours for the reported bottleneck-conditional table.

I Licenses

The MDM codebase and the pretrained `kuleshov-group/mdlm-owt` checkpoint are released under the Apache-2.0 license. The Hugging Face OpenWebText dataset card lists the dataset packaging under CC0-1.0, while noting that the curators do not own the underlying web text; the Hugging Face GPT-2 Large model card lists the model under the MIT license.

J Additional details for Graph Random-Walk Samplers

Notation. All graphs are unweighted and have an additional self-edge. ST–ER(p) denotes the spanning-tree plus Erdős–Rényi graph family used in the experiments: a spanning-tree backbone

Table 1: Graph data and training settings used for the coherence tables. ST–ER denotes a spanning-tree backbone plus independent random non-tree edges.

Setting	Value
Model	DDiT model with hidden size 256, conditioning dimension 64, 4 blocks, 4 heads, sigma-scaled logits, untied embeddings
Diffusion	Absorbing-state MDM, substitution parameterization, continuous time, log-linear noise, antithetic time sampling
Training size	100,000 walks per graph family
Test/cache size	1,000 held-out walks per graph family
Walk kernel	Lazy Markov random walk on each graph family
Walk length	Fixed length 24 nodes
Optimizer	AdamW, lr 3×10^{-4}, betas $(0.9, 0.999)$, eps 10^{-8}, wd 0
Schedule	Cosine decay with 5,000 warmup steps, warmup/min lr 10^{-6}
Training steps	50,000 steps, global batch size 256, bf16 precision
Regularization	Dropout 0.1, EMA 0.9999, gradient clip 1.0
Validation/checkpointing	Every 2,000 steps; checkpoint monitor is validation walk coherence; tables use the last checkpoint
Evaluation	512 unconditional samples, or 512 prompts with 32 samples per prompt for conditional settings
ST–ER $p = 0$	$p = 0.0$
ST–ER avg. degree 7	$p = 0.0100683294$
Bottleneck $m = 2$	$b \in \{1, 8, 64\}$, edge probability 0.02
Bottleneck $m = 10$	chain communities, edge probability 0.106377551

guarantees connectivity, and each remaining non-tree edge is added independently with probability p. Bottleneck graphs are composed of m dense communities connected by rare inter-community bridge edges; b is the number of bridge edges per adjacent community pair. Endpoint-conditional experiments condition only on the prescribed start and terminal nodes of the walk. Bottleneck-conditional experiments further condition on the bottleneck-crossing constraints in the bottleneck graph family, and therefore test whether the sampler can satisfy both endpoint constraints and the required inter-community transitions. In sampler names, a suffix such as `_x4` means that four masked positions are updated per model call, while `_exponential` uses a multiplicative growing update budget.

ST-ER(p) graph construction. We begin by constructing a spanning-tree backbone. Starting from a single root vertex, we grow a connected set one vertex at a time: at each step, we attach one not-yet-added vertex to one vertex already in the connected set, both chosen uniformly at random. After $|V| - 1$ steps, this gives a spanning tree T, and therefore guarantees that the graph is connected. We then add each remaining non-tree edge independently with probability p. The tree backbone guarantees connectivity, while the Erdős-Rényi edges controls the average degree.

Model training details. The graph experiments use a masked discrete diffusion model over tokenized random walks. The backbone is the DDiT configuration: hidden size 256, conditioning dimension 64, 4 transformer blocks, 4 attention heads, dropout 0.1, untied input/output embeddings, and sigma-scaled logits. All graph models use the absorbing-state diffusion parameterization with substitution loss, continuous time ($T = 0$), a log-linear noise schedule with $\sigma_{\min} = 10^{-4}$ and $\sigma_{\max} = 20$, antithetic time sampling, and EMA decay 0.9999.

All reported graph models are trained from scratch on synthetic random-walk datasets generated by our code. The following data sizes and training budgets apply to the coherence tables; the order-K experiment uses the settings in Section J.3. Each dataset has $N = 500$ nodes, fixed walk length 24, 100,000 training walks, and 1,000 held-out test walks. Optimization uses AdamW with learning rate 3×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and weight decay 0. We train for 50,000 steps with global batch size 256, gradient clipping at 1.0, bfloat16 precision, and a cosine decay schedule with linear warmup for the first 10% of training steps. The warmup starts at learning rate 10^{-6} and the cosine schedule decays to minimum learning rate 10^{-6}. Validation and checkpointing are run every 2,000 training steps, and the checkpoint selected for the table evaluations is the last checkpoint from each run.

For the two-community bottleneck comparison, lower-bridge graphs are nested subgraphs of higher-bridge graphs: the $b = 64$ graph is generated first, the $b = 8$ graph is obtained by deterministically trimming bridge edges from the $b = 64$ graph, and the $b = 1$ graph is obtained by trimming the $b = 8$ graph. This keeps the within-community graph structure fixed while varying only the number of inter-community bridges. The ten-community bottleneck experiment uses one chain-of-communities graph and reports aggregate bottleneck-conditional metrics as well as span-specific bottleneck-conditional metrics.

All entries report coherence as mean $\pm$ standard deviation. Unconditional experiments use 512 samples per sampler. Conditional and bottleneck-conditional experiments use 512 prompts with 32 samples per prompt. Standard deviations are computed by splitting samples or prompts into four equal groups and taking the standard deviation of the four group means. The ST–ER average-degree-7 entries use the 100k-training-sample model from the primary experiments, not the sample-scaling 12.5k model.

J.1 Decomposition of the two sources of error for random walks

This section illustrates how we compute the two sources of error in (1), i.e., the marginal approximation error and the dependence error, for first order random-walks. All expectations are over target samples $X \sim \nu$ and the randomness of the block rule. Throughout this section, $p_i(\cdot \mid x_{U_r})$ denotes the network's output at position i restricted to V. Concretely, we take the held-out test walks, run each sampler's block rule on the network's outputs at the true context X_{U_r} exactly as during generation, and then reveal the true values X_{B_r} rather than sampling.

Exact conditional marginals. Fix a round r and context x_{U_r}. Let $\delta_\ell(v) = \mathbf{1}\{v = x_\ell\}$ for $\ell \in U_r$ and $\delta_\ell \equiv 1$ for $\ell \in M_r$, and define

$$\begin{aligned} &\alpha_1(v) = \rho(v)\,\delta_1(v), \quad \alpha_{\ell+1}(v) = \delta_{\ell+1}(v) \sum_u \alpha_\ell(u)\,P(v \mid u), \\ &\beta_L(u) = 1, \qquad\qquad \beta_\ell(u) = \sum_v P(v \mid u)\,\delta_{\ell+1}(v)\,\beta_{\ell+1}(v). \end{aligned} \tag{2}$$

By the Markov property, $\alpha_\ell(v) = \nu(X_\ell = v,\ X_{U_r \cap [\ell]} = x_{U_r \cap [\ell]})$ and $\beta_\ell(u) = \nu(X_{U_r \cap \{\ell+1,\dots,L\}} = x_{U_r \cap \{\ell+1,\dots,L\}} \mid X_\ell = u)$, so for every masked $i \in M_r$,

$$\gamma_i(v) \;:=\; \nu(X_i = v \mid x_{U_r}) \;=\; \frac{\alpha_i(v)\,\beta_i(v)}{\sum_w \alpha_i(w)\,\beta_i(w)}. \tag{3}$$

Marginal approximation error. With γ_i from (3) and p_i the model marginal at the same context,

$$\sum_{i \in B_r} \Delta_\theta(i; x_{U_r}) \;=\; \sum_{i \in B_r} \mathrm{KL}\big(\gamma_i \,\big\|\, p_i(\cdot \mid x_{U_r})\big) \;=\; \sum_{i \in B_r} \sum_{v \in V} \gamma_i(v)\big[\log \gamma_i(v) - \log p_i(v \mid x_{U_r})\big]. \tag{4}$$

This is the only place the trained network enters. It is finite since $p_i > 0$ on V, and it does not depend on how B_r is grouped.

Conditional total correlation. $\mathrm{TC}_\nu(X_{B_r} \mid x_{U_r})$ depends only on ν and on which positions were revealed together; the network plays no role. Call a maximal run of consecutive masked positions a *segment*; two masked positions lie in the same segment iff no revealed position lies strictly between them.

Lemma 6 (Pairwise reduction of TC_ν for first-order walks)**.** *Fix a context x_{U_r} with $\nu(X_{U_r} = x_{U_r}) > 0$.*

(i) Let $i_1 < i_2$ be masked positions in the same segment, $g = i_2 - i_1$, and $P^g(v \mid u)$ the g-step transition probability. Then

$$\pi(v \mid u) \;:=\; \nu(X_{i_2} = v \mid X_{i_1} = u,\ x_{U_r}) \;=\; \frac{P^g(v \mid u)\,\beta_{i_2}(v)}{\beta_{i_1}(u)}, \tag{5}$$

where $\beta_{i_1}(u) = \sum_w P^g(w \mid u)\,\beta_{i_2}(w)$.

(ii) List the positions of B_r in increasing order. Then

$$\begin{aligned} \mathrm{TC}_\nu(X_{B_r} \mid x_{U_r}) &= \sum_{\substack{(i_1,i_2)\ \text{consecutive in } B_r \\ \text{same segment}}} I(X_{i_1}; X_{i_2} \mid x_{U_r}), \\ I(X_{i_1}; X_{i_2} \mid x_{U_r}) &= \sum_u \gamma_{i_1}(u)\, \mathrm{KL}\big(\pi(\cdot \mid u) \,\big\|\, \gamma_{i_2}\big). \end{aligned} \tag{6}$$

In particular $\mathrm{TC}_\nu(X_{B_r} \mid x_{U_r}) = 0$ whenever B_r contains at most one position per segment, as for our bisection sampler with $k = 1$.

Proof. (i) Summing ν over all coordinates other than i_1, i_2 and U_r, the positions left of i_1 contribute $\alpha_{i_1}(u)$, those right of i_2 contribute $\beta_{i_2}(v)$, and the positions strictly between i_1 and i_2 are all masked, so summing over them composes g copies of P. Hence $\nu(X_{i_1} = u, X_{i_2} = v \mid x_{U_r}) \propto \alpha_{i_1}(u)\, P^g(v \mid u)\, \beta_{i_2}(v)$; dividing by $\gamma_{i_1}(u) \propto \alpha_{i_1}(u)\beta_{i_1}(u)$ gives the kernel, and iterating the backward recursion over the g unclamped positions gives the normaliser.

(ii) Conditionally on x_{U_r}, the factors $P(x_{\ell+1} \mid x_\ell)$ that couple two different segments involve a revealed coordinate and are constants, so distinct segments are independent and TC_ν is additive over them. Within a segment, the same computation as in (i) with m retained positions $i_1 < \cdots < i_m$ gives a joint law $\propto \alpha_{i_1}(u_1) \prod_s P^{i_{s+1}-i_s}(u_{s+1} \mid u_s)\, \beta_{i_m}(u_m)$, a Markov chain, whose entropy telescopes as $H(\gamma_{i_1}) + \sum_s H(X_{i_{s+1}} \mid X_{i_s})$. Substituting into $\mathrm{TC}_\nu = \sum_s H(\gamma_{i_s}) - H(X_{i_1}, \ldots, X_{i_m})$ leaves $\sum_s [H(\gamma_{i_{s+1}}) - H(X_{i_{s+1}} \mid X_{i_s})] = \sum_s I(X_{i_s}; X_{i_{s+1}})$. The expression for I in (6) is the standard form $I = \mathbb{E}_{u \sim \gamma_{i_1}} \mathrm{KL}(\pi(\cdot \mid u) \,\|\, \gamma_{i_2})$, valid because $\gamma_{i_2} = \sum_u \gamma_{i_1}(u)\pi(\cdot \mid u)$. □

Figure 1 reports the average TC and marginal approximation errors aggregated over rounds, i.e., we sum (4) and (6) over rounds per walk and average over the held-out walks. The following lemma identifies the sum with the left-hand side of (1). Let $p_\theta(x) := \prod_r \prod_{i \in B_r} p_i(x_i \mid x_{U_r})$ along the teacher-forced schedule of x; for a block rule that is deterministic given the revealed values, which includes all greedy and bisection samplers, this is exactly the sampler's output law $p_\theta(X)$ in (1).

Lemma 7. *For any target law ν on V^L and any block rule that selects B_r from x_{U_r} alone,*

$$\mathbb{E}_{X \sim \nu}\big[\log \nu(X) - \log p_\theta(X)\big] = \mathbb{E}_{X \sim \nu}\Big[\sum_r \sum_{i \in B_r} \Delta_\theta(i; X_{U_r})\Big] + \mathbb{E}_{X \sim \nu}\Big[\sum_r \mathrm{TC}_\nu(X_{B_r} \mid X_{U_r})\Big]. \tag{7}$$

If the rule is deterministic given the revealed values, the left side is $\mathrm{KL}(\nu(X) \,\|\, p_\theta(X))$ and (1) holds with equality. If the rule uses independent randomness, the sampler's law is a mixture over schedules, and by convexity of KL in its second argument the left side of (1) is at most the expectation of (7) over that randomness.

Proof. Since $U_{r+1} = U_r \sqcup B_r$ and B_r is a function of x_{U_r}, the definition of conditional probability gives $\log \nu(x) = \sum_r \log \nu(x_{B_r} \mid x_{U_r})$ for every x in the support. Subtracting $\log p_\theta(x)$ and taking the expectation under ν, condition on X_{U_r} in the r-th term: given $X_{U_r} = x_{U_r}$ the block is fixed and $X_{B_r} \sim \nu(\cdot \mid x_{U_r})$, so the term equals $\mathrm{KL}\big(\nu(X_{B_r} \mid x_{U_r}) \,\|\, \prod_{i \in B_r} p_i(\cdot \mid x_{U_r})\big)$. Inserting $\prod_i \nu(X_i \mid x_{U_r})$ inside the logarithm splits this into $\mathrm{TC}_\nu(X_{B_r} \mid x_{U_r}) + \sum_{i \in B_r} \Delta_\theta(i; x_{U_r})$, using that the i-th marginal of $\nu(X_{B_r} \mid x_{U_r})$ is $\nu(X_i \mid x_{U_r})$. □

J.2 Endpoint-conditional TV evaluation

Let $G = (V, E)$ be the graph, P its random-walk transition matrix, and L the number of nodes in each walk. For an endpoint condition $c = (s, e)$, the true conditional transition kernel at position $t \in \{1, \ldots, L-1\}$ is

$$P_t^c(i, j) = \Pr(X_{t+1} = j \mid X_t = i, X_L = e) = \frac{P_{ij}(P^{L-t-1})_{je}}{(P^{L-t})_{ie}},$$

whenever the denominator is positive.

Given M_c generated walks conditioned on c, define

$$C_t^c(i,j) = \sum_{r=1}^{M_c} \mathbf{1}\Big\{x_t^{(r,c)} = i,\ x_{t+1}^{(r,c)} = j\Big\}$$

$$n_t^c(i) = \sum_{j \in V} C_t^c(i,j)$$

$$\widehat{P}_t^c(i,j) = \frac{C_t^c(i,j)}{n_t^c(i)} \qquad \text{for } n_t^c(i) > 0.$$

For each observed row, its TV error is

$$d_t^c(i) = \begin{cases} \dfrac{1}{2} \displaystyle\sum_{j \in V} \left| \widehat{P}_t^c(i,j) - P_t^c(i,j) \right|, & (P^{L-t})_{ie} > \varepsilon, \\ 1, & (P^{L-t})_{ie} \le \varepsilon, \end{cases}$$

where $\varepsilon = 10^{-15}$ is the numerical cutoff used to identify states that cannot reach the prescribed endpoint.

Let $\mathcal{C}$ be the set of distinct sampled endpoint conditions. If condition c appears m_c times among R endpoint draws from the empirical training endpoint distribution, its weight is $\omega_c = m_c/R$. The reported conditional TV error is

$$\widehat{\mathrm{TV}}_{\mathrm{cond}} = \frac{\displaystyle\sum_{c \in \mathcal{C}} \omega_c \sum_{t=1}^{L-1} \sum_{\substack{i \in V \\ n_t^c(i) > 0}} n_t^c(i)\, d_t^c(i)}{\displaystyle\sum_{c \in \mathcal{C}} \omega_c \sum_{t=1}^{L-1} \sum_{i \in V} n_t^c(i)}. \tag{8}$$

Thus, transition-row errors are weighted by generated transition counts and endpoint-draw multiplicities. Counts include transitions from incoherent generated walks.

For the heavy conditional-TV evaluation, we use $L = 24$, $R = 64$ endpoint draws, and $M_c = 12{,}288$ generated walks per distinct endpoint condition.

Trend summary. The results support the central claim that the efficiency–coherence tradeoff is governed by the conditional-dependence structure induced by the unmasking order. Greedy constant-x and exponential schedules degrade sharply because their score rules often select local clusters of tokens from a single denoiser call. This effect is especially pronounced for confidence-based decoding, where locally confident neighboring positions are likely to be conditioned on one another but are nevertheless updated in parallel. In contrast, entropy-guided bisection preserves high coherence while reducing sequential depth: its recursive middle-out schedule separates the next revealed positions across subintervals, limiting local dependence among simultaneous updates. This is reflected in the tables, where bisection entropy is the strongest accelerated samplers and consistently performs well compared with methods that obtain speedup by updating several positions from the same denoiser call. The bottleneck experiments further show that graph structure controls the difficulty of conditional generation: coherence is lowest when inter-community crossings are rare ($b = 1$) or must traverse longer community chains ($m = 10$), and improves as wider bottlenecks provide more valid bridge choices.

Table 2: Unconditional coherence across samplers and graph families.

Sampling method	ST–ER $p = 0$ lazy 0.5	ST–ER avg. deg. 7 lazy 0.125	Bottleneck $m = 2, b = 1$	Bottleneck $m = 2, b = 8$	Bottleneck $m = 2, b = 64$	Bottleneck $m = 10$
bisection	0.996 ± 0.004	0.777 ± 0.051	0.812 ± 0.062	0.840 ± 0.055	0.799 ± 0.029	0.887 ± 0.004
bisection entropy	0.996 ± 0.004	0.844 ± 0.031	0.869 ± 0.012	0.891 ± 0.018	0.830 ± 0.038	0.916 ± 0.024
greedy_entropy	0.998 ± 0.003	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
greedy_entropy_x2	0.002 ± 0.003	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_x4	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_x6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_x8	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_exponential	0.002 ± 0.003	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
random	0.988 ± 0.012	0.783 ± 0.026	0.818 ± 0.028	0.830 ± 0.019	0.758 ± 0.017	0.879 ± 0.031
random_x2	0.146 ± 0.024	0.248 ± 0.013	0.199 ± 0.025	0.250 ± 0.034	0.258 ± 0.032	0.127 ± 0.037
random_x4	0.000 ± 0.000	0.023 ± 0.012	0.004 ± 0.004	0.008 ± 0.006	0.006 ± 0.003	0.004 ± 0.004
random_x6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.004 ± 0.004	0.000 ± 0.000
random_x8	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
random_exponential	0.320 ± 0.054	0.006 ± 0.006	0.004 ± 0.004	0.004 ± 0.007	0.002 ± 0.003	0.012 ± 0.009
greedy_confidence	0.996 ± 0.004	0.998 ± 0.003	0.996 ± 0.004	0.992 ± 0.006	0.990 ± 0.006	0.926 ± 0.012
greedy_confidence_x2	0.002 ± 0.003	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.002 ± 0.003	0.000 ± 0.000
greedy_confidence_x4	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_confidence_x6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_confidence_x8	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_margin	0.994 ± 0.003	0.973 ± 0.004	0.977 ± 0.020	0.971 ± 0.016	0.936 ± 0.019	0.877 ± 0.022
greedy_margin_x2	0.006 ± 0.006	0.012 ± 0.007	0.004 ± 0.004	0.006 ± 0.010	0.010 ± 0.009	0.010 ± 0.010
greedy_margin_x4	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_margin_x6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_margin_x8	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000

Table 3: Endpoint-conditional coherence across samplers and graph families.

Sampling method	ST–ER $p = 0$ lazy 0.5	ST–ER avg. deg. 7 lazy 0.125	Bottleneck $m = 2, b = 1$	Bottleneck $m = 2, b = 8$	Bottleneck $m = 2, b = 64$	Bottleneck $m = 10$
bisection	0.947 ± 0.012	0.620 ± 0.008	0.755 ± 0.004	0.773 ± 0.007	0.680 ± 0.012	0.836 ± 0.008
bisection entropy	0.956 ± 0.013	0.745 ± 0.006	0.818 ± 0.009	0.822 ± 0.007	0.748 ± 0.003	0.973 ± 0.004
greedy_entropy	0.952 ± 0.011	0.927 ± 0.003	0.941 ± 0.041	0.968 ± 0.006	0.943 ± 0.001	0.973 ± 0.003
greedy_entropy_x2	0.611 ± 0.022	0.593 ± 0.007	0.555 ± 0.022	0.560 ± 0.009	0.537 ± 0.006	0.479 ± 0.008
greedy_entropy_x4	0.126 ± 0.013	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_x6	0.041 ± 0.003	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_x8	0.015 ± 0.003	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_exponential	0.029 ± 0.004	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
random	0.938 ± 0.015	0.712 ± 0.005	0.788 ± 0.013	0.800 ± 0.006	0.701 ± 0.005	0.850 ± 0.007
random_x2	0.736 ± 0.009	0.284 ± 0.007	0.327 ± 0.008	0.326 ± 0.004	0.289 ± 0.004	0.424 ± 0.006
random_x4	0.424 ± 0.013	0.039 ± 0.001	0.048 ± 0.006	0.049 ± 0.002	0.042 ± 0.003	0.105 ± 0.006
random_x6	0.219 ± 0.011	0.003 ± 0.001	0.005 ± 0.001	0.004 ± 0.001	0.004 ± 0.001	0.020 ± 0.001
random_x8	0.105 ± 0.003	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.004 ± 0.001
random_exponential	0.399 ± 0.012	0.016 ± 0.002	0.012 ± 0.002	0.014 ± 0.002	0.011 ± 0.002	0.028 ± 0.002
greedy_confidence	0.937 ± 0.008	0.917 ± 0.001	0.930 ± 0.035	0.951 ± 0.003	0.919 ± 0.004	0.914 ± 0.003
greedy_confidence_x2	0.662 ± 0.017	0.111 ± 0.027	0.101 ± 0.023	0.099 ± 0.028	0.065 ± 0.002	0.161 ± 0.042
greedy_confidence_x4	0.168 ± 0.011	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_confidence_x6	0.063 ± 0.008	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_confidence_x8	0.024 ± 0.006	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_margin	0.928 ± 0.010	0.882 ± 0.007	0.900 ± 0.036	0.922 ± 0.001	0.858 ± 0.002	0.868 ± 0.003
greedy_margin_x2	0.523 ± 0.019	0.500 ± 0.020	0.516 ± 0.025	0.460 ± 0.054	0.466 ± 0.020	0.483 ± 0.023
greedy_margin_x4	0.198 ± 0.011	0.016 ± 0.004	0.011 ± 0.002	0.003 ± 0.002	0.005 ± 0.002	0.055 ± 0.027
greedy_margin_x6	0.089 ± 0.011	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.007 ± 0.003
greedy_margin_x8	0.040 ± 0.009	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000

Table 4: Bottleneck-conditional coherence across samplers and bottleneck graph families.

Sampling method	Bottleneck $m = 2, b = 1$	Bottleneck $m = 2, b = 8$	Bottleneck $m = 2, b = 64$	Bottleneck $m = 10$
bisection	0.253 ± 0.028	0.572 ± 0.010	0.587 ± 0.008	0.182 ± 0.052
bisection entropy	0.285 ± 0.037	0.703 ± 0.010	0.696 ± 0.004	0.284 ± 0.091
greedy_entropy	0.172 ± 0.025	0.897 ± 0.002	0.928 ± 0.004	0.288 ± 0.090
greedy_entropy_x2	0.084 ± 0.010	0.478 ± 0.007	0.541 ± 0.007	0.113 ± 0.028
greedy_entropy_x4	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_x6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_x8	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_entropy_exponential	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
random	0.224 ± 0.025	0.668 ± 0.007	0.656 ± 0.005	0.199 ± 0.060
random_x2	0.083 ± 0.010	0.248 ± 0.005	0.254 ± 0.005	0.091 ± 0.028
random_x4	0.009 ± 0.001	0.029 ± 0.001	0.031 ± 0.002	0.019 ± 0.004
random_x6	0.001 ± 0.001	0.002 ± 0.000	0.002 ± 0.001	0.002 ± 0.001
random_x8	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
random_exponential	0.004 ± 0.001	0.012 ± 0.000	0.010 ± 0.001	0.009 ± 0.003
greedy_confidence	0.160 ± 0.024	0.903 ± 0.004	0.911 ± 0.002	0.287 ± 0.090
greedy_confidence_x2	0.015 ± 0.003	0.090 ± 0.008	0.082 ± 0.004	0.028 ± 0.010
greedy_confidence_x4	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.001 ± 0.001
greedy_confidence_x6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_confidence_x8	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_margin	0.170 ± 0.018	0.845 ± 0.005	0.846 ± 0.010	0.270 ± 0.090
greedy_margin_x2	0.077 ± 0.007	0.272 ± 0.010	0.407 ± 0.004	0.093 ± 0.025
greedy_margin_x4	0.002 ± 0.001	0.003 ± 0.000	0.004 ± 0.001	0.008 ± 0.003
greedy_margin_x6	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
greedy_margin_x8	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000

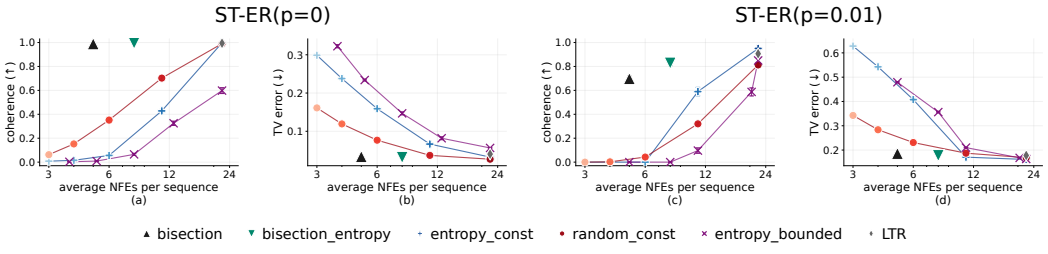

Figure 12: Coherence and TV error for endpoint-conditioned random-walk generation on ST–ER graphs. Higher coherence and lower TV error indicate better performance.

J.3 ORDER K RANDOM WALKS

Let $G = (V, E)$ be the fixed undirected graph. For a vertex $u \in V$, $N(u) = \{v \in V : (u, v) \in E\}$ denotes its set of neighbors, and $|N(u)|$ is its degree. The graph used below is connected and has no isolated vertices. Here K is the maximum history lag, and `skip_recent` is the number of most recent lags excluded from the distant-history rule. For a trajectory of length L, $X_\ell \in V$ is the random vertex at position ℓ, and $X_{1:\ell} = (X_1, \ldots, X_\ell)$ is its history. We draw X_1 uniformly from V and generate the remaining vertices using the conditional transition law P below. We write $\mathbf{1}\{\cdot\}$ for an indicator and $|A|$ for the number of elements in a finite set A. Given a history $X_{1:\ell}$, define the set of eligible lags $A_\ell = \{j : \texttt{skip_recent} + 1 \le j \le K,\ j \le \ell - 1\}$. If $A_\ell \neq \emptyset$, the transition kernel is

$$P(X_{\ell+1} = v \mid X_{1:\ell}) = \frac{1}{|A_\ell|} \sum_{j \in A_\ell} \frac{\mathbf{1}\{v \in N(X_{\ell+1-j})\}}{|N(X_{\ell+1-j})|}.$$

If $A_\ell = \emptyset$, we use the one-step non-lazy random-walk kernel $P(X_{\ell+1} = v \mid X_{1:\ell}) = \mathbf{1}\{v \in N(X_\ell)\}/|N(X_\ell)|$. In our main experiment, $K = 4$ and $\texttt{skip_recent} = 2$, so once enough history is available, $A_\ell = \{3, 4\}$. Thus $X_{\ell+1}$ is sampled from a neighbor of either $X_{\ell-2}$ or $X_{\ell-3}$, while the two most recent states are ignored by the data-generating rule. The initialization uses $A_\ell = \emptyset$ for $\ell \le 3$, $A_4 = \{3\}$, and $A_\ell = \{3, 4\}$ for $\ell \ge 5$.

For evaluation, $\widetilde{X}_{1:L} = (\widetilde{X}_1, \ldots, \widetilde{X}_L)$ denotes a sequence produced by the learned sampler after removing padding and other special tokens; $\widetilde{X}_\ell$ is its token at position ℓ. Here L is the resulting sequence length, which may vary across generated samples. We use the same coherence metric as above to check support validity, specialized to the nonlocal transition rule. A generated transition is valid if either $A_\ell = \emptyset$ and $\widetilde{X}_{\ell+1} \in N(\widetilde{X}_\ell)$, or $A_\ell \neq \emptyset$ and $\widetilde{X}_{\ell+1} \in N(\widetilde{X}_{\ell+1-j})$ for some $j \in A_\ell$. Writing $\phi_\ell(\widetilde{X}_{1:L})$ for this indicator, we report $\mathrm{coh}_{\text{dist-}K}(\widetilde{X}_{1:L}) = \prod_{\ell=1}^{L-1} \phi_\ell(\widetilde{X}_{1:L})$, averaged over generated samples. Empty unmasked sequences or sequences containing tokens outside V receive score zero. A repeated token is valid only when it satisfies the same adjacency condition, since this experiment uses no lazy transitions.

We use one ST-ER(p) graph with $|V| = 500$ and $p = 0.01$. The $50{,}000$ training trajectories have lengths drawn uniformly from $\{2, \ldots, 24\}$ and are right-padded to 24 positions, without beginning- or end-of-sequence tokens. We use $1{,}000$ held-out trajectories for validation. The model architecture and optimizer follow the graph-walk setup above; for this experiment, we train for $15{,}000$ steps with batch size 256 and select the checkpoint minimizing validation negative log-likelihood. The graph and transition rule are not provided to the model.

We generate from 24 masked positions and remove padding and other special tokens before scoring the unmasked vertex sequence. We compare divide-half bisection and entropy-guided bisection at separator sizes $1, \ldots, 6$, with entropy pivots restricted to the central half of each interval, against one-token random and entropy baselines. Each setting uses four independently seeded batches of 256 samples from the same fitted model. Error bars report the sample SD of the four batch coherence means; they measure generation variability for this graph and model.

At separator size 4, divide-half and entropy-guided bisection have mean coherence 0.659 and 0.827, using 13 and 15.19 model calls per sequence, respectively. Random one-token sampling has coherence 0.714 and uses 24 calls. Entropy one-token sampling has coherence 0.896 (SD 0.021) and also uses 24 calls. Mean unmasked sequence lengths range from 6.63 to 7.16 for divide-half bisection and from 4.52 to 5.21 for entropy-guided bisection; they are 6.62 and 6.21 for the random and entropy one-token baselines. These are support-validity comparisons under each sampler's output-length distribution, rather than comparisons at a fixed unmasked sequence length.