ANCHOR AND FOLLOWER: NARROWING VALID ANSWERS FOR PARALLEL DECODING IN DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion large language models (dLLMs) can accelerate decoding by committing tokens at multiple masked positions in parallel. However, when multiple valid answers remain compatible with the current context, parallel commitment can mix tokens from different answers and produce an output consistent with none of them. Sequential decoding reduces this risk by refining between commitments, suggesting a view of denoising as progressively narrowing the set of compatible valid answers. To study how predictions over valid answers change during denoising, we introduce *valid answer uncertainty* N_t. Across a controlled graph task and three ParallelBench tasks, we find that commitments differ sharply in their effect on N_t. Commitments that distinguish among the remaining valid answers can substantially reduce it, while shared commitments change it little. Intermediate refinement tends to reduce it further. Building on these findings, we propose **Anchor and Follower (AnF)**, a training-free decoding method that selects an anchor whose commitment can guide subsequent refinement toward lower answer uncertainty. AnF then commits the anchor, refines the remaining predictions, and commits high-confidence followers in parallel, combining sequential refinement with parallel efficiency. With LLaDA-8B-Instruct, AnF achieves 16.9% higher accuracy with 31.0% fewer NFEs than DAPD on HumanEval and the highest accuracy among compared methods on HumanEval, MBPP, and IFEval.

1 INTRODUCTION

Diffusion large language models (dLLMs) have emerged as a promising approach to text generation. At each denoising step, dLLMs simultaneously predict token distributions across masked positions conditioned on the tokens already revealed at that step. This ability to predict multiple positions at once enables parallel decoding, offering the potential to accelerate generation (Nie et al., 2026; Ye et al.; Wu et al., 2026a).

Even after some tokens are revealed, the remaining sequence may still admit multiple plausible answers, so predictions at different masked positions can reflect different alternatives (Mündler et al., 2026; Suresh et al., 2026; Bansal & Sanghavi, 2025). Sequential decoding commits one token at a time and refines the remaining predictions after each commitment, allowing later predictions to adapt to the earlier choice (Sahoo et al., 2024; Nie et al., 2026). Parallel decoding instead commits multiple tokens from the same prediction state before refinement, so predictions reflecting different alternatives can be combined into an inconsistent output (Wu et al., 2026a).

Prior work addresses inconsistency in parallel decoding through predictive uncertainty or dependencies between token predictions. Some methods avoid committing strongly dependent tokens together or learn token dependencies for joint generation (Wu et al., 2026a; Ben-Hamu et al., 2026; Azangulov et al., 2026; Kim et al., 2026a; Ringel et al., 2026; Bansal & Sanghavi, 2025; Hersche et al., 2026; Zhang et al., 2026), while others use token commitments to make the remaining predictions more confident and enable subsequent parallel commitment (Luo et al., 2026; Fu et al., 2025; Xu et al., 2025; Cao et al., 2026; Huang et al., 2026). However, prior work provides limited analysis of how a token commitment changes which answer alternatives remain possible and how this change affects the remaining predictions.

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.

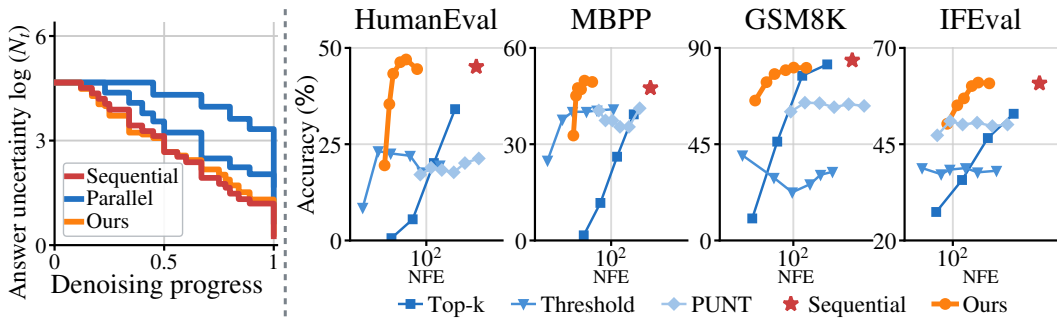

Figure 1: **Answer uncertainty and accuracy–NFE trade-offs.** Left: Mean valid answer uncertainty $\log N_t$ over denoising progress on the Graph task under sequential and parallel decoding. Right: Accuracy versus NFE for AnF and baselines on four benchmarks.

Figure 2 illustrates the answer-level process behind this problem. We refer to complete outputs satisfying the validity criterion as *valid answers*. When several valid answers remain compatible, the top-1 tokens at different masked positions can correspond to different answers. Committing these tokens in parallel can therefore mix tokens from different valid answers and produce a combination that is not consistent with any one of them (Figure 2, Left). If one token is committed first, however, valid answers incompatible with that choice are ruled out. Refining the remaining predictions under the updated context can then shift their top choices toward the answers that remain compatible, bringing several positions into agreement (Figure 2, Right). We view this process as progressively narrowing the set of compatible valid answers during denoising. This raises a central question: what happens to this narrowing process under parallel decoding?

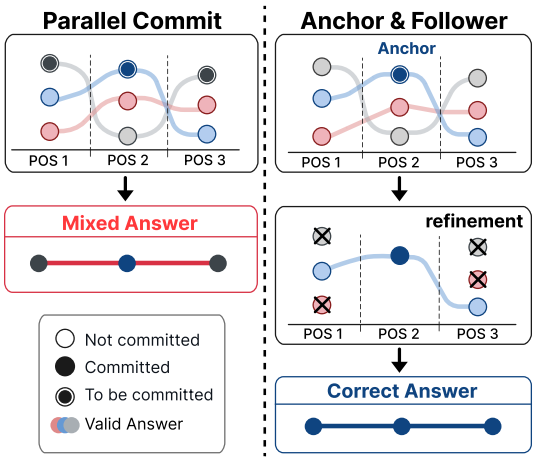

Figure 2: **Parallel commitment versus anchor-conditioned refinement.** Left: Parallel commitment can mix tokens from different valid answers. Right: Committing an anchor narrows the set of compatible valid answers, and anchor-conditioned refinement aligns the remaining predictions.

To examine this quantitatively, we introduce *valid answer uncertainty* N_t, which captures how broadly answer contribution mass is distributed across valid answers and how much such mass is available. We evaluate N_t on a controlled graph pathfinding task and three structured tasks from ParallelBench (Kang et al., 2026), where multiple valid answers can be explicitly identified. As shown in Figure 1 (Left), sequential decoding progressively reduces N_t, whereas parallel decoding retains substantially higher N_t throughout denoising. Further analysis shows that this effect depends on the role of each commitment: some commitments substantially reduce N_t, while others change it little, and intermediate refinement tends to reduce it further.

Motivated by these findings, we propose *Anchor and Follower* (AnF), which selects and commits an anchor, refines the remaining predictions under that choice, and then commits high-confidence followers in parallel. This combines sequential-style adaptation with the efficiency of parallel commitment. Because valid answer uncertainty is unavailable at inference time without enumerating valid answers, AnF uses current token confidence to form anchor candidates and entropy reduction after anchor-conditioned refinement to select among them, requiring no additional training. Across four generation benchmarks, AnF improves the accuracy–NFE trade-off; on HumanEval, LLaDA-8B-Instruct reaches 47.0% accuracy at 51.7 mean NFE, compared with 40.2% at 74.9 NFE for DAPD.

2 Related Work

Diffusion large language models. Diffusion large language models (dLLMs) have emerged as an alternative to autoregressive language models, offering competitive language modeling capabilities and the potential for parallel generation (Sahoo et al., 2024; Nie et al., 2026; Ye et al.). They can accelerate generation by committing multiple tokens in a single forward pass. To enable parallel decoding, existing methods use confidence-based selection with fixed token budgets or adaptive commitment rules based on confidence thresholds and entropy bounds (Nie et al., 2026; Wu et al., 2026a; Ben-Hamu et al., 2026). Beyond parallel generation, dLLMs can generate diverse responses to the same input, offering alternatives that differ in content rather than merely wording (Wu et al., 2026b; Koh et al., 2025). This capability enables the exploration of multiple candidate solutions, which can be evaluated and selected to improve reasoning performance (Lamont et al., 2026).

Consistency challenges in parallel decoding. In dLLMs, tokens at different masked positions are sampled independently given the current context, which can lead to inconsistent outputs when multiple tokens are committed together (Wu et al., 2026a; Ben-Hamu et al., 2026; Azangulov et al., 2026). To address this issue, prior work selects tokens for parallel commitment by estimating token dependencies, testing conditional independence, or controlling the approximation error of parallel sampling (Kim et al., 2026a; Azangulov et al., 2026; Ben-Hamu et al., 2026; Luo et al., 2026). Other work prioritizes structurally important tokens to guide subsequent generation and improve output consistency (Xue et al., 2026). Another line of work seeks to accelerate decoding by exploring which commitments make the remaining predictions more confident, enabling more tokens to be committed in subsequent steps (Fu et al., 2025; Xu et al., 2025). These approaches guide token commitments using dependencies between positions, structural information, or predictive uncertainty in the current or conditionally updated predictions.

Output consistency under multiple plausible answers. These token dependencies can also be understood in terms of multiple plausible answers, where independent token choices may mix incompatible alternatives. In machine translation, independently generated tokens can mix different valid translations into an inconsistent output, motivating approaches based on knowledge distillation and latent variables (Zhou et al., 2019; Bao et al., 2022; Gu et al., 2017). A similar issue arises in compositional diffusion planning, where combining incompatible modes from local plan distributions can produce globally incoherent plans (Mishra et al., 2026; Du et al., 2023). These studies suggest that coordinating local predictions into a consistent output can become more difficult as the number of plausible alternatives with conflicting choices increases.

3 Preliminaries

3.1 Diffusion Large Language Models

Diffusion large language models (dLLMs) generate text by iteratively predicting token distributions at masked positions in parallel, conditioned on the current partially masked sequence (Nie et al., 2026; Ye et al.).

Let $c_t = (c_{t,1}, \ldots, c_{t,L})$ denote the partially masked sequence at denoising step t, and let

$$M_t = \{j \in \{1, \ldots, L\} : c_{t,j} = [\text{MASK}]\}$$

denote the set of masked positions. At each step, the model predicts a token distribution $p_{\theta,j}(\cdot \mid c_t)$ for each masked position $j \in M_t$.

For a complete sequence $o = (o_1, \ldots, o_L)$, the current predictive distribution is

$$p_\theta(o \mid c_t) = \prod_{j=1}^{L} p_{\theta,j}(o_j \mid c_t), \tag{1}$$

where the distribution at each committed position is deterministic.

A decoding strategy selects one or more masked positions and commits a token at each selected position to form the next state c_{t+1}.

3.2 Denoising as Progressive Answer Narrowing

A denoising trajectory is the sequence of partially masked states $c_0, c_1, \ldots, c_T$ across denoising steps. The initial state c_0 may admit multiple answers that differ in content or expression (Gupta et al., 2025; Zhang et al., 2025; Qiu et al., 2019; Wang et al., 2026). At an intermediate denoising step, multiple such answers may remain compatible with all unmasked tokens in c_t (Mündler et al., 2026; Suresh et al., 2026).

We define the set of complete sequences compatible with c_t as

$$\mathcal{C}(c_t) = \left\{ o \in \mathcal{V}^L : o_j = c_{t,j} \text{ for all } j \notin M_t \right\}, \tag{2}$$

where $\mathcal{V}$ denotes the token vocabulary. Compatibility requires only agreement with the unmasked tokens in the current state.

We view denoising as progressively narrowing this set through token commitments. Committing additional tokens updates the sequence from c_t to c_{t+1}, so the compatible set can only stay the same or shrink:

$$\mathcal{C}(c_{t+1}) \subseteq \mathcal{C}(c_t). \tag{3}$$

This assumes that previously committed tokens remain fixed. Each new commitment rules out sequences that assign a different token to that position, progressively narrowing the compatible set.

3.3 Valid Answer Uncertainty

In parallel decoding, multiple positions are committed from the same predictive state. When position-wise predictions reflect different answer alternatives, committing them together can produce a sequence that is not consistent with any single answer. To characterize this predictive state, we introduce *valid answer uncertainty*, which captures both how much predictive mass can be coherently assigned to valid answers and how broadly that mass is distributed across them.

Valid answers. We first identify the valid answers to which the model's predictive mass should be assigned. We use $\mathrm{Valid}(o) \in \{0, 1\}$ to indicate whether a complete sequence o satisfies an externally specified validity criterion. This criterion may reflect linguistic plausibility, semantic coherence, factual or logical correctness, or satisfaction of specific constraints. Let $\mathcal{Y} = \{o \in \mathcal{C}(c_0) : \mathrm{Valid}(o) = 1\} = \{y^{(1)}, \ldots, y^{(R)}\}$ be the fixed set of R valid answers.

Local units. A complete sequence may be invalid as a whole while containing valid parts. To distinguish partial validity from inconsistent combinations of valid parts, we decompose each sequence into local units. For each task, let $\mathcal{D}(o) = (u_1^o, \ldots, u_m^o)$ denote the local unit decomposition of o. Collect the distinct units appearing in valid answers as $\mathcal{U} = \bigcup_{i=1}^{R} \mathcal{D}(y^{(i)}) = \{u_1, \ldots, u_Q\}$. For each valid answer and unit, define

$$v_{ir} = \mathbf{1}[u_r \in \mathcal{D}(y^{(i)})], \qquad \mathbf{V} = (v_{ir}) \in \{0, 1\}^{R \times Q}. \tag{4}$$

Thus, $\mathbf{V}$ records which valid answers contain each unit in $\mathcal{U}$.

Answer contribution mass. At step t, a complete assignment $o = (o_1, \ldots, o_L) \in \mathcal{V}^L$ drawn from $p_\theta(\cdot \mid c_t)$ may be invalid or contain mutually inconsistent local units.

Let $\mathbf{x}(o) \in \mathbb{N}_0^Q$ have entries $x_r(o) = \sum_{k=1}^{m} \mathbf{1}[u_k^o = u_r]$, counting the occurrences of each individually valid local unit in o. Thus, $\|\mathbf{x}(o)\|_1$ counts the individually valid local-unit occurrences in o, and $(\mathbf{V}\mathbf{x}(o))_i$ counts how many of them are contained in $y^{(i)}$. We define the answer contribution mass assigned to valid answers by

$$\mathbf{s}_t = (s_{t,1}, \ldots, s_{t,R}) = \mathbb{E}_{o \sim p_\theta(\cdot \mid c_t)} \left[\frac{\|\mathbf{x}(o)\|_1}{m} \cdot \frac{\mathbf{b}(o)}{\|\mathbf{b}(o)\|_1} \right]. \tag{5}$$

Here, $b_i(o) = \mathbf{1}[(\mathbf{V}\mathbf{x}(o))_i = \|\mathbf{x}(o)\|_1]$ indicates whether $y^{(i)}$ contains every individually valid unit in o. The first factor weights o by its normalized valid unit count and the second distributes that weight equally among valid answers that contain every individually valid unit in o. We define $\mathbf{b}(o)/\|\mathbf{b}(o)\|_1 = \mathbf{0}$ when $\mathbf{b}(o) = \mathbf{0}$. Assignments with conflicting valid units or no valid units contribute nothing to $\mathbf{s}_t$.

Valid answer uncertainty. We define *valid answer uncertainty* from these answer contribution masses as

$$N_t = \underbrace{\frac{\|\mathbf{s}_t\|_1^2}{\|\mathbf{s}_t\|_2^2}}_{\text{effective answer spread}} \times \underbrace{\frac{1}{\|\mathbf{s}_t\|_1^2}}_{\text{low mass penalty}} = \frac{1}{\|\mathbf{s}_t\|_2^2}. \tag{6}$$

The first factor measures how broadly answer contribution mass is distributed across valid answers. The second factor penalizes low total answer contribution mass. Thus, N_t becomes large when the mass is broadly distributed, its total is low, or both. When $\mathbf{s}_t = \mathbf{0}$, N_t is undefined, and such states are excluded from all computations involving N_t. Details of the metric and its task-specific computation are provided in Appendix A.

4 Answer Uncertainty in Diffusion Denoising

In this section, we examine when committing multiple tokens at once may combine tokens from different valid answers and produce an invalid sequence. We use answer uncertainty N_t to characterize how broadly answer contribution mass is distributed across valid answers and how much such mass is available, and examine how N_t changes after refining the remaining masked positions.

We compare two decoding modes: *sequential decoding*, which refines the remaining predictions after each token commitment, and *parallel decoding*, which commits multiple tokens before refinement. We conduct this analysis on a controlled graph pathfinding task (Graph) and three structured tasks from ParallelBench (Kang et al., 2026) (Replace Random, Shuffle, and Latin Square) using LLaDA-8B-Instruct (Nie et al., 2026). Each task admits multiple valid answers that can be explicitly identified, allowing us to evaluate N_t at intermediate denoising states. Detailed task descriptions and experimental settings are provided in Appendix B.

4.1 Valid Answer Uncertainty during Denoising

Commitments have uneven effects on answer uncertainty. Figure 1 (Left) shows that mean $\log N_t$ decreases during denoising under top-1, top-2, and top-4 decoding (Ghazvininejad et al., 2019; Nie et al., 2026) on Graph. This trend suggests that denoising progressively reduces answer uncertainty, reducing the risk of combining tokens from different answers in a parallel commitment. However, Figure 3 (Left) shows that some commitments barely change N_t, whereas others sharply reduce it. Figure 3 further shows the distribution of changes in $\log N_t$ across top-1 commitments. These results show that not all commitments reduce answer uncertainty by the same amount.

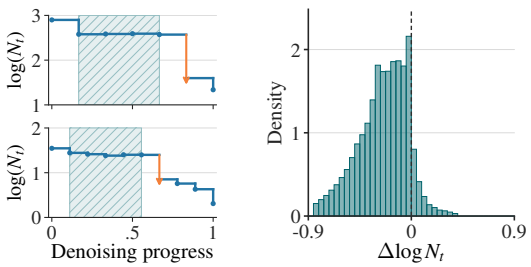

Figure 3: **Uneven changes in answer uncertainty on Graph.** Left: Two example top-1 trajectories of $\log N_t$. Right: Distribution of per-commitment changes $\Delta \log N_t$. The dashed line marks zero. Outliers are excluded for visualization.

4.2 Commitment Effects and Parallel Decoding

Commitment roles explain different changes in answer uncertainty. Section 4.1 shows that different commitments can have very different effects on N_t. We therefore examine what distinguishes commitments that substantially reduce answer uncertainty from those that do not. For this analysis, we obtain contexts c_t from top-1 decoding trajectories at unmasking ratios $\rho \in \{0, 0.25, 0.5\}$. We then characterize each commitment by how its committed token relates to the valid answers remaining compatible with the current state. Within each fixed context c_t, we separately commit the argmax token at each masked position j, refine the remaining predictions, and measure the resulting change in answer uncertainty as $\Delta N_j = N_{t+1}^{(j)} - N_t$.

Figure 4 (Left) illustrates three such roles. At position 2, committing B retains only the valid paths that follow that branch, whereas E at position 4 is shared by all remaining valid paths. In contrast, committing E at position 6 is incompatible with all remaining valid paths. We refer to these as *branch*, *shared*, and *incompatible* commitments, respectively. More generally, they agree with some, all, or none of the remaining valid answers.

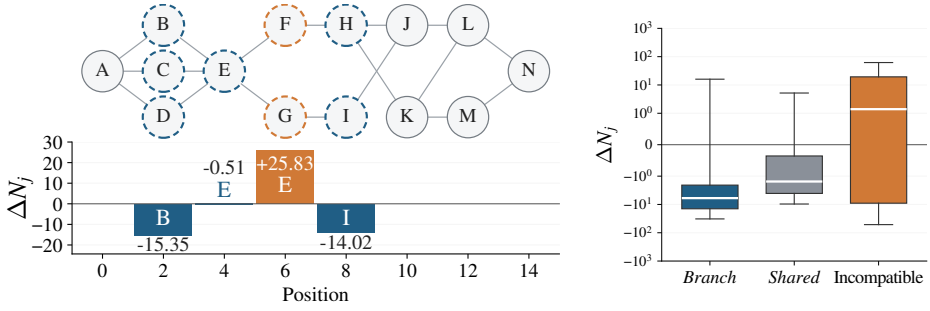

Figure 4: **Single-position commitment effects on valid answer uncertainty.** Left: Changes ΔN_j from committing each masked position separately in one Graph example. Right: Distribution of ΔN_j across Graph states at $\rho = 0.25$, grouped by commitment role (*Branch*, *Shared*, *Incompatible*).

Figure 4 (Left) shows how these roles translate into different changes in answer uncertainty. Committing B at position 2, followed by refinement, rules out the alternatives on the other branches and substantially reduces N_t. In contrast, committing E at position 4, which is shared across the remaining valid paths, changes N_t only slightly. The same token E at position 6 is incompatible with every remaining valid path and instead increases N_t. Figure 4 (Right) shows that these role-dependent effects extend across Graph states. Together, these results show that N_t reflects whether a commitment narrows, preserves, or violates the set of remaining valid answers.

Intermediate refinement can change subsequent token choices. Since commitment effects depend on their relation to the remaining valid answers, we examine how refinement between commitments affects subsequent decoding.

For each intermediate context c_t, we fix the same four masked positions for both modes. Parallel decoding commits all four positions from the same predictive state and refines once, yielding N_{t+1}^{par}. Sequential decoding commits the positions one at a time with refinement after each commitment, yielding N_{t+4}^{seq}. The positions are fixed across the two modes, while later committed tokens may differ after refinement.

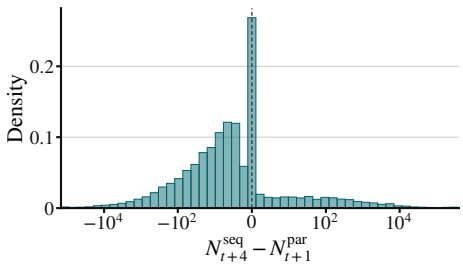

Figure 5: **Sequential vs. parallel commitment on Graph.** Distribution of $N_{t+4}^{\text{seq}} - N_{t+1}^{\text{par}}$. Negative values indicate lower valid answer uncertainty with intermediate refinement.

Figure 5 shows that sequential decoding more often yields lower answer uncertainty, although many comparisons are close to zero. This indicates that refinement between commitments can alter the subsequent decoding trajectory and the resulting answer uncertainty.

5 Anchor and Follower

Parallel decoding improves efficiency by committing multiple tokens from the same predictive state, but prevents them from incorporating information from one another. When multiple valid answers remain compatible, an earlier commitment can narrow the compatible set and, through refinement, alter subsequent predictions. Motivated by this, we introduce *Anchor and Follower* (AnF), which first selects and commits an anchor, refines the remaining predictions, and then commits high-confidence followers in parallel. This allows later commitments to adapt to the anchor while retaining parallel efficiency.

5.1 Selecting Anchors for Answer Narrowing

A useful anchor should be sufficiently confident under the current prediction and should reduce predictive ambiguity at the remaining positions after refinement. We therefore use current token confidence to form a small set of anchor candidates and entropy reduction after anchor-conditioned refinement to select among them.

We first identify anchor candidates from the current predictions. Let $W_t \subseteq M_t$ denote the masked positions considered for anchor evaluation. Given a target confidence γ and candidate budget K, we construct the anchor candidate set A_t from the K positions in W_t whose top-1 probabilities are closest to γ:

$$A_t = \mathrm{TopK}_{j \in W_t} \left(- \left| \max_{o_j} p_{\theta,j}(o_j \mid c_t) - \gamma \right| \right). \tag{7}$$

Positions whose top-1 prediction is the padding token are excluded from anchor candidates, since padding predictions can bias the confidence scores used for anchor selection (Kim et al., 2026b; Yang et al., 2025).

Current token confidence alone does not indicate how a commitment will affect the remaining predictions. Ideally, this effect could be evaluated through valid answer uncertainty, but N_t is not available during decoding. We therefore use the reduction in predictive entropy after anchor-conditioned refinement as a practical selection signal. This choice draws on prior work connecting reduced output diversity with lower entropy token-level predictions (Gu et al., 2017; Ding et al., 2020). For each anchor candidate $a \in A_t$, we commit its top-1 token to obtain context $c_t^{(a)}$ and refine the remaining predictions. We define the anchor score as the normalized entropy reduction over the remaining evaluated positions:

$$G_t(a) = \frac{\sum_{j \in W_t \setminus \{a\}} \left[H(p_{\theta,j}(\cdot \mid c_t)) - H\left(p_{\theta,j}(\cdot \mid c_t^{(a)})\right) \right]}{\sum_{j \in W_t \setminus \{a\}} H(p_{\theta,j}(\cdot \mid c_t))}. \tag{8}$$

The selected anchor is

$$a_t^* = \arg\max_{a \in A_t} G_t(a). \tag{9}$$

This selection favors anchors that most strongly reduce predictive ambiguity at the remaining evaluated positions, providing refined predictions for subsequent follower selection.

5.2 Follower Selection for Parallel Commitment

After selecting the anchor, the next design question is which remaining positions can be committed in parallel. The anchor commitment can narrow the set of compatible valid answers, while anchor-conditioned refinement updates the remaining predictions under that choice. Tokens shared across the remaining valid answers can receive higher marginal probabilities because those answers agree on the corresponding choices (Gu et al., 2017; Ding et al., 2020). Such choices are natural candidates for parallel commitment because committing them does not require selecting among the remaining valid answers.

Because this sharedness is not directly available during decoding, we use confidence in the anchor-conditioned predictions as a practical selection criterion. Given a follower threshold τ, we define the follower set as

$$F_t = \left\{ j \in W_t \setminus \{a_t^*\} : \max_{o_j} p_{\theta,j}\left(o_j \mid c_t^{(a_t^*)}\right) \geq \tau \right\}. \tag{10}$$

The selected follower positions are committed in parallel using their top-1 tokens under the refined predictions. Thus, AnF first uses the anchor to adapt the remaining predictions and then recovers parallelism through follower commitment.

6 Experiments

We evaluate accuracy–efficiency trade-offs for AnF on four generation benchmarks, then examine how anchor-conditioned refinement changes follower availability and whether followers align with

the valid answers remaining after the anchor. We use Graph, Replace Random, Shuffle, and Latin Square as mechanism probes, allowing follower predictions to be classified by their relation to the remaining valid answers.

Experimental Setup We evaluate AnF on GSM8K (Cobbe et al., 2021) for mathematical reasoning, HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for code generation, and IFEval (Zhou et al., 2023) for instruction following. We use three pretrained diffusion language models: LLaDA-8B-Instruct (Nie et al., 2026), LLaDA-1.5 (Zhu et al., 2026), and Dream-Instruct-7B (Ye et al.). Generation length is 512 tokens for GSM8K, HumanEval, and MBPP, and 1024 for IFEval. We compare AnF with top-k decoding (Nie et al., 2026; Sahoo et al., 2024), threshold decoding, Fast-dLLM (Wu et al., 2026a), PUNT (Azangulov et al., 2026), DAPD (Kim et al., 2026a), and ETE (Fu et al., 2025). Detailed settings and additional ablations are provided in Appendix E.

6.1 Accuracy–Efficiency Trade-offs on Generation Benchmarks

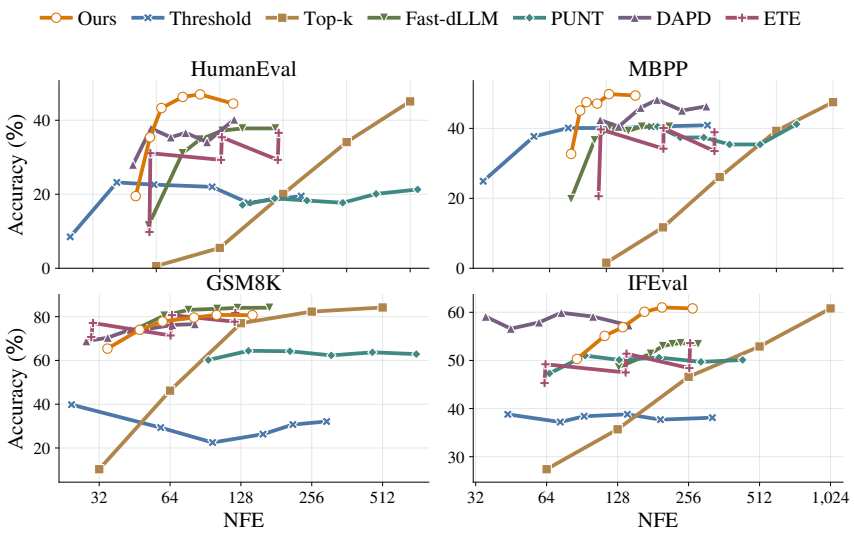

Figure 6: **Accuracy–NFE trade-offs on four generation benchmarks.** LLaDA-8B-Instruct results on HumanEval, MBPP, GSM8K, and IFEval. The x-axis shows mean physical model forward calls (NFE), and the y-axis shows accuracy.

Performance on generation benchmarks. Figure 6 shows the accuracy–NFE results across the four benchmarks. AnF attains the highest observed accuracy on HumanEval, MBPP, and IFEval. On HumanEval, AnF reaches 47.0% at 51.7 NFE, compared with 40.2% at 74.9 NFE for DAPD. Table 1 reports GPU-forward throughput at each method's highest-accuracy evaluated setting. Including candidate-anchor evaluations, AnF reaches 362.2 tokens/s, compared with 231.3 tokens/s for DAPD (1.57×). AnF achieves both the highest accuracy and GPU-forward throughput among the timed methods on HumanEval.

Table 1: **GPU-forward throughput on HumanEval.** Settings are in Appendix E.

Method	Acc. (%)	Tokens/s
Sequential	45.1	33.6
Fast-dLLM	37.8	222.2
PUNT	21.3	30.5
DAPD	40.2	231.3
ETE	36.6	125.2
AnF	**47.0**	362.2

6.2 Effect of Anchor-Conditioned Refinement

We examine whether anchor-conditioned refinement makes the remaining predictions more suitable for parallel follower commitment. We use AnF with target confidence $\gamma = 1$, candidate budget $K = 3$, and follower threshold $\tau = 0.9$ on all four generation benchmarks. Figure 7 (Left) compares the number of positions passing the follower threshold before and after conditioning on the selected anchor, and reports how often the top-1 token changes after conditioning at the positions selected as followers.

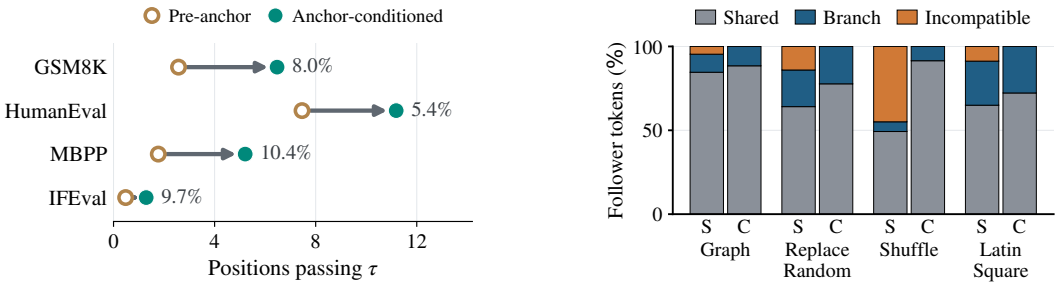

Figure 7: **Effects of anchor conditioning.** Left: Number of positions with top-1 confidence above $\tau = 0.9$ before and after anchor conditioning; percentages indicate the fraction of post-conditioning followers whose top-1 token differs from the pre-anchor prediction. Right: Roles of pre-anchor (S) and anchor-conditioned (C) tokens at the same follower positions.

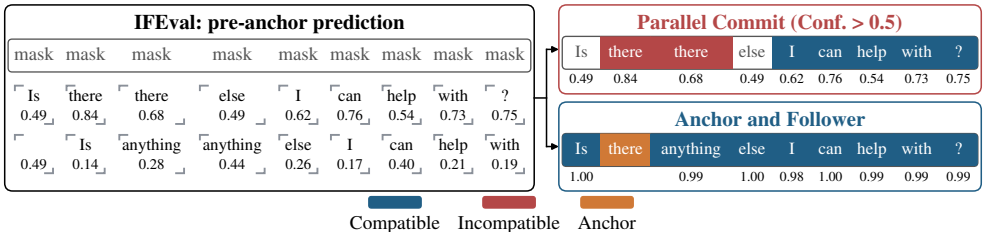

Figure 8: **Example of Parallel Commit vs. Anchor and Follower on IFEval.** Parallel Commit commits a high-confidence incompatible "there," whereas anchor-conditioned refinement changes it to "anything," yielding a coherent continuation.

Anchor conditioning increases the mean number of positions passing the follower threshold across all four benchmarks, with GSM8K increasing from 2.57 to 6.47, and the top-1 token changes at 5–11% of the selected follower positions across the benchmarks. These results show that anchor-conditioned refinement allows more positions to pass the follower threshold and can also change the token selected for parallel commitment. Figure 8 illustrates this change in an IFEval example.

6.3 Follower Alignment after Anchor Conditioning

We next examine whether anchor-conditioned followers are shared across the remaining valid answers. On Graph and the three ParallelBench tasks, we classify matched pre-anchor and conditioned top-1 tokens as *Shared*, *Branch*, or *Incompatible*.

Figure 7 (Right) shows that the Shared fraction increases after anchor conditioning on all four structured tasks. The largest change occurs on Shuffle, where it rises from 49.3% to 91.5%. At the same follower positions, the pre-anchor predictions contain Incompatible tokens on every task, whereas none remain after conditioning. These results show that anchor conditioning shifts follower predictions toward choices shared by the valid answers remaining after the anchor. Additional comparisons of anchor selection and sensitivity to γ, K, and τ are provided in Appendix C.

7 Conclusion

We view denoising in dLLMs as progressively narrowing the set of compatible valid answers, and quantify this process with *valid answer uncertainty*, based on the spread and total mass of answer contributions. We find that commitments affect this uncertainty unevenly, while intermediate refinement can further reduce it. Motivated by these findings, we propose Anchor and Follower (AnF), a training-free method that uses token confidence and entropy reduction to select an anchor, refines the remaining predictions, and commits high-confidence followers in parallel. AnF improves accuracy–NFE trade-offs and attains the highest observed accuracy on HumanEval, MBPP, and IFEval among the compared methods on LLaDA-8B-Instruct.

AI USE STATEMENT

We used generative AI tools to search for relevant literature, generate research code, edit the manuscript, and provide feedback on experimental design. The authors reviewed the AI-assisted work and take responsibility for the final paper.

REFERENCES

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

Iskander Azangulov, Teodora Pandeva, Niranjani Prasad, Javier Zazo, and Sushrut Karmalkar. Parallel sampling from masked diffusion models via conditional independence testing. In *International Conference on Learning Representations*, volume 2026, pp. 140485–140521, 2026.

Parikshit Bansal and Sujay Sanghavi. Enabling approximate joint sampling in diffusion lms. *arXiv preprint arXiv:2509.22738*, 2025.

Yu Bao, Hao Zhou, Shujian Huang, Dongqi Wang, Lihua Qian, Xinyu Dai, Jiajun Chen, and Lei Li. latent-glat: Glancing at latent variables for parallel text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8398–8409, 2022.

Heli Ben-Hamu, Itai Gat, Daniel Severo, Niklas S Nolte, and Brian Karrer. Accelerated sampling from masked diffusion models via entropy bounded unmasking. *Advances in Neural Information Processing Systems*, 38:55981–56007, 2026.

Mingyu Cao, Alvaro HC Correia, Christos Louizos, Shiwei Liu, and Lu Yin. Search or accelerate: Confidence-switched position beam search for diffusion language models. *arXiv preprint arXiv:2602.10953*, 2026.

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.

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

Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao, and Zhaopeng Tu. Understanding and improving lexical choice in non-autoregressive translation. *arXiv preprint arXiv:2012.14583*, 2020.

Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International conference on machine learning*, pp. 8489–8510. PMLR, 2023.

Hengyu Fu, Baihe Huang, Virginia Adams, Charles Wang, Venkat Srinivasan, and Jiantao Jiao. From bits to rounds: Parallel decoding with exploration for diffusion language models. *arXiv preprint arXiv:2511.21103*, 2025.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 6112–6121, 2019.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Sonam Gupta, Yatin Nandwani, Asaf Yehudai, Dinesh Khandelwal, Dinesh Raghu, and Sachindra Joshi. Selective self-to-supervised fine-tuning for generalization in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 6255–6264, 2025.

Michael Hersche, Nicolas Menet, Ronan Tanios, and Abbas Rahimi. Locally coherent parallel decoding in diffusion language models. *arXiv preprint arXiv:2603.20216*, 2026.

Zheng Huang, Kiran Ramnath, Yueyan Chen, Aosong Feng, Sangmin Woo, Balasubramaniam Srinivasan, Zhichao Xu, Kang Zhou, Shuai Wang, Haibo Ding, et al. Diffusion language model inference with monte carlo tree search. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 3493–3512, 2026.

Wonjun Kang, Kevin Galim, Seunghyuk Oh, Minjae Lee, Yuchen Zeng, Shuibai Zhang, Coleman Hooper, Yuezhou Hu, Hyung Koo, Nam Ik Cho, et al. Parallelbench: Understanding the trade-offs of parallel decoding in diffusion llms. In *International Conference on Learning Representations*, volume 2026, pp. 89900–89938, 2026.

Bumjun Kim, Dongjae Jeon, Moongyu Jeon, and Albert No. Dapd: Dependency-aware parallel decoding via attention for diffusion llms. *arXiv preprint arXiv:2603.12996*, 2026a.

Bumjun Kim, Dongjae Jeon, Dueun Kim, Wonje Jeung, and Albert No. Rainbow padding: Mitigating early termination in instruction-tuned diffusion llms. In *International Conference on Learning Representations*, volume 2026, pp. 18977–19001, 2026b.

Hyukhun Koh, Minha Jhang, Dohyung Kim, Sangmook Lee, and Kyomin Jung. Conditional [mask] discrete diffusion language model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8910–8934, 2025.

Sean Lamont, Christian Walder, Paul Montague, Amir Dezfouli, and Michael Norrish. Free lunch for pass@ k? low cost diverse sampling for diffusion language models. *arXiv preprint arXiv:2603.04893*, 2026.

Lizhuo Luo, Zhuoran Shi, Jiajun Luo, Zhi Wang, Shen Ren, Wenya Wang, and Tianwei Zhang. Dawn: Dependency-aware fast inference for diffusion llms. *arXiv preprint arXiv:2602.06953*, 2026.

Utkarsh Mishra, David He, Yongxin Chen, and Danfei Xu. Compositional diffusion with guided search for long-horizon planning. In *International Conference on Learning Representations*, volume 2026, pp. 68308–68322, 2026.

Niels Mündler, Jasper Dekoninck, and Martin Vechev. Constrained decoding of diffusion llms with context-free grammars. In *International Conference on Learning Representations*, volume 2026, pp. 99362–99408, 2026.

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. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

Lisong Qiu, Juntao Li, Dongyan Zhao, Rui Yan, et al. Are training samples correlated? learning to generate dialogue responses with multiple references. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3826–3835, 2019.

Liran Ringel, Ameen Ali, and Yaniv Romano. Dependency-guided parallel decoding in discrete diffusion language models. *arXiv preprint arXiv:2604.02560*, 2026.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Tarun Suresh, Debangshu Banerjee, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. Dingo: Constrained inference for diffusion llms. *Advances in Neural Information Processing Systems*, 38:160518–160551, 2026.

Yuhan Wang, Shiyu Ni, Zhikai Ding, Zihang Zhan, Yuanzi Li, and Keping Bi. Evaluating and calibrating llm confidence on questions with multiple correct answers. *arXiv preprint arXiv:2602.07842*, 2026.

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 *International Conference on Learning Representations*, volume 2026, pp. 57027–57051, 2026a.

Jingxuan Wu, Zhenglin Wan, Xingrui Yu, Yuzhe Yang, Yiqiao Huang, Ivor Tsang, and Yang You. Time-annealed perturbation sampling: Diverse generation for diffusion language models. *arXiv preprint arXiv:2601.22629*, 2026b.

Chenkai Xu, Yijie Jin, Jiajun Li, Yi Tu, Guoping Long, Dandan Tu, Mingcong Song, Hongjie Si, Tianqi Hou, Junchi Yan, et al. Lopa: Scaling dllm inference via lookahead parallel decoding. *arXiv preprint arXiv:2512.16229*, 2025.

Anton Xue, Litu Rout, Constantine Caramanis, and Sanjay Shakkottai. Ancoder: Anchored code generation via discrete diffusion models. *arXiv preprint arXiv:2602.17688*, 2026.

Jingyi Yang, Guanxu Chen, Xuhao Hu, and Jing Shao. Taming masked diffusion language models via consistency trajectory reinforcement learning with fewer decoding step. *arXiv preprint arXiv:2509.23924*, 2025.

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

Chi Zhang, Xixi Hu, Bo Liu, and Qiang Liu. Gumbel distillation for parallel text generation. *arXiv preprint arXiv:2603.22216*, 2026.

Yiming Zhang, Harshita Diddee, Susan Holm, Hanchen Liu, Xinyue Liu, Vinay Samuel, Barry Wang, and Daphne Ippolito. Noveltybench: Evaluating language models for humanlike diversity. *arXiv preprint arXiv:2504.05228*, 2025.

Chunting Zhou, Graham Neubig, and Jiatao Gu. Understanding knowledge distillation in non-autoregressive machine translation. *arXiv preprint arXiv:1911.02727*, 2019.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

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

A Valid Answer Uncertainty

This appendix provides a detailed definition of valid answer uncertainty (Section 3.3) and explains how we compute it.

Current predictive distribution. Let c_t denote the partially masked sequence at denoising step t. As in Equation 1, the position-wise token distributions define the predictive distribution

$$p_\theta(o \mid c_t) = \prod_{j=1}^{L} p_{\theta,j}(o_j \mid c_t) \tag{11}$$

for a complete sequence $o = (o_1, \ldots, o_L) \in \mathcal{V}^L$. This distribution is defined over all complete token sequences, including invalid ones. We do not renormalize it over the valid answers.

Let $\mathrm{Valid}(o) \in \{0, 1\}$ indicate whether a complete sequence o satisfies the validity criterion. Let

$$\mathcal{Y} = \{o \in \mathcal{C}(c_0) : \mathrm{Valid}(o) = 1\} = \{y^{(1)}, \ldots, y^{(R)}\} \tag{12}$$

denote the set of all valid answers at the initial state. Throughout denoising, we compute answer contribution mass over this fixed set of R answers. The subset compatible with the current context is $\mathcal{Y} \cap \mathcal{C}(c_t)$.

Valid local units. Let $\mathcal{D}$ decompose a complete token sequence o into an ordered collection of m local units, $\mathcal{D}(o) = (u_1^o, \ldots, u_m^o)$. The choice of local unit depends on the task. In generated text, a local unit may be a token or a word. In our structured tasks, it is an edge in Graph, a position–value assignment in Replace Random, a position–item assignment in Shuffle, and a cell–symbol assignment in Latin Square.

Let $\mathcal{U} = \bigcup_{i=1}^{R} \mathcal{D}(y^{(i)}) = \{u_1, \ldots, u_Q\}$ denote the set of distinct local units that occur in valid answers. Define the binary matrix

$$\mathbf{V} = (v_{ir}) \in \{0, 1\}^{R \times Q}, \qquad v_{ir} = \mathbf{1}[u_r \in \mathcal{D}(y^{(i)})]. \tag{13}$$

Each $u_r \in \mathcal{U}$ is individually valid by construction. A local unit outside $\mathcal{U}$ is individually invalid. For a complete sequence o, let $\mathbf{x}(o) \in \mathbb{N}_0^Q$ count the occurrences of each individually valid local unit:

$$x_r(o) = \sum_{k=1}^{m} \mathbf{1}[u_k^o = u_r]. \tag{14}$$

Thus, $\|\mathbf{x}(o)\|_1$ counts the individually valid local-unit occurrences in o, and $\|\mathbf{x}(o)\|_1/m$ is the fraction of local-unit slots occupied by valid units. Every occurrence is counted, including repeated occurrences of the same unit. Individually valid local units in the same sequence need not be jointly consistent with any single valid answer.

Local units by task. We compute N_t on four structured tasks with enumerable valid answers: Graph, Replace Random, Shuffle, and Latin Square. Their local units are defined as follows.

Graph. A local unit is a directed edge between adjacent node slots in the generated sequence. For a path of length ℓ, there are $m = \ell$ edge slots. A generated node sequence can contain the same edge more than once, and each occurrence contributes to its count. Here $x_r(o) \in \{0, 1, \ldots, m\}$ is the number of times the valid edge u_r occurs in o. Thus $\|\mathbf{x}(o)\|_1/m$ is the fraction of edge slots occupied by valid edges and is at most one. An invalid edge does not contribute to $\mathbf{x}(o)$, but its slot is still counted in m. The same holds for edges involving the non-node OTHER category.

Replace Random. A local unit is a position–value assignment in the output list. For a list of n entries, there are $m = n$ local-unit slots. A unit is individually valid if the value occurs at that position in at least one valid answer satisfying the exactly-one-replacement constraint.

Shuffle. A local unit is a position–item assignment in the output permutation, with $m = n$ slots for n input items. A unit is individually valid if it occurs in at least one valid permutation different from the input ordering.

Latin Square. A local unit is a cell–symbol assignment in the output grid. For the 4×4 grids used here, there are $m = 16$ local-unit slots. A unit is individually valid if it occurs in at least one valid Latin square over the specified symbols.

In the three ParallelBench tasks, each unit includes its position or cell, so it cannot occur more than once in a well-formed assignment and its count is either zero or one. GSM8K, HumanEval, MBPP, and IFEval are used for generation benchmark evaluation; we do not enumerate their valid answers or compute N_t on these benchmarks.

Joint consistency of local units. Individually valid units in o need not be consistent with a single valid answer. The ith entry of $\mathbf{V}\mathbf{x}(o)$ counts the individually valid local-unit occurrences in o whose units are contained in $y^{(i)}$. Define

$$b_i(o) = \mathbf{1}[(\mathbf{V}\mathbf{x}(o))_i = \|\mathbf{x}(o)\|_1], \qquad i = 1, \ldots, R. \tag{15}$$

Let $\mathbf{b}(o) = (b_1(o), \ldots, b_R(o))$. Each entry equals one if and only if the corresponding answer contains every individually valid unit in o. The matrix $\mathbf{V}$ records membership, while $\mathbf{x}(o)$ records occurrence counts; the test requires each unit to be present in the answer, not to occur there with the same multiplicity as in o. Invalid local units lower the fraction of valid local-unit occurrences but do not affect which entries of $\mathbf{b}(o)$ are one. The vector $\mathbf{b}(o)$ is zero when no answer contains all the individually valid units in o.

Answer contribution mass. We define the answer contribution mass assigned to valid answers as

$$\mathbf{s}_t = (s_{t,1}, \ldots, s_{t,R}) = \mathbb{E}_{o \sim p_\theta(\cdot \mid c_t)} \left[\frac{\|\mathbf{x}(o)\|_1}{m} \cdot \frac{\mathbf{b}(o)}{\|\mathbf{b}(o)\|_1} \right]. \tag{16}$$

The factor $\|\mathbf{x}(o)\|_1/m$ weights an assignment by the fraction of its local-unit occurrences that are individually valid. Normalizing $\mathbf{b}(o)$ distributes this weight equally among the answers that contain all those units. We set $\mathbf{b}(o)/\|\mathbf{b}(o)\|_1 = \mathbf{0}$ when $\mathbf{b}(o) = \mathbf{0}$, so a sequence with mutually inconsistent valid units contributes no mass. If $\mathbf{x}(o) = \mathbf{0}$, the first factor is zero and o also contributes no mass.

The total answer contribution mass is

$$\begin{aligned} S_t &= \|\mathbf{s}_t\|_1 \\ &= \sum_{\substack{o \in \mathcal{V}^L \\ \|\mathbf{b}(o)\|_1 > 0}} p_\theta(o \mid c_t) \frac{\|\mathbf{x}(o)\|_1}{m}. \end{aligned} \tag{17}$$

Thus, $0 \le S_t \le 1$. S_t is not simply the expected fraction of valid local-unit occurrences. A sequence with individually valid units contributes no mass when those units are mutually inconsistent.

Valid answer uncertainty. We define valid answer uncertainty from the answer contribution mass as

$$N_t = \underbrace{\frac{\|\mathbf{s}_t\|_1^2}{\|\mathbf{s}_t\|_2^2}}_{\text{effective answer spread}} \times \underbrace{\frac{1}{\|\mathbf{s}_t\|_1^2}}_{\text{low mass penalty}} = \frac{1}{\|\mathbf{s}_t\|_2^2}. \tag{18}$$

For $S_t > 0$, the first factor can be written as

$$K_t = \frac{S_t^2}{\|\mathbf{s}_t\|_2^2}. \tag{19}$$

Equivalently,

$$N_t = K_t \cdot \frac{1}{S_t^2}. \tag{20}$$

The first factor K_t, the effective answer spread, measures how broadly answer contribution mass is distributed across valid answers. It is the reciprocal of the sum of squared entries of the normalized mass vector $\mathbf{s}_t/S_t$. If exactly K answers receive equal positive mass and all others receive none, then $K_t = K$. In general, K_t is an effective count, not the literal number of answers with nonzero mass.

The second factor, the low mass penalty, grows as the total answer contribution mass S_t decreases. It penalizes invalid local evidence and locally valid evidence that no single valid answer can explain.

Answer uncertainty is therefore small when the total answer contribution mass is high and concentrated on a few valid answers. It is large when the mass is spread across many answers, its total is low, or both. Because of the low mass penalty, N_t is neither a pure measure of spread nor the number of valid answers, and it can exceed R. When $S_t = 0$, N_t is undefined.

Exact computation. Computing the expectation over complete sequences does not require enumerating $\mathcal{V}^L$. For Graph, exact dynamic programming evaluates the expectation under the predictive distribution, counting each occurrence of a valid edge as defined above. For Replace Random, Shuffle, and Latin Square, local units include their position or cell and cannot repeat in a well-formed assignment. The same occurrence-count definition therefore reduces to binary counts for these tasks, and their expectations can also be computed without enumerating $\mathcal{V}^L$.

Task	Problems	Valid answers	Local unit
Graph	30,000	Instance-dependent	Directed edge
Replace Random	100	n	Position–value
Shuffle	100	$n! - 1$	Position–item
Latin Square	100	576	Cell–symbol

Table 2: Task sets for the controlled analysis. For the two list tasks, $n \in \{3, 4, 5, 6\}$ is the number of distinct input entries, with 25 problems per length. Latin Square uses four symbols on a 4×4 grid without a fixed first row.

B Task Descriptions and Analysis Settings

This appendix describes the four tasks and the analysis settings used in Section 4. Each task has an enumerable set of valid answers and constraints that can be checked automatically. We use a controlled graph pathfinding task (Graph). Replace Random, Shuffle, and Latin Square are taken from ParallelBench (Kang et al., 2026), and we use the benchmark's original validators. We first describe how each task defines its valid answers and then give the decoding and measurement settings. Table 2 summarizes the fixed problem sets. Some analyses use fewer problems because they require outputs that can be mapped to structural units or states with finite N_t.

B.1 Controlled graph pathfinding

Each instance is a layered directed acyclic graph with a designated start node and goal node. Edges connect adjacent layers. The prompt lists the directed edges and asks for one start-to-goal path. Valid paths can branch and merge, so several answers may share a node or an edge.

Let ℓ denote the number of edges in a complete path. The corpus covers path lengths $\ell \in \{3, 4, 5, 6, 7, 10\}$, with 5,000 instances per length and 30,000 in total. For each length, ten width/branching configurations contribute 500 instances each. A path has $\ell + 1$ node slots. The start and goal are fixed, and the $\ell - 1$ interior node slots are initially masked. The path length ℓ is therefore different from the response token-canvas length in Section 3.

The set of valid answers $\mathcal{Y}$ consists of all valid start-to-goal paths supplied with the instance. An answer is correct when its full node sequence belongs to this set. For example, a graph may admit both $S \to A \to B \to T$ and $S \to C \to D \to T$. Selecting nodes from both paths does not give a valid answer if the required connecting edge is absent. The ℓ adjacent node pairs, including the start and goal boundaries, define the local-unit slots. A directed edge is locally valid when it occurs in at least one valid path. Its entry in $\mathbf{x}(o)$ counts every occurrence, so a repeated valid edge contributes once per slot it occupies (Appendix A). Invalid edges occupy slots but do not enter $\mathbf{x}(o)$. The joint-consistency test identifies paths that contain every valid edge recorded in $\mathbf{x}(o)$.

B.2 Replace Random

The input is a list $x = (x_1, \ldots, x_n)$ of distinct names and a replacement name w that does not occur in the input. The model must replace exactly one entry with w and keep every other entry in its position. Any position may be replaced, giving

$$\mathcal{Y}_{\text{replace}}(x, w) = \{(x_1, \ldots, x_{i-1}, w, x_{i+1}, \ldots, x_n) : i = 1, \ldots, n\}. \tag{21}$$

For example, replacing one entry of $[A, B, C]$ with Z allows $[Z, B, C]$, $[A, Z, C]$, or $[A, B, Z]$. The output $[Z, Z, C]$ is invalid. Each replacement is individually allowed, but their combination violates the exactly-one constraint. There are n valid answers, each consisting of n position–value local units. The corpus contains 100 problems, evenly divided among list lengths three through six, and uses a 32-token generation canvas. The benchmark validator parses the list and checks the one-replacement rule, so any valid answer is accepted, not only the provided example.

B.3 SHUFFLE

The model receives a list $x = (x_1, \ldots, x_n)$ and must return a different ordering of the same entries. The output must preserve the input multiset and must differ from the original sequence. Thus

$$\mathcal{Y}_{\text{shuffle}}(x) = \{y : \text{multiset}(y) = \text{multiset}(x),\ y \neq x\}. \tag{22}$$

The inputs contain distinct entries, so there are $n! - 1$ valid answers, from 5 to 719 for $n = 3$ through 6. For input $[A, B, C]$, both $[B, A, C]$ and $[C, B, A]$ are valid. In contrast, $[B, B, C]$ duplicates one item and omits another, and $[A, B, C]$ violates the non-identity requirement. Each local unit is a position–item assignment, so individually plausible choices can conflict through duplication or omission. As in Replace Random, we use 25 problems at each of the four list lengths and a 32-token generation canvas. The benchmark validator parses the output list and checks that it preserves the input multiset while differing from the original ordering.

B.4 LATIN SQUARE

The prompt specifies four distinct symbols and asks for a 4×4 Latin square in comma-separated rows. Every row and every column must contain each symbol exactly once. For a symbol set Σ, the set of valid answers is

$$\mathcal{Y}_{\text{Latin}}(\Sigma) = \left\{ y \in \Sigma^{4\times 4} : \begin{array}{ll} \{y_{r,c} : c = 1, \ldots, 4\} = \Sigma & \text{for every } r, \\ \{y_{r,c} : r = 1, \ldots, 4\} = \Sigma & \text{for every } c \end{array} \right\}. \tag{23}$$

For example, the rows $[A, B, C, D]$, $[B, C, D, A]$, $[C, D, A, B]$, and $[D, A, B, C]$ form a valid square. The fixed set contains 100 problems of size 4×4, with no fixed first row and a 64-token generation canvas. All 576 Latin squares over the specified symbols are valid answers. Each assignment of a symbol to one of the 16 cells forms a local unit. A symbol can be individually admissible in several cells, yet assigning it to those cells simultaneously can repeat it within a row or column. The benchmark validator parses candidate CSV rows and checks the row and column symbol sets. We enumerate all grids that satisfy these constraints rather than using a single example as the reference answer.

B.5 DECODING AND ANALYSIS SETTINGS

Model and commitment rule. The analysis uses LLaDA-8B-Instruct. At each denoising step, we compute a token distribution at each masked position. The current token confidence at a position is the probability of its top-1 token. A commitment fixes that token at the selected position. Top-k decoding commits the tokens at the k most confident masked positions together and then refines the remaining predictions. When fewer than k masked positions remain, it selects all of them. The trajectory comparisons use $k \in \{1, 2, 4\}$. Here, k is the number of committed *positions*, not the number of vocabulary candidates used for token sampling.

Generation length and progress. In Graph, the model generates the $\ell - 1$ interior nodes. The start and goal nodes and formatting separators are fixed in advance and require no generation steps. Replace Random and Shuffle use 32-token canvases, and Latin Square uses a 64-token canvas. We measure progress as

$$\rho_t = \frac{m_0 - m_t}{m_0}, \tag{24}$$

where m_0 is the number of initially masked generation positions and m_t is the number still masked at state c_t. Progress starts at zero and reaches one when all initially masked positions have been committed. For the list and grid tasks, this is token-canvas progress, not the fraction of completed entries or cells.

Single-position interventions. We take base states along top-1 trajectories at target progress $\rho \in \{0, 0.25, 0.50\}$, using $\text{round}(\rho m_0)$ commitments. For each remaining masked position, we make a separate copy of the base state and commit its top-1 token from the base prediction. We then refine the remaining predictions and measure $\Delta N_j = N_{t+1}^{(j)} - N_t$. Positive values indicate increased answer uncertainty, and negative values indicate decreased answer uncertainty. Each intervention is evaluated independently from the same base state. The Graph position-effect plots use the target $\rho = 0.25$ across all six path lengths. Rounding can cause the actual committed fraction to differ from the target, especially for short graph paths.

Matched sequential–parallel comparisons. The fork analysis uses $k \in \{2, 3, 4\}$ and the same target progress values. At each eligible base state, we fix a set of k positions by base-state confidence. The parallel arm commits the top-1 tokens at all selected positions from the same base prediction. The sequential arm commits one position at a time, with intermediate refinement after each commitment. At each step, it selects the most confident remaining position *within the same set* and commits its current top-1 token. The two arms therefore commit the same positions but may assign different tokens. We compare their N values after refining the predictions at the final states. A comparison is included only when at least $k + 1$ masked positions remain in the base state, ensuring that at least one remains after the k commitments. Under this rule, the Graph comparisons cover $\ell \in \{6, 7, 10\}$ and contain no $\rho = 0.75$ base states. The Graph comparisons comprise 105,000 pairs from 15,000 problems. Similar scalar N values do not imply identical answer contribution mass vectors or equal final-answer accuracy.

Trajectory aggregation and numerical evaluation. For the Graph trajectory plot, all 30,000 problems contribute to a 101-point progress grid. Between commitments, each trajectory is held at its most recent observed state. We report the arithmetic mean of finite $\log_2 N$ values and the interquartile range, without interpolating unobserved model predictions. Nonfinite values are counted separately and are not replaced by finite constants. Graph trajectory values are evaluated by exact dynamic programming. We retain probability mass on non-node tokens in a separate invalid-evidence category without renormalizing the node probabilities. The N values plotted in Section 4 are computed from exact expectations under the model's position-wise token distributions, using the occurrence-count convention in Appendix A. The Graph fork comparison also uses exact dynamic programming for all 210,000 arms.

B.6 Valid Answer Uncertainty on the Analysis Tasks

In every task, the instance fixes $\mathcal{Y}$. The answer–unit matrix $\mathbf{V}$ and the observed-unit vector $\mathbf{x}(o)$ determine the joint membership $\mathbf{b}(o)$. Taking the expectation under $p_\theta(o \mid c_t)$ then gives the answer contribution masses $\mathbf{s}_t$ defined in Appendix A. The task's correctness check evaluates the complete answer, whereas N_t characterizes intermediate predictive distributions through answer contribution mass. A locally valid unit does not by itself determine a valid complete answer. The local units and numbers of valid answers differ across tasks, so we report answer uncertainty separately for each task.

B.7 Aggregation and Analysis Coverage

Figure 9 shows the population trajectories for all four tasks. Each panel reports the mean of $\log N_t$ and its interquartile range at committed fractions from 0 to 1. Values between observed commitments are held constant, and tasks are never pooled.

The Graph population plot includes 500 problems in each of 60 configurations. The three ParallelBench corpora contain 100 problems each. Replace Random and Shuffle each provide 100 measurable trajectories per decoding method. For Latin Square, the corresponding counts are 68, 87, and 74 for top-1, top-2, and top-4, respectively. We exclude problems whose reference layout is malformed and cannot be mapped to structural units, so the Latin Square curves need not use identical problem sets. For ParallelBench, we map the structural units in each method's serialized output to their token spans. Differences in serialization can therefore change this mapping even at the initial prediction state.

Finite values and progress alignment. For each task and method, a trajectory contributes its most recently observed state to each point on a 101-point progress grid. The curves report the arithmetic mean of $\log_2 N$ and the interquartile range over the finite values at each point. As N becomes nonfinite, the set of contributing problems can change; late-stage averages therefore describe only the states with finite N. The measurable problem counts are given above. The step-change distributions in the main text and this appendix use consecutive top-1 states before grid alignment. Values repeated during alignment are therefore not counted as additional commitments with no change in N. Each finite step has equal weight within its task, and tasks are never pooled. Steps with a nonfinite endpoint are counted separately.

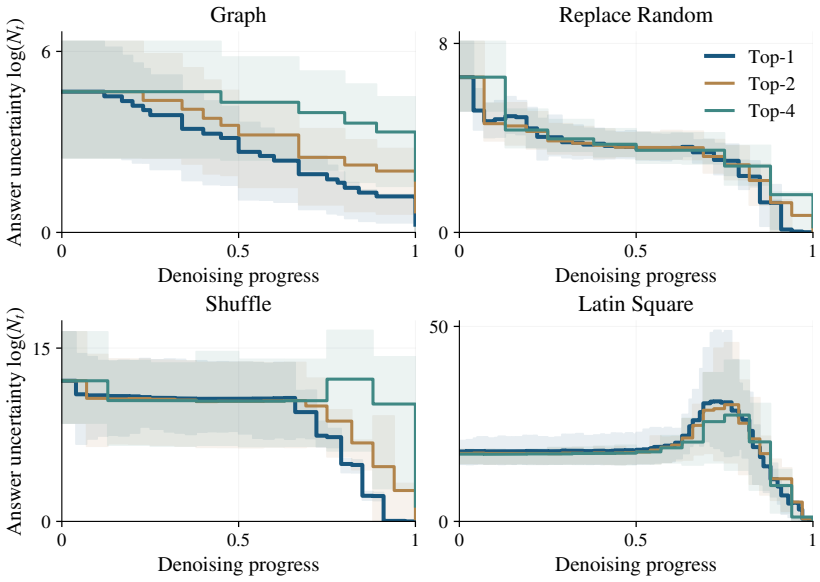

Figure 9: Answer uncertainty during denoising on Graph, Replace Random, Shuffle, and Latin Square. Lines show top-1, top-2, and top-4 decoding, and shaded bands show the finite-value interquartile range. Graph uses 30,000 problems, and each ParallelBench task uses up to 100 measurable problems.

Example selection. The three Graph trajectory examples use $\ell = 6, 7, 10$, respectively, and have finite N at every measured state. We select trajectories with stable intervals followed by sharp decreases, ranking them by the length of the stable interval, the size of the decrease, and problem ID. These examples illustrate the pattern of plateaus followed by sharp decreases. The step-change distributions, rather than these selected examples, characterize this behavior across the population.

Figure 4 uses Graph item `L07_w4_b3/283` at target $\rho = 0.25$ (attained $\rho = 1/3$). We select it from states in which every remaining position yields a finite intervention result. The example panel shows twelve valid paths and four independent interventions from the same base state. At serialized token positions $2, 4, 6, 8$, the predicted tokens are `Ed`, `Ha`, `Ha`, and `Rt`, respectively. The paths first split among three nodes (`B`, `C`, `D`) and merge at `E`. They then branch again across the `F`/`G`, `H`/`I`, `J`/`K`, and `L`/`M` columns before reconverging at `N`. The `E` bar at position 6 denotes token `Ha`, which is shared at position 4 but incompatible at position 6. Starting from $N_t = 19.28$, the plotted changes are -15.35, -0.51, $+25.83$, and -14.02. Both *branch* choices reduce N, the *incompatible* choice increases it, and the *shared* choice changes it only slightly. The four interventions each start from the same context and are not consecutive decoding steps. The right panel compares commitment roles across the Graph population, as described below. This example illustrates position dependence and does not estimate how often it occurs.

Commitment roles in the Graph population view. Figure 4 groups predictions by their relation to the valid paths compatible with the current context. A *branch* prediction agrees with a nonempty proper subset of those paths at its position. A *shared* prediction agrees with all of them, and an *incompatible* prediction agrees with none. These commitment roles are used to interpret the interventions; the metric itself is still computed over the instance's fixed set of valid answers $\mathcal{Y}$. The population panel and the example bars both report the raw difference $\Delta N_j = N_{t+1}^{(j)} - N_t$. The population panel uses a symmetric logarithmic vertical axis with a linear region between -1 and 1. Within each role, we retain values within 1.5 interquartile ranges of the lower and upper quartiles before drawing the box plot; outlier markers are hidden. Zero denotes no change in N; negative and positive values indicate decreases and increases, respectively. A *shared* commitment need not leave N unchanged, because refinement can redistribute predictive mass without narrowing the set of compatible valid answers. Colors distinguish commitment roles rather than categories based on

Task	Finite steps	Nonfinite steps
Graph	136,199	8,801
Replace Random	3,200	0
Shuffle	3,200	0
Latin Square	3,979	373

Table 3: Coverage of top-1 step changes. A step is classified as nonfinite if N is undefined or nonfinite at either endpoint; these steps are excluded from the plotted densities.

Task	Finite pairs	Nonfinite pairs	Layout failures	Corpus
Graph	97,639	7,361	0	15,000
Replace Random	1,200	0	0	100
Shuffle	1,191	9	0	100
Latin Square	776	40	32	100

Table 4: Coverage of matched sequential–parallel comparisons. Layout failures are problems whose saved reference layout cannot be mapped to the required structural units. A pair is classified as nonfinite if either arm has a nonfinite final value of N.

the magnitude of the change in N. The task-wise step-change distributions and galleries below show the same effects without grouping by role.

Tables 3 and 4 report the finite and nonfinite observations used in the step-change and matched-fork plots.

B.8 Accuracy–NFE Trade-offs on the Analysis Tasks

Table 5 reports accuracy and mean NFE for AnF and the baselines on the four analysis tasks. We evaluate all methods and parameter settings on the same 1,000 Graph problems and all 100 problems in each ParallelBench task. Generation length is 128 tokens for Graph and 64 tokens for each ParallelBench task; the earlier trajectory analyses use 32 tokens for Replace Random and Shuffle.

For Graph, we evaluate 27 AnF settings using all combinations of $\gamma \in \{0.8, 0.9, 1.0\}$, $K \in \{2, 3, 4\}$, and $\tau \in \{0.7, 0.8, 0.9\}$. For each ParallelBench task, we extend the target confidence grid to $\gamma \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, giving 54 AnF settings with the same K and τ grids. On every task, we evaluate top-k decoding with $k \in \{1, 2, 4, 8\}$, Fast-dLLM with confidence thresholds in $\{0.5, 0.7, 0.9, 0.95\}$, PUNT with $\epsilon \in \{0.01, 0.04, 0.08, 0.32\}$, and DAPD with maximum thresholds in $\{0.02, 0.05, 0.15, 0.4\}$. For ETE, we evaluate all 12 combinations of $C \in \{0.5, 0.7, 0.9, 0.95\}$ and $N \in \{3, 8, 16\}$. Threshold decoding uses confidence thresholds in $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ on Graph and $\{0.7, 0.8, 0.9\}$ on each ParallelBench task. In total, we evaluate 60 settings on Graph and 85 on each ParallelBench task across all methods, including AnF.

AnF uses a task-specific decoder with full forward passes on Graph and the general cached decoder on the three ParallelBench tasks. Fast-dLLM and ETE use blocks of $\min(64, L)$ tokens, where L is the generation length; top-k decoding treats the full output as one block. Accuracy is the fraction of complete outputs that satisfy the task's correctness criterion. Incomplete outputs are counted as incorrect, and their forward calls are included in mean NFE. For each method and task, we report the evaluated setting with the highest accuracy, breaking ties by lower mean NFE.

B.9 Single-Commitment Effects across Tasks

This subsection extends the single-position commitment analysis in Section 4.2 to all four tasks.

Intervention and context construction. We extend Figure 4 with ten examples per task (Figures 12–15) and a population-level summary of finite comparisons at target $\rho = 0.25$ (Figure 11). Within each example, every intervention starts from the same state on a top-1 denoising trajectory. We commit the top-1 token at one masked position and refine the remaining predictions with one ad-

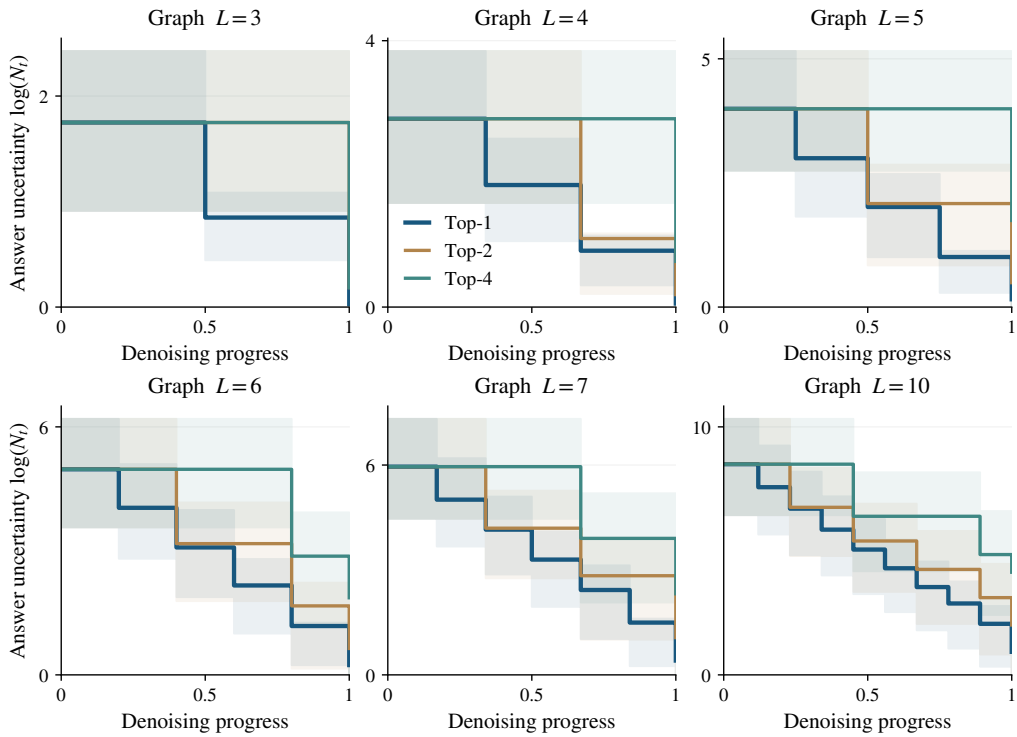

Figure 10: Graph trajectories separated by path length. Each panel uses all 5,000 problems at that length and the same finite-value aggregation as Figure 1 (Left). Lines denote top-1/2/4, and shading denotes the IQR.

Table 5: **Highest-accuracy operating points on the four analysis tasks.** Accuracy (Acc.) is in percent, and NFE is the mean per problem. For each method and task, we report the setting with the highest accuracy, breaking ties by lower mean NFE. Bold marks the highest accuracy for each task, including ties. Forward-call budgets vary across the reported settings.

	Graph		Replace Random		Shuffle		Latin Square	
Method	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE
Top-k	**78.0**	4.78	**100.0**	64.00	**100.0**	64.00	31.0	64.00
Threshold	**78.0**	4.64	98.0	6.29	99.0	9.26	29.0	29.11
Fast-dLLM	54.9	4.69	80.0	9.45	67.0	11.24	23.0	29.13
PUNT	74.7	12.30	98.0	29.70	**100.0**	33.42	18.0	111.03
DAPD	71.8	4.36	98.0	23.97	**100.0**	18.38	40.0	27.79
ETE	72.8	4.71	99.0	5.97	97.0	6.93	24.0	17.79
AnF	74.6	4.22	**100.0**	6.13	98.0	8.58	**45.0**	15.97

ditional forward pass. We report $\Delta N_j = N_{t+1}^{(j)} - N_t$, with N_t evaluated exactly under the model's position-wise predictions. In Graph, the start and goal tokens and separators are supplied by the task; all other visible tokens were committed by top-1 decoding. In ParallelBench, the entire response canvas is initially masked. No intermediate state is built by revealing an oracle answer. Each example shows the actual committed fraction, which can differ from the target after rounding.

Population summary and coverage. Figure 11 summarizes all finite comparisons, independently of the forty gallery examples. For state s, define $z_{s,j} = \log_2(N_{s,t+1}^{(j)}/N_{s,t})$. The left column shows an empirical cumulative distribution function (ECDF) in which each state receives equal total weight, divided equally among its available positions. Negative values denote a decrease in N and positive values an increase, with $z = -1$ denoting halving and $z = 1$ doubling. The right column

plots the ECDF of $R_s = \max_j z_{s,j} - \min_j z_{s,j}$ over states with at least two finite interventions. $R_s = 1$ means that the largest and smallest post-intervention values of N from the same state differ by a factor of two. This range measures how the effect of a commitment varies with its position while accounting for differences in the scale of N across states. The horizontal axes are linear in these log-ratio statistics and display the full range of values.

Graph covers 30,000 problems and 110,000 possible interventions at this progress. The analysis includes 109,612 finite comparisons from 29,996 states; the remaining 388 comparisons are excluded rather than assigned a value of zero. Replace Random covers all 100 problems. Shuffle contributes 96 of 100 problems, and Latin Square contributes 68 of 100 problems. The Graph within-state ECDF uses 29,939 states with at least two available positions. The population summaries describe only these measurable subsets.

The committed position changes answer uncertainty. The median within-state range is positive for every task. The choice of position therefore changes the resulting N even when the starting context and the number of commitments are fixed. Some interventions decrease N, while others increase it. Because N combines effective answer spread with a low mass penalty, an increase does not necessarily indicate a larger number of valid answers. These single-token interventions alone do not establish the safety of a parallel decoding policy.

Gallery selection. We select ten states per task from those with finite comparisons for every remaining position. The Graph examples span path lengths $3, 4, 5, 6, 7, 10$; examples with the same path length use different width/branching configurations. For the list tasks we balance input-list lengths, and for Latin Square we prioritize distinct symbol sets. Within each stratum, we use a deterministic procedure to select examples spanning the ranked values of $(\max_j \Delta N_j - \min_j \Delta N_j)/N_t$. These examples illustrate a range of structures and intervention effects; they are not a random sample from which to estimate population frequencies.

Reading the galleries. Each example reports $R = |\mathcal{Y}|$, the number of valid answers in the fixed set defined in Section 3.3. R counts all valid answers to the problem, rather than only the displayed examples or the remaining valid answers at the intermediate context. Graph shows the full task graph, which contains all valid start-to-goal paths. Dark nodes mark supplied or already committed tokens. Dashed outlines mark masked positions and use the colors of the corresponding interventions. Replace Random shows all n valid answers. Shuffle and Latin Square show $y^{(1)}, y^{(2)}, \ldots, y^{(R)}$, with the ellipsis indicating omitted valid answers. The two list tasks show the literal input entries and, for Replace Random, the replacement text. The bar chart below each structure shows all independent interventions from the same state, indexed by zero-based serialized token position. In Graph, each bar is also labeled with the committed token. Blue denotes a negative change in N and orange a positive change. Axes marked symlog use a linear scale within $0.01N_t$ of zero and a logarithmic scale outside that interval. In the three ParallelBench galleries, each bar is labeled with its top-1 token. A bracket groups consecutive bars with the same token and is labeled with that token and the number of bars. EOS and EOT abbreviate end tokens, and the visible space mark denotes a leading space.

ParallelBench commitments act on tokenizer positions, not on whole list entries or grid cells. We therefore evaluate interventions across the full 32-token list and 64-token grid canvases, including positions for separators and end tokens. Committing these tokens can change the refined predictions at content positions. Already committed positions have no bar. For ParallelBench, we derive the mapping from token positions to structural units from the final response produced by top-1 decoding. We hold this mapping fixed across interventions rather than recomputing it from potentially malformed intermediate predictions.

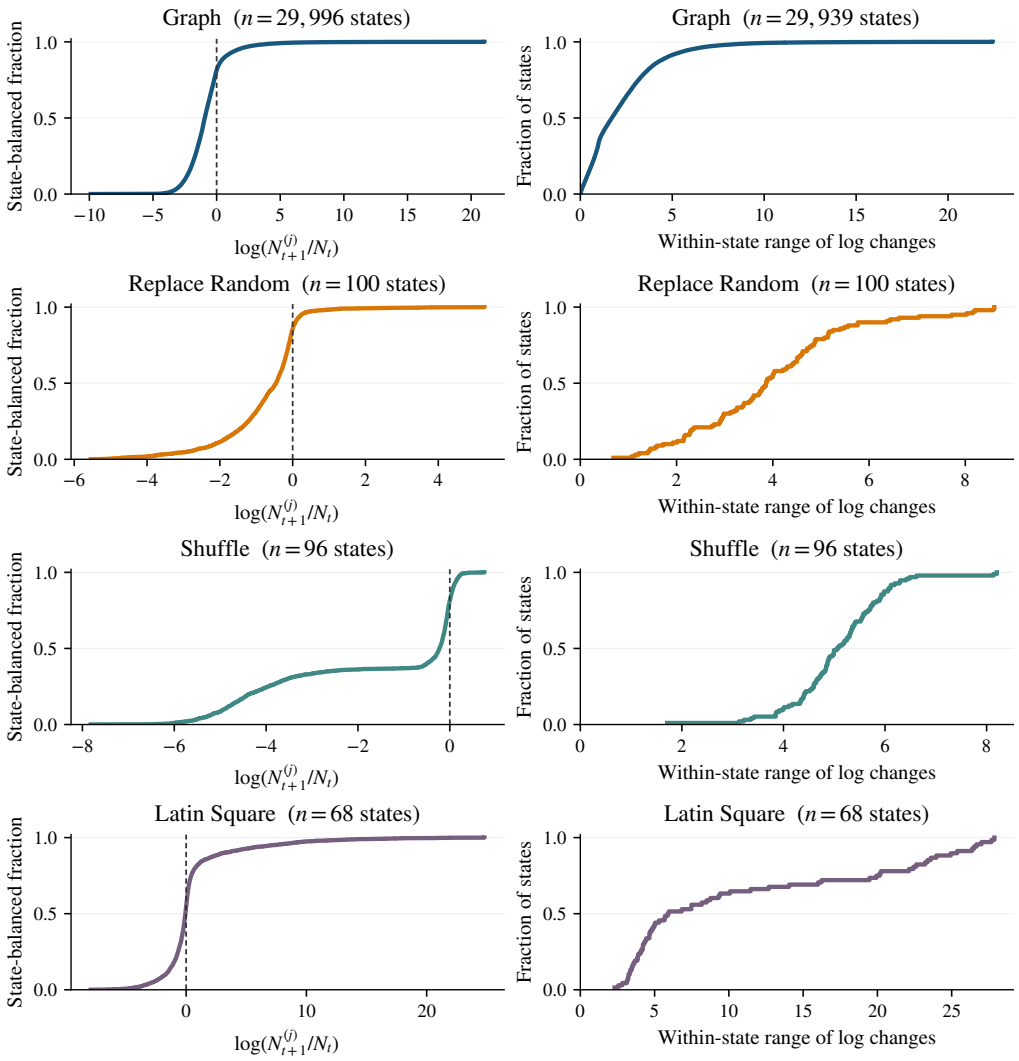

Figure 11: Position-dependent effects across all available finite single-token interventions. Left: distributions of the log ratio of N after and before each commitment, with equal total weight for each state. Right: distributions of the within-state range of that ratio. All comparisons use target $\rho = 0.25$. The right panels measure variation across positions in the same context, not variation between unrelated problems.

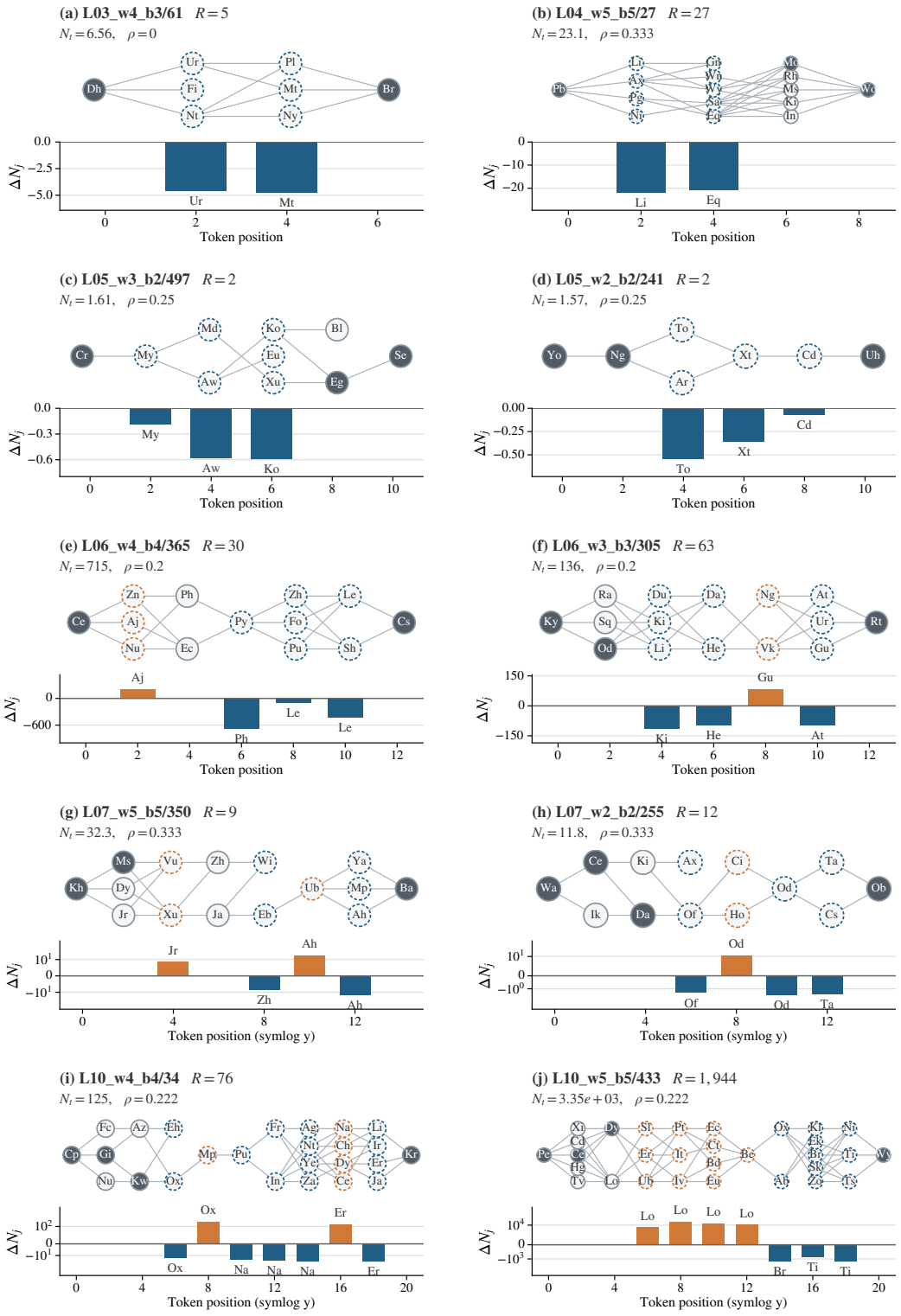

Figure 12: Graph: ten single-commitment examples. Each panel shows the full task graph and $R = |\mathcal{Y}|$, followed by the changes in N after independently committing each remaining position. Dark nodes mark supplied or committed tokens, and dashed outlines mark masked positions. Bar labels give the committed token. Headers report item IDs, initial N_t, and actual committed fractions ρ. Blue denotes a decrease and orange an increase.

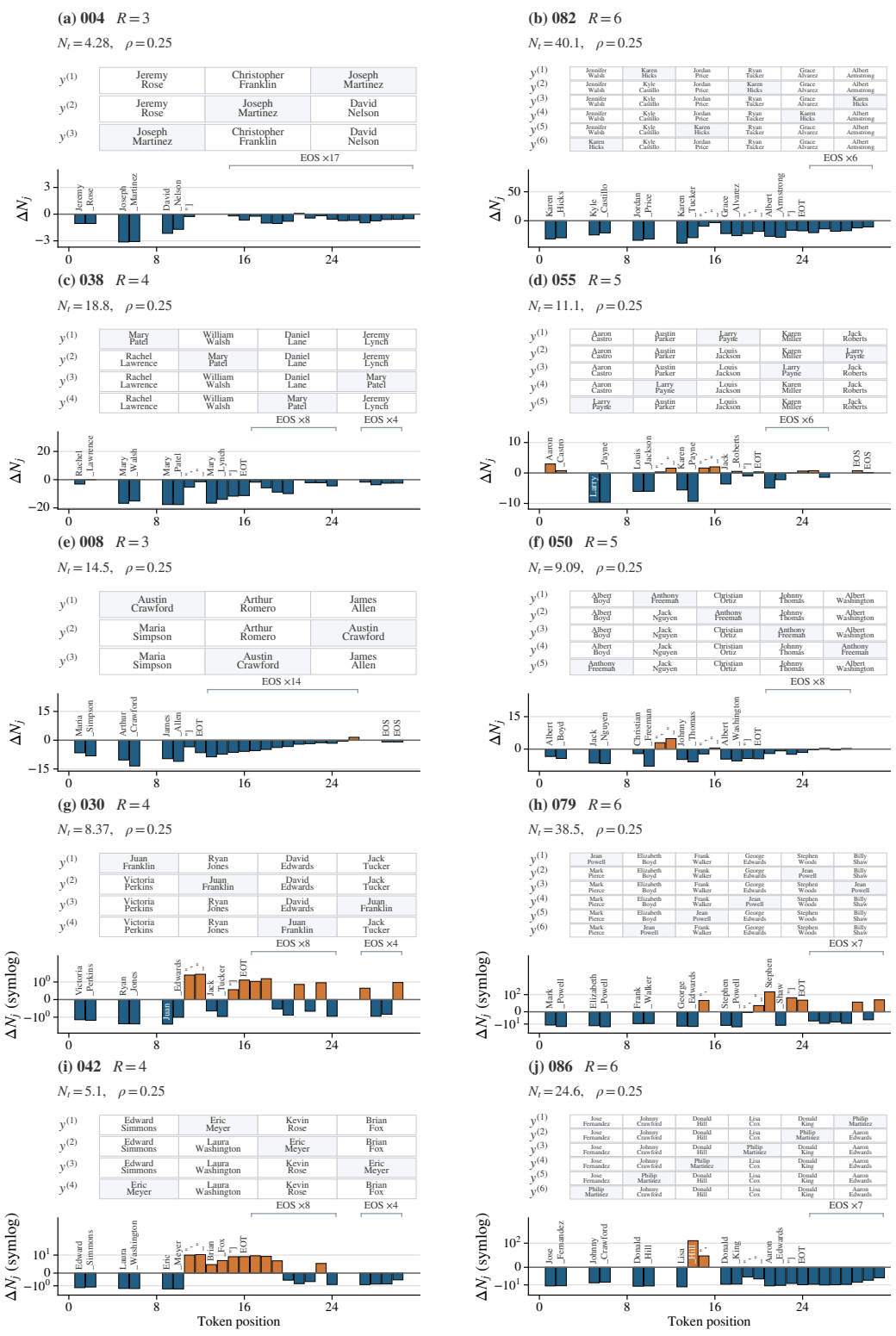

Figure 13: Replace Random: ten single-commitment examples. All $R = n$ valid answers are shown with the actual list entries and replacement text. Each bar is an independent token-position intervention from the same state. Headers give $|\mathcal{Y}|$, item IDs, initial N_t, and actual committed fractions ρ. Blue denotes a decrease in N and orange an increase.

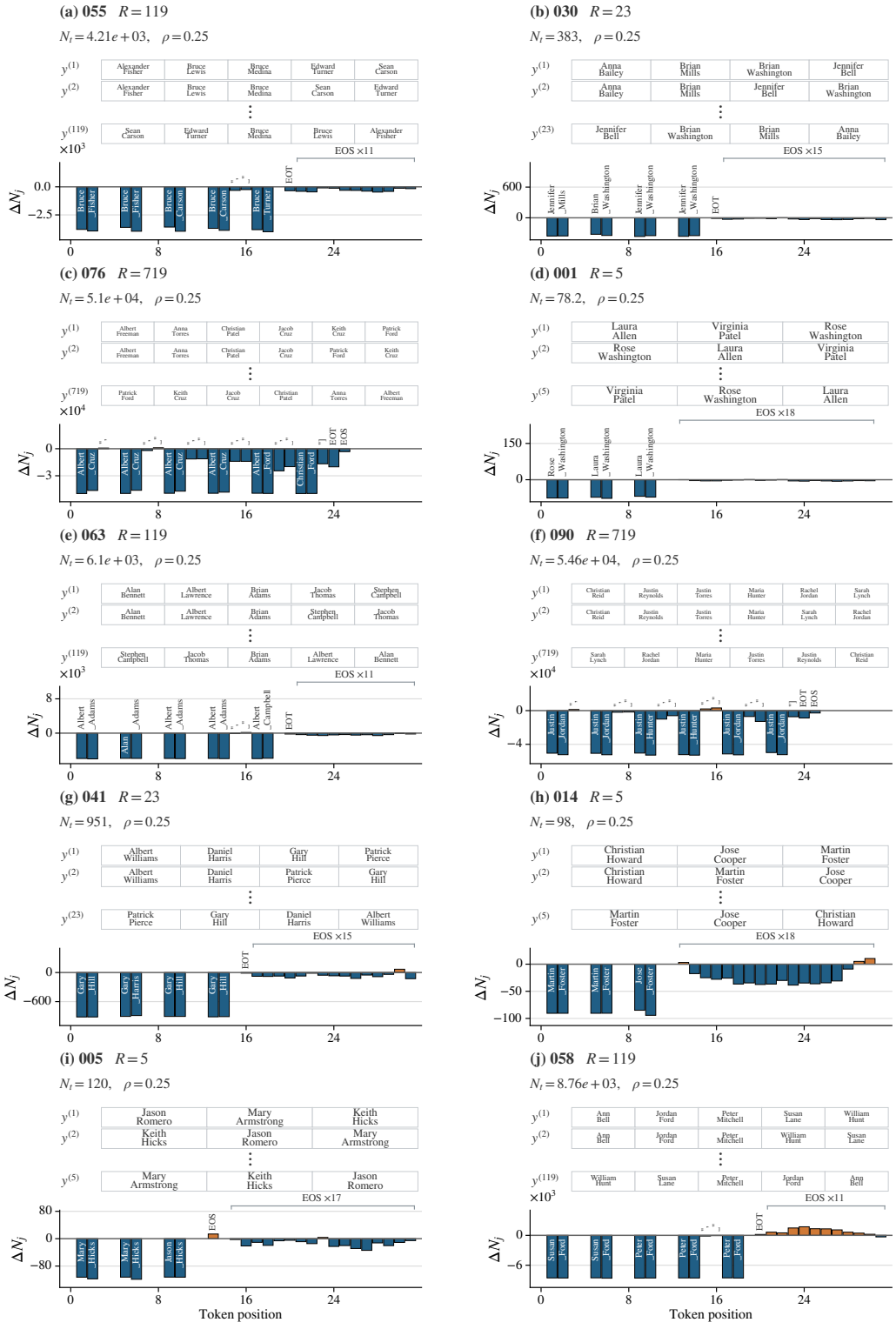

Figure 14: Shuffle: ten single-commitment examples. The full set contains $R = n! - 1$ valid answers; in $y^{(1)}, y^{(2)}, \ldots, y^{(R)}$, the ellipsis indicates omitted valid answers. Actual list entries are shown. Bars show independent token-position interventions from the same state. Headers give $|\mathcal{Y}|$, item IDs, initial N_t, and actual committed fractions ρ. Blue denotes a decrease in N and orange an increase.

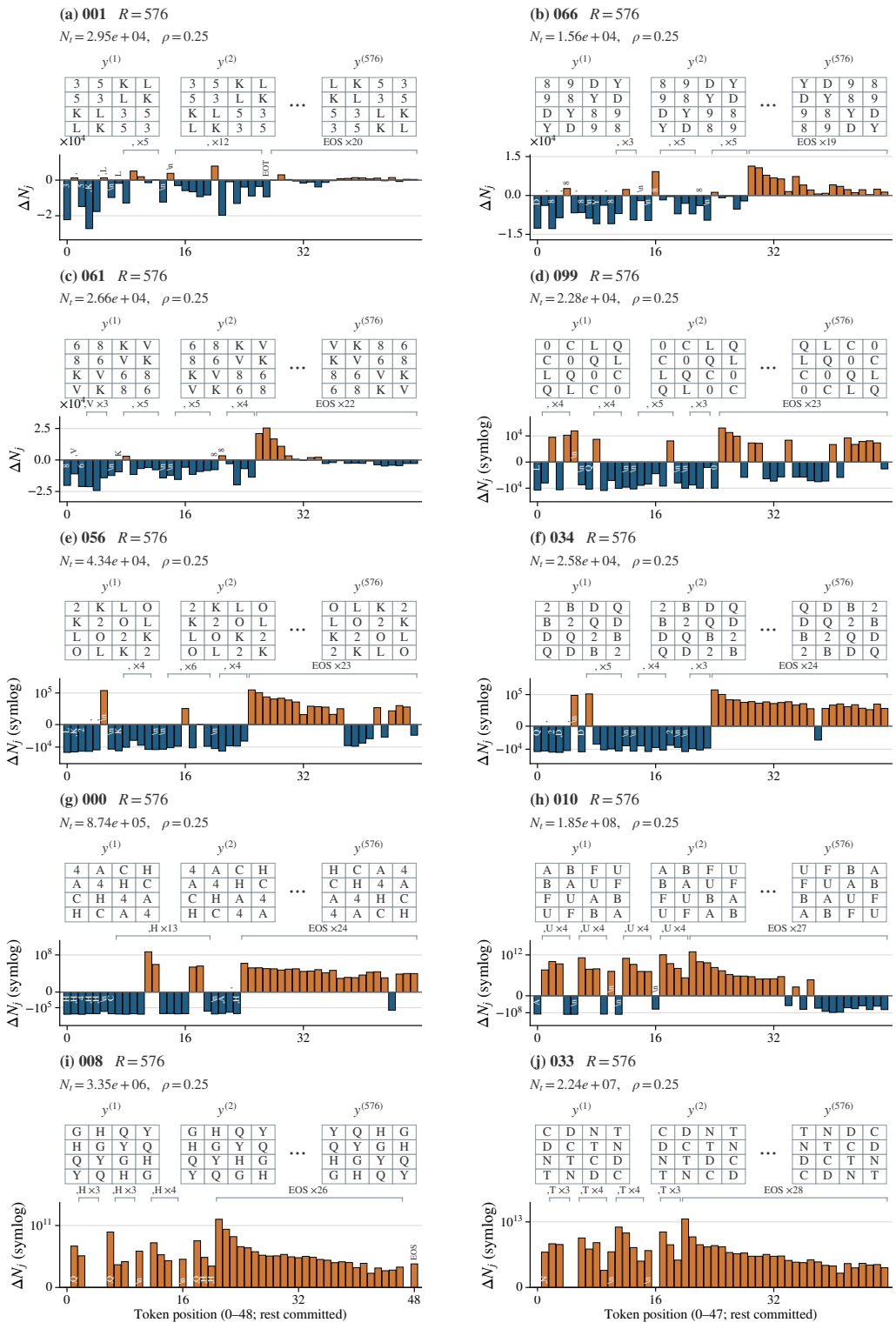

Figure 15: Latin Square: ten single-commitment examples. Each problem has $R = 576$ valid answers. The displayed grids are $y^{(1)}, y^{(2)}, y^{(576)}$, with an ellipsis indicating the omitted valid answers. The metric uses all 576 valid answers. Bars show independent token-position interventions from the same state. Headers give item IDs, initial N_t, and actual committed fractions ρ. Blue denotes a decrease in N and orange an increase.

B.10 Task-wise Trajectories and Sequential–Parallel Comparisons

Figures 16–19 extend Figures 3 and 5 to all four tasks. The Graph comparisons include the $k = 4$ pairs summarized in Figure 5, and the other three tasks extend the analysis beyond graph pathfinding. In each figure, the upper row shows three top-1 examples and the task's signed step-change distribution, expanding Figure 3 (Right). The lower row compares sequential and parallel commitments separately for $k = 2, 3, 4$ using all measurable matched pairs, extending Figure 5. All plots use exact expectations under the model's position-wise predictions and the task-specific local-unit convention described above.

Reading the step-change distributions. The horizontal axis is $\Delta \log N_t = \log N_{t+1} - \log N_t$, computed with base-2 logarithms. Negative values indicate decreases and positive values increases, and the distance from zero gives the size of the multiplicative change. The dashed line marks no change. For visualization, we exclude outliers with the 1.5-IQR rule and estimate each kernel density from the retained transitions. Numerical statistics use all transitions with finite, positive N at both endpoints, before outlier filtering. The distributions describe variation across steps and do not classify individual commitments as correct or incorrect.

Example selection and panel scales. We select trajectory examples with stable intervals followed by sharp decreases, as described above. The Graph examples use path lengths $\ell = 6, 7, 10$. For each list task, we rank eligible problems by the length of their longest stable interval and then by the size of the subsequent decrease. We select three problems with distinct input-list lengths. For Latin Square, we apply the same ranking and require distinct symbol sets. These selected examples do not establish how often such patterns occur. Each task's distribution is computed from all adjacent finite top-1 states, independent of the displayed examples. Each density panel uses its own y-axis range, so peak heights should be interpreted using the scale of that panel.

Matched-fork coverage. Each lower panel shows the density of $N^{\mathrm{seq}}_{t+k} - N^{\mathrm{par}}_{t+1}$ for one k. The horizontal axis shows the endpoint difference with signed-log spacing, as in the main figure. Kernel densities are estimated from all finite matched pairs. For display, we limit the horizontal range to the 0.1–99.9 percentile interval with a small margin on either side. Panel titles give the finite-pair counts. Table 4 reports the numbers of finite pairs, nonfinite pairs, and problems excluded because their structural layouts could not be mapped.

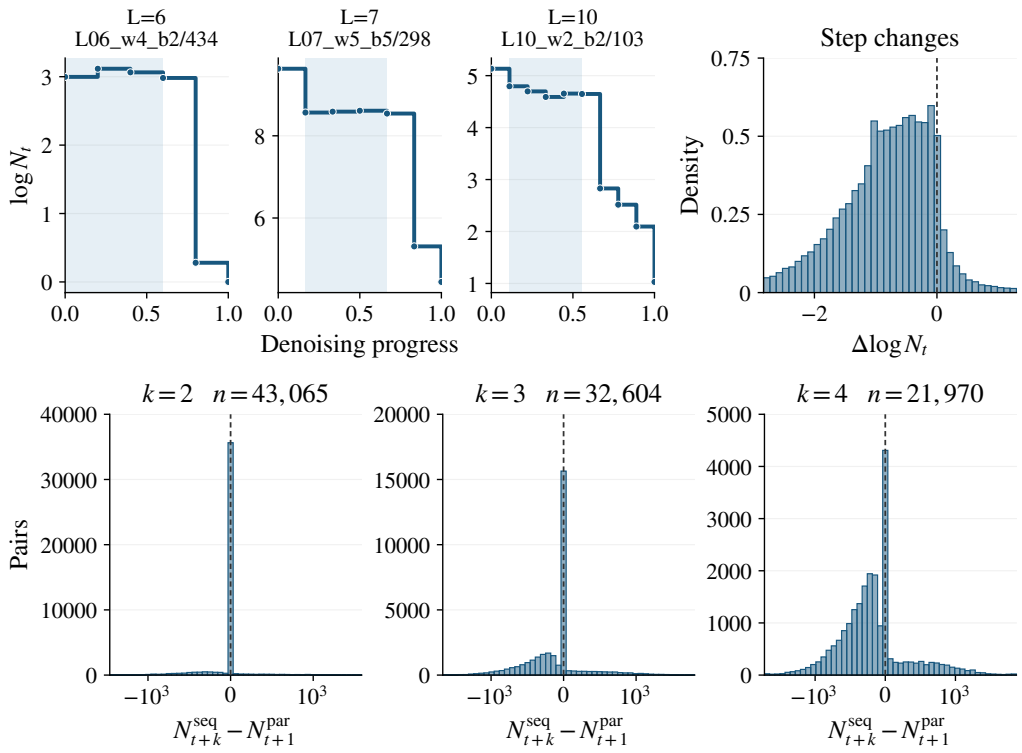

Figure 16: Additional Graph trajectories and sequential–parallel comparisons, supplementing Figures 3 and 5. Top: examples at path lengths $6, 7, 10$ and the top-1 signed step-change distribution, with outliers excluded for visualization. Bottom: distributions of sequential–parallel differences over finite pairs among 105,000 Graph comparisons, shown separately for each k. Shading in the trajectory panels marks selected stable runs before a sharp drop.

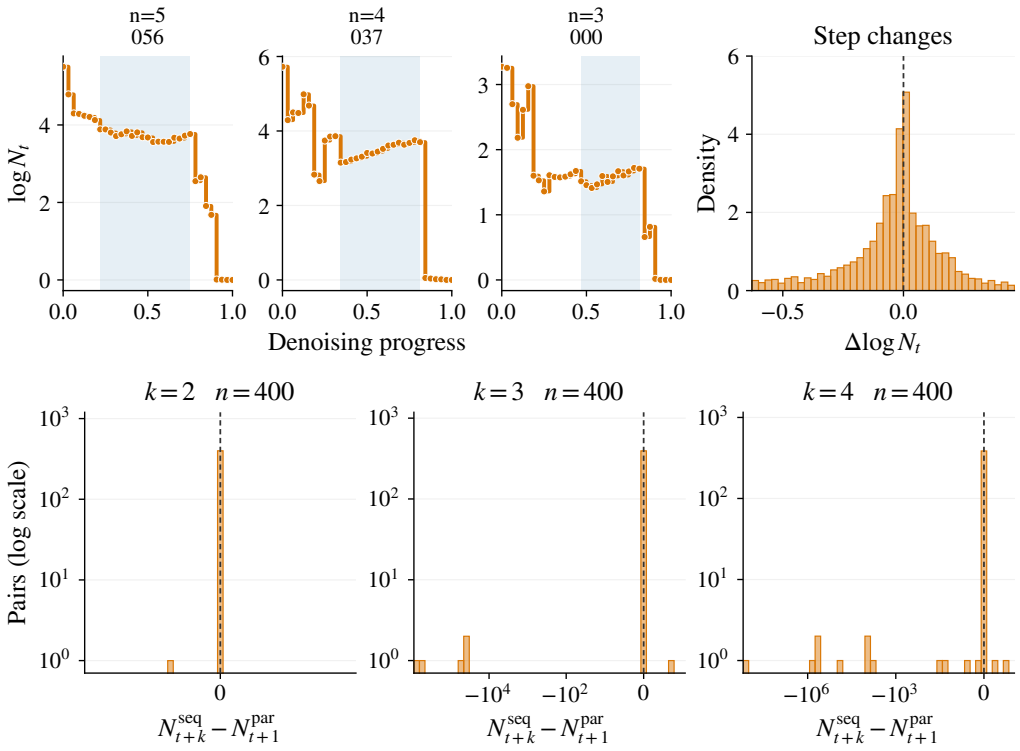

Figure 17: Replace Random. Top: top-1 examples with distinct input-list lengths and the signed step-change distribution, with outliers excluded for visualization. Bottom: distributions of the difference in final N between matched sequential and parallel commitments for $k = 2, 3, 4$. The evaluation covers all 100 problems, and the comparison covers 1,200 matched pairs. Plot conventions match Figure 16.

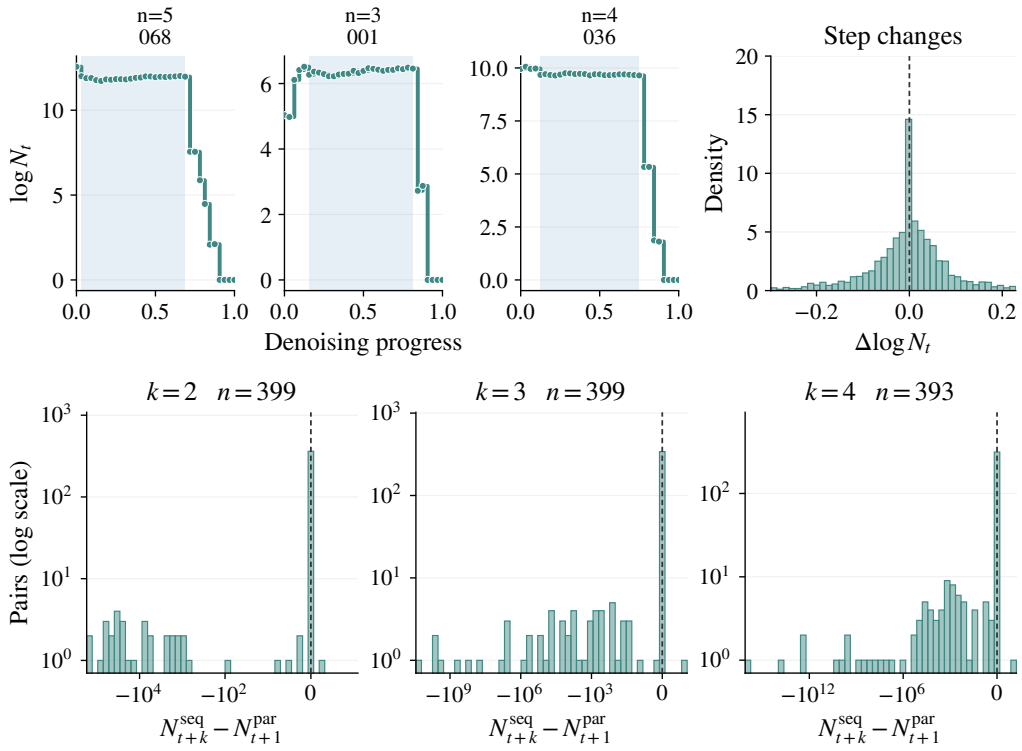

Figure 18: Shuffle. Top: top-1 examples with distinct input-list lengths and the signed step-change distribution, with outliers excluded for visualization. Bottom: distributions of the difference in final N between matched sequential and parallel commitments for $k = 2, 3, 4$. The distribution uses all 100 top-1 trajectories, and 1,191 of the 1,200 fork pairs have finite endpoints. Plot conventions match Figure 16.

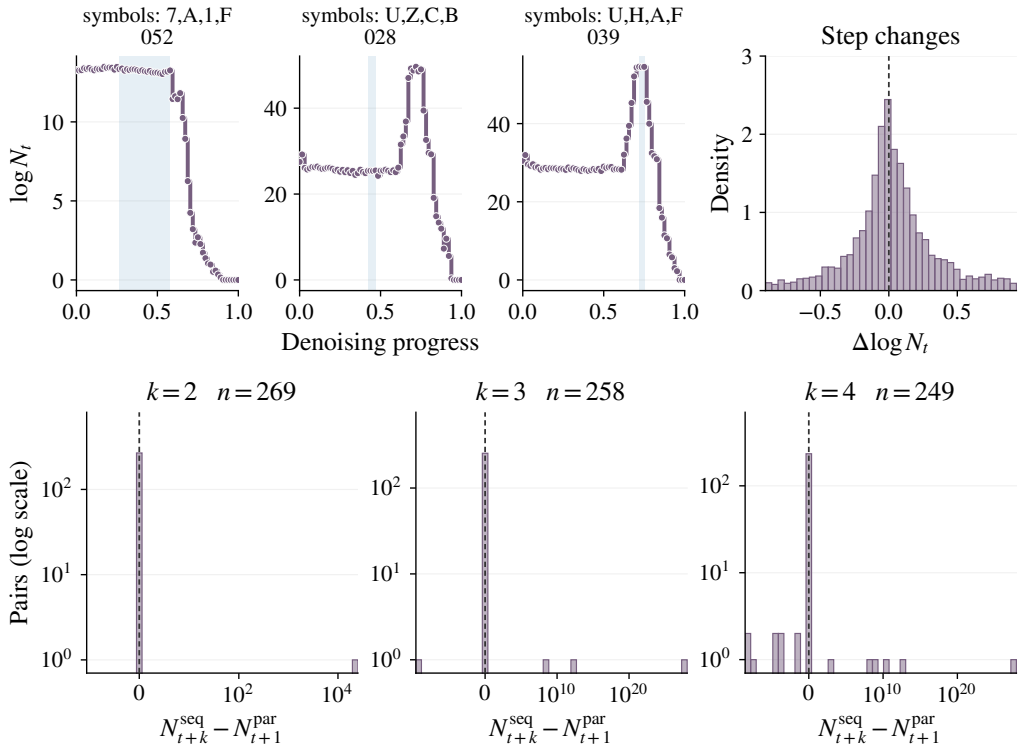

Figure 19: Latin Square. Top: top-1 examples with distinct symbol sets and the signed step-change distribution, with outliers excluded for visualization. Bottom: distributions of the difference in final N between matched sequential and parallel commitments for $k = 2, 3, 4$. The distribution uses finite steps from 68 measurable top-1 trajectories. Of 816 measurable fork pairs, 776 have finite endpoints, and 32 problems lack a usable reference layout. Plot conventions match Figure 16.

C Anchor and Follower Selection on Generation Benchmarks

This appendix supplements Section 6.3 with comparisons of anchor selection and follower thresholds on the four generation benchmarks.

Entropy-based anchors outperform random anchors. We compare AnF with a decoder that selects a random anchor. Both methods select anchors within the same 64-position window and use a follower threshold of $\tau = 0.9$. AnF uses $(\gamma, K) = (1, 3)$, while the random-anchor baseline samples one eligible anchor uniformly with seed 0. Table 6 shows that AnF has higher accuracy on all four tasks. The gains are 8.0, 17.0, 20.3, and 1.5 percentage points on GSM8K, HumanEval, MBPP, and IFEval, respectively. The methods are not matched on mean NFE, and the random-anchor baseline is evaluated with only one seed.

Table 6: **Anchor selection on four generation benchmarks.** Accuracy (Acc.) is in percent and NFE is the mean per problem, on the full benchmark cohorts (GSM8K: 1,319; HumanEval: 164; MBPP: 257; IFEval: 541). IFEval uses prompt-level loose accuracy. Both methods use follower threshold $\tau = 0.9$ and the same window. The random-anchor baseline uses one seed, and the methods are not matched on mean NFE.

	GSM8K		HumanEval		MBPP		IFEval	
Method	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE
Random anchor	71.6	66.7	29.3	44.3	26.8	30.1	58.6	174.9
AnF	79.6	80.9	46.3	42.7	47.1	28.6	60.1	166.3

The follower threshold controls the accuracy–NFE trade-off. The follower threshold controls how many follower tokens are committed in parallel from the anchor-conditioned predictions. With $(\gamma, K) = (1, 3)$ fixed, raising τ from 0.7 to 0.95 increases HumanEval accuracy from 35.4% to 47.0%. Mean NFE increases from 29.9 to 51.7 over the same range. Raising τ further to 0.99 gives 44.5% accuracy at 74.1 mean NFE. Thus, a higher threshold does not necessarily improve accuracy. Figure 20 shows the full τ sweep and the sensitivity to γ and K.

D Follower Role Analysis Details

This appendix describes how we assign token roles to followers in Section 6.3.

Traces and tasks. We analyze saved AnF traces at $(\gamma, K, \tau) = (1, 3, 0.9)$ without new model inference. For Graph, we use 100 items sampled across the graph configurations in the valid answer uncertainty analysis. Replace Random and Shuffle use 100 saved items each. For Latin Square, we use the 17 of 100 saved items for which every enumerated valid answer aligns exactly with the output tokens. Of the 83 excluded items, 17 are also excluded in Appendix G: 9 have the wrong number of structural units, and in 8 the saved output and the retokenized layout disagree. In the other 66 items, the structural units occupy different numbers of tokens across valid answers, so a follower position does not map to the same structural unit in every valid answer. We also omit 59 anchor events that lack the pre-anchor or anchor-conditioned predictions needed for the comparison.

Token roles are defined relative to the remaining valid answers. At each anchor event, we enumerate the valid answers compatible with the previous commitments and the selected anchor token. We classify the top-1 tokens from the pre-anchor and anchor-conditioned predictions relative to this same set. The comparison uses only positions selected as followers, and the pre-anchor token need not pass the follower threshold. We count only content tokens, excluding punctuation-only tokens and trailing special tokens. For the ParallelBench tasks, classification uses the observed serialization and a fixed token alignment. States with no compatible valid answer are counted separately and excluded when computing the proportions of *shared*, *branch*, and *incompatible* tokens.

Anchor conditioning increases the fraction of shared tokens within items. For each event with at least one content follower token and at least one compatible valid answer, we compute the change

in the fraction of *shared* tokens before and after anchor conditioning at the same follower positions. We average these paired changes over events within each item and then over items. This analysis includes 79 Graph, 100 Replace Random, 100 Shuffle, and 17 Latin Square items, comprising 97, 248, 333, and 71 events, respectively. The mean change is positive in 5, 56, 93, and 6 of these items, respectively, and zero in the rest. Figure 7 (Right) pools tokens across events, giving more weight to events with more follower tokens. These token roles describe individual followers; followers classified as *branch* may still be inconsistent with one another when committed in parallel.

E Decoding Configurations and Ablation Studies

This appendix describes the parameter settings for AnF and the baseline decoding methods. Generation length is 512 tokens on GSM8K, HumanEval, and MBPP, and 1,024 on IFEval. For each model and task, all methods use the same prompts and evaluation procedures. We select tokens at temperature zero. Block sizes and parameter ranges are method-specific.

AnF. We sweep the follower threshold over $\tau \in \{0.5, 0.7, 0.8, 0.9, 0.95, 0.99\}$ with target confidence $\gamma = 1.0$ and candidate budget $K = 3$. Figure 6 shows the LLaDA accuracy–NFE trade-offs, and the sensitivity analysis below shows the threshold sweep. For each anchor candidate, we commit its top-1 token from the current prediction in a separate copy of the context and perform anchor-conditioned refinement. We select the anchor candidate with the largest anchor score, defined as the normalized entropy reduction over the other masked positions in the evaluation window. We select follower positions whose top-1 token probabilities meet the follower threshold in the anchor-conditioned predictions. We evaluate the K anchor candidates in a single batched forward pass, which counts as one physical model forward call in NFE. When a step commits only the anchor token, we reuse the anchor-conditioned predictions at the next step without an additional forward pass.

Top-k and threshold decoding. Top-k decoding uses a block length of 32 tokens. We vary the number of positions committed at each denoising step over $k \in \{1, 2, 4, 8, 16\}$. The $k = 1$ configuration, which commits one token per step, appears as Sequential in Table 1 and is also shown in Figure 6. For threshold decoding, we vary the confidence threshold over $\{0.5, 0.7, 0.8, 0.9, 0.95, 0.99\}$.

Fast-dLLM. We use Fast-dLLM with dual caching and a block length of 64 tokens. Its confidence threshold varies over $\{0.5, 0.7, 0.8, 0.9, 0.95, 0.99\}$.

PUNT. We vary its tolerance parameter over $\epsilon \in \{0.01, 0.02, 0.04, 0.08, 0.16, 0.32\}$.

DAPD. We use staged decoding with a layer ratio of 0.3. For all models, we fix $\tau_{\min} = 0.01$ and sweep $\tau_{\max} \in \{0.02, 0.05, 0.1, 0.15, 0.25, 0.4\}$. For Dream-Instruct-7B, we also evaluate $(\tau_{\min}, \tau_{\max}) = (0.005, 0.05)$ on GSM8K, HumanEval, and MBPP, and $(0.1, 0.5)$ on IFEval.

ETE. We use a block length of 64 tokens and evaluate all combinations of the confidence parameter $C \in \{0.5, 0.7, 0.9, 0.95\}$ and ETE parameter $N \in \{3, 8, 16\}$ on each benchmark. ETE's parameter N is distinct from valid answer uncertainty N_t.

Measured HumanEval throughput. Table 1 is measured with LLaDA-8B-Instruct on all 164 HumanEval problems with a 512-token generation length, measuring each problem once on an NVIDIA H200 GPU. We record CUDA events around each physical model forward pass, including batched candidate-anchor evaluations and cached refinement passes. Throughput is the total number of committed output tokens divided by the total GPU-forward time; prompt tokens and temporary candidate tokens are excluded. Model loading, prompt encoding, scoring, and file writes are excluded, so these measurements quantify model-forward execution rather than end-to-end decoding latency. The operating points are AnF with $(\gamma, K, \tau) = (1, 3, 0.95)$, Sequential (top-1) with a block of 32 tokens, Fast-dLLM with a block of 64 tokens and threshold 0.95, PUNT with $\epsilon = 0.01$, DAPD with $(\tau_{\min}, \tau_{\max}) = (0.01, 0.02)$ and layer ratio 0.3, and ETE with $(C, N) = (0.95, 16)$ and a block of 64 tokens.

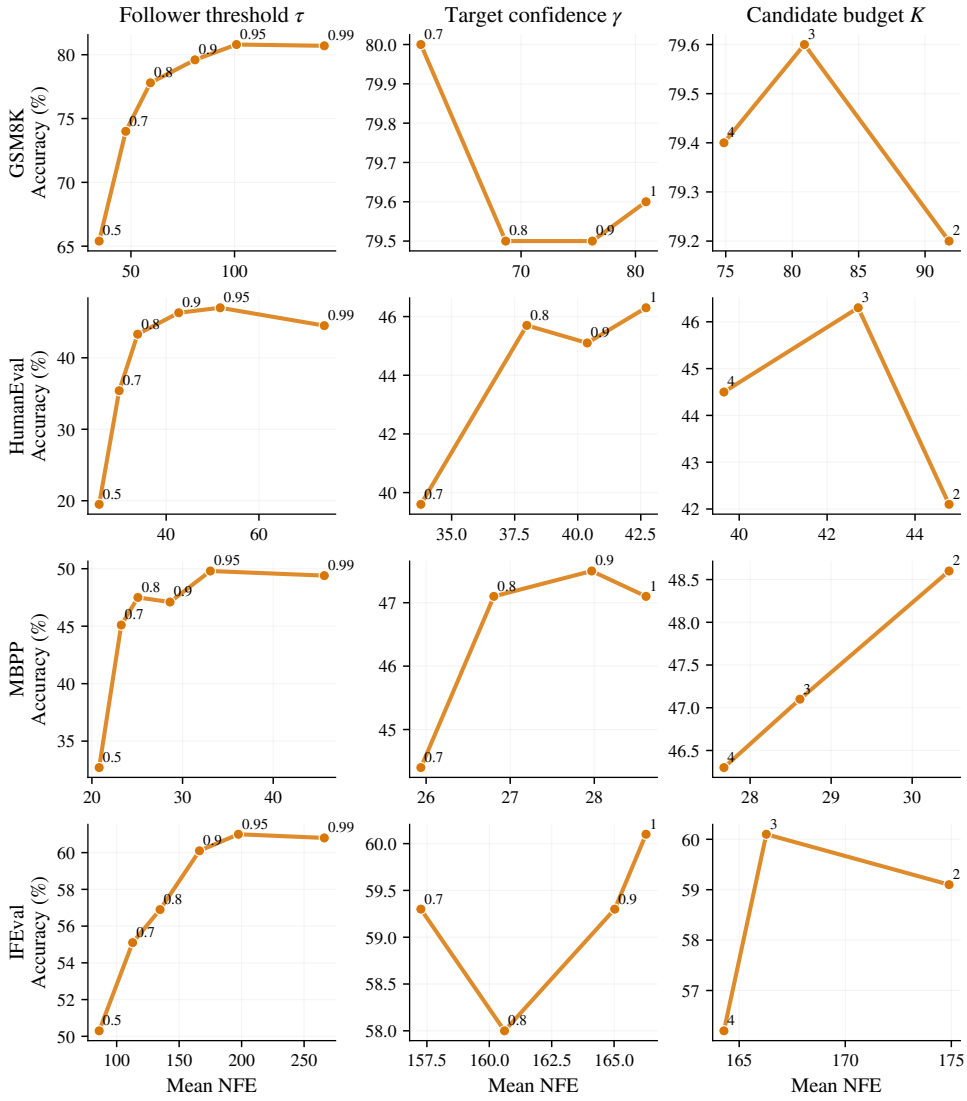

Figure 20: AnF hyperparameter sensitivity on four benchmarks with LLaDA-8B-Instruct. Each column varies one parameter: follower threshold τ with $(\gamma, K) = (1.0, 3)$, target confidence γ with $(\tau, K) = (0.9, 3)$, or candidate budget K with $(\tau, \gamma) = (0.9, 1.0)$. The window size is $W = 64$ throughout. Labels mark parameter values, and lines connect settings in parameter order. Each panel shows one task.

Broader sweeps. We also evaluate $\gamma \in \{0.7, 0.8, 0.9, 1.0\}$ and $K \in \{2, 3, 4\}$ for LLaDA, and $\gamma \in \{0.9, 1.0\}$ with $K = 3$ for LLaDA-1.5 and Dream. The highest-accuracy operating points in Appendix F.4 are selected from these broader sweeps, not only from $\gamma = 1.0$, $K = 3$.

E.1 Sensitivity to Decoding Parameters

We examine the follower threshold τ, target confidence γ, and candidate budget K on LLaDA-8B-Instruct. The window size is $W = 64$ throughout. The γ and K ablations fix $\tau = 0.9$ and vary one parameter at a time. The follower threshold sweep fixes $(\gamma, K) = (1.0, 3)$. The evaluation uses 1,319 GSM8K, 164 HumanEval, 257 MBPP, and 541 IFEval examples at the generation lengths above. Figure 20 shows the accuracy–NFE curves of all three sweeps on the same examples. Point labels give the varied parameter value.

Higher target confidence increases NFE. Increasing γ increases mean NFE on all four tasks at the fixed follower threshold. Accuracy does not improve monotonically. HumanEval accuracy increases from 39.6% at $\gamma = 0.7$ to 46.3% at $\gamma = 1.0$, whereas GSM8K accuracy stays between 79.5% and 80.0%. Thus, increasing target confidence raises mean NFE, while its effect on accuracy varies across tasks.

More anchor candidates lower NFE. We fix $\tau = 0.9$ and $W = 64$ and vary the candidate budget over $K \in \{2, 3, 4\}$ within the same decoder. Increasing K from 2 to 4 lowers mean NFE on all four tasks, from 44.8 to 39.7 on HumanEval and from 91.8 to 74.9 on GSM8K. $K = 3$ gives the highest accuracy on HumanEval (46.3%) and IFEval (60.1%), while GSM8K accuracy stays between 79.2% and 79.6%. MBPP accuracy decreases from 48.6% to 46.3% as K increases from 2 to 4. Thus, the best candidate budget depends on the task.

F Additional Generation Benchmark Results

This appendix supplements the LLaDA-8B-Instruct comparison in Section 6.1 with additional models and generation lengths. Decoding configurations are given in Appendix E.

F.1 Additional Results on LLaDA-1.5

Figure 21 shows the accuracy–NFE trade-offs on LLaDA-1.5. We evaluate AnF with candidate budget $K = 3$, target confidence $\gamma \in \{0.9, 1.0\}$, and six values of the follower threshold. Its highest accuracies are 78.5% on GSM8K, 46.3% on HumanEval, 49.8% on MBPP, and 63.6% on IFEval, at 78.2, 79.2, 29.1, and 211.9 mean NFE, respectively. These are the highest-accuracy settings within the evaluated parameter grid; they are not selected to match the computational budgets of other methods.

F.2 Additional Results on Dream-Instruct-7B

Figure 22 shows the accuracy–NFE trade-offs on Dream-Instruct-7B. We evaluate AnF with candidate budget $K = 3$, target confidence $\gamma \in \{0.9, 1.0\}$, and six values of the follower threshold. Its highest accuracies are 50.0% on HumanEval, 65.8% on MBPP, 84.8% on GSM8K, and 60.8% on IFEval, at 31.6, 37.6, 65.9, and 206.8 mean NFE, respectively (Table 9). These are the highest-accuracy settings within the evaluated parameter grid; they are not selected to match the computational budgets of other methods.

F.3 Generation-Length Comparison

We examine how accuracy changes with generation length. Table 7 reports accuracy on four generation benchmarks at lengths 256, 512, and 1,024. We evaluate on 1,319 GSM8K, 164 HumanEval, 257 MBPP, and 541 IFEval problems, reporting prompt-level loose accuracy for IFEval. For each method, we select the setting with the highest accuracy at length 512, breaking ties by lower NFE, and use it at all three lengths. AnF uses $(\gamma, K, \tau) = (1, 3, 0.95)$ on all four benchmarks.

F.4 Highest-Accuracy Settings by Model and Task

Figure 6 shows the accuracy–NFE trade-offs on LLaDA-8B-Instruct. Tables 8–9 report each method's highest accuracy and the corresponding mean NFE on each task for LLaDA-1.5 and Dream. For each method, settings are selected separately for each task, with ties broken by lower NFE. The tables summarize the best results within each method's parameter sweep; they do not compare matched computational budgets or use a single configuration across tasks. AnF uses entropy reduction to select an anchor from the anchor candidates; the tables omit other anchor-selection variants. Bold marks the best accuracy for each model and task. For IFEval, we report prompt-level loose accuracy. NFE counts physical model forward calls; it does not measure latency or FLOPs.

Additional models. Table 8 reports LLaDA-1.5 results, with follower threshold sweeps shown in Appendix F.1. Table 9 reports Dream-Instruct-7B results, with follower threshold sweeps shown in

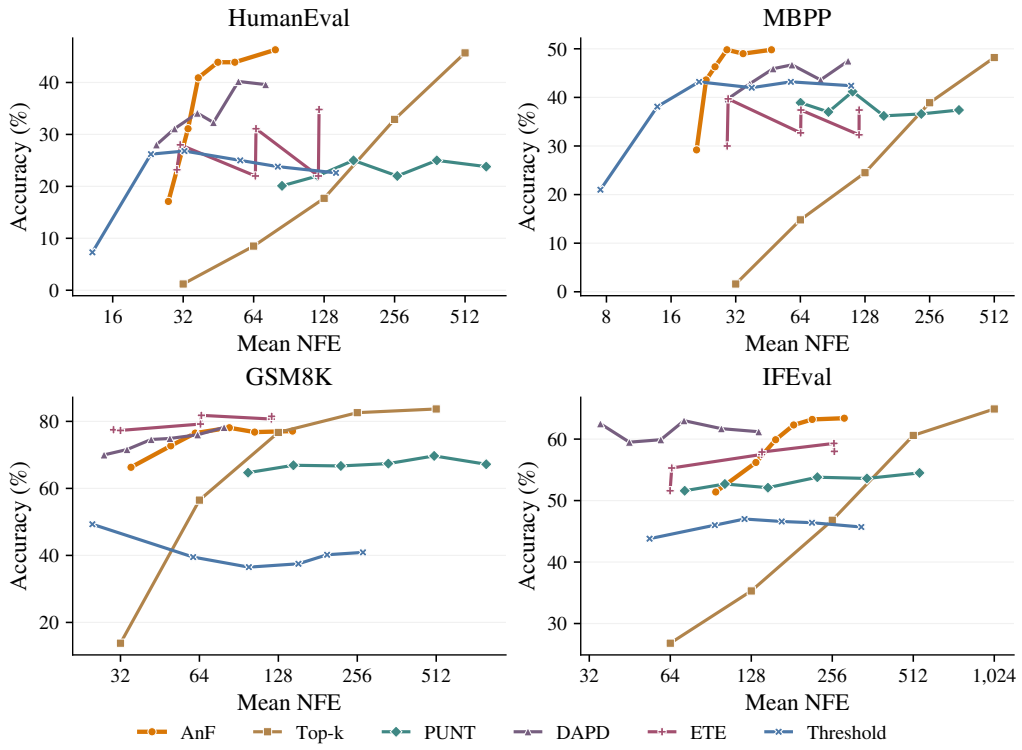

Figure 21: **Accuracy–NFE trade-offs on LLaDA-1.5.** Panels show measured results for AnF, top-k decoding, PUNT, DAPD, ETE, and threshold decoding on four benchmarks. The AnF curve fixes $(\gamma, K) = (1, 3)$ and connects results in order of the six follower threshold values. The other curves connect results from each method's parameter sweep. The NFE axis is logarithmic. NFE counts physical model forward calls; it does not measure latency or FLOPs.

Table 7: **Accuracy across generation lengths on LLaDA-8B-Instruct.** Entries give complete-answer accuracy (%) on the full benchmarks, with prompt-level loose accuracy for IFEval. Each method uses one parameter setting across lengths. Bold marks the highest accuracy in each column.

	HumanEval			MBPP			GSM8K			IFEval		
Method / Length	256	512	1024	256	512	1024	256	512	1024	256	512	1024
Fast-dLLM	**38.4**	37.8	38.4	42.8	40.5	40.9	**79.2**	**84.1**	83.1	53.4	53.6	53.4
PUNT	26.8	21.3	18.3	41.2	41.2	37.4	59.4	64.4	61.9	—	54.0	49.7
DAPD	32.3	40.2	39.0	45.5	48.2	**47.1**	51.9	76.6	76.8	61.0	60.3	59.5
ETE	29.9	36.6	36.0	41.6	40.1	38.5	77.9	81.7	81.4	50.5	50.3	53.4
AnF	36.6	**47.0**	**43.3**	**49.8**	**49.8**	45.9	61.6	80.8	**84.3**	61.0	**61.2**	**61.4**

Appendix F.2. On LLaDA-1.5, AnF reaches 46.3% on HumanEval at 79.2 mean NFE. On Dream-Instruct-7B, AnF reaches 84.8% on GSM8K, 50.0% on HumanEval, 65.8% on MBPP, and 60.8% on IFEval, at 65.9, 31.6, 37.6, and 206.8 mean NFE, respectively. Each operating point is selected independently from the parameter sweeps described in Appendix E.

The Dream-Instruct-7B evaluation includes 1,319 GSM8K, 164 HumanEval, 257 MBPP, and 541 IFEval problems. Generation length is 512 for the first three tasks and 1,024 for IFEval. AnF uses $K = 3$, target confidence $\gamma \in \{0.9, 1.0\}$, and follower thresholds in $\{0.5, 0.7, 0.8, 0.9, 0.95, 0.99\}$. For Dream, the DAPD sweep includes an additional task-specific threshold pair, giving seven settings per task. As in the other tables, each entry is the highest accuracy within the method's sweep.

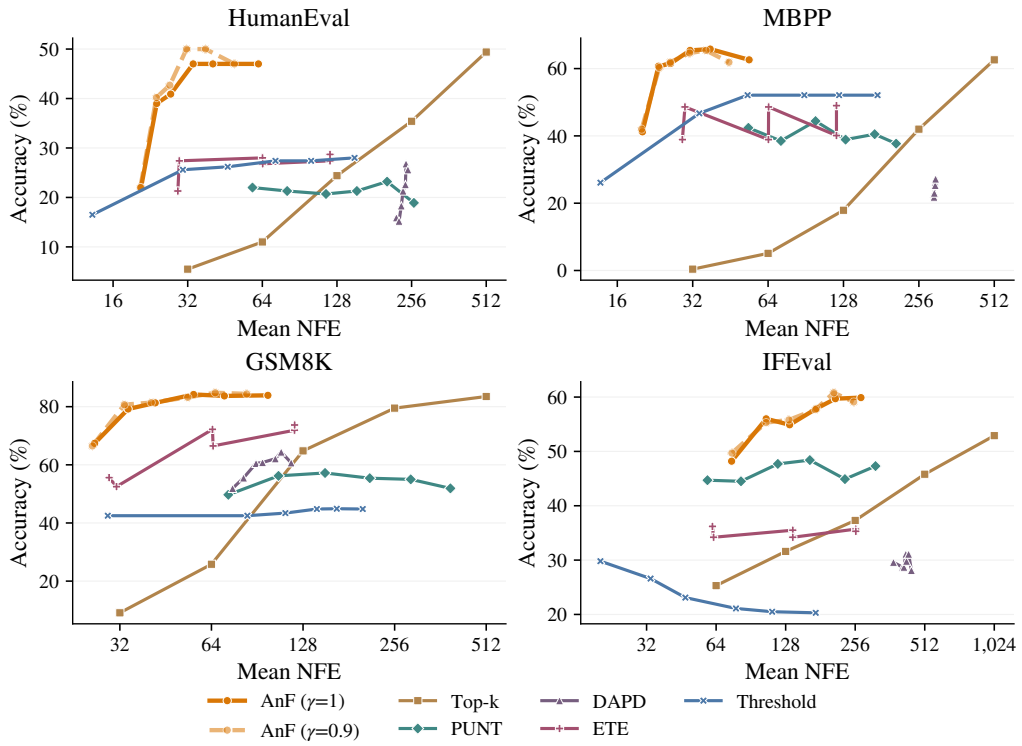

Figure 22: **Accuracy–NFE trade-offs on Dream-Instruct-7B.** Panels show measured results for AnF, top-k decoding, PUNT, DAPD, ETE, and threshold decoding on four benchmarks. Each AnF curve uses $K = 3$ and one fixed value of $\gamma \in \{1, 0.9\}$, connecting results in order of the six follower threshold values. The other curves connect results from each method's parameter sweep. The NFE axis is logarithmic. NFE counts physical model forward calls; it does not measure latency or FLOPs.

Table 8: **LLaDA-1.5: highest-accuracy operating points.** Accuracy (Acc.) is in percent, and NFE is the mean per problem.

	HumanEval		MBPP		GSM8K		IFEval	
Method	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE
Threshold	26.8	32.4	43.2	21.6	49.3	25.0	47.0	120.8
PUNT	25.0	170.9	41.2	112.1	69.7	502.9	54.5	540.0
DAPD	40.2	55.1	47.5	106.8	78.2	79.5	63.0	72.0
Top-k	45.7	512.0	48.2	512.0	**83.7**	512.0	**64.9**	1024.0
ETE	34.8	121.8	40.1	29.9	81.9	121.0	59.3	259.7
AnF	**46.3**	79.2	**49.8**	29.1	78.5	78.2	63.6	211.9

G Additional Prediction Analyses

This appendix supplements Section 4 with comparisons of model predictions across decoding methods on the same problems. To analyze the effects of selected anchors, we use AnF with $(\gamma, K, \tau) = (1, 3, 0.9)$ on 100 Graph, 100 Replace Random, 100 Shuffle, and 83 Latin Square items. We exclude 17 Latin Square items whose token sequences cannot be aligned with the structural units. In 9 of these items the output has the wrong number of structural units, and in 8 the saved output and the retokenized layout disagree. For Graph, we compute valid answer uncertainty N_t using the occurrence-count convention in Appendix A. Unless otherwise noted, analyses of changes in N_t include only pairs with finite values at both endpoints. States with zero answer contribution mass have undefined N_t and are not counted as finite observations.

Table 9: **Dream-Instruct-7B: highest-accuracy operating points.** Accuracy (Acc.) is in percent, and NFE is the mean per problem.

	HumanEval		MBPP		GSM8K		IFEval	
Method	Acc.	NFE	Acc.	NFE	Acc.	NFE	Acc.	NFE
Threshold	28.0	150.7	52.1	53.0	44.9	165.2	29.8	20.2
PUNT	23.2	203.9	44.4	99.0	57.2	151.2	48.4	162.9
DAPD	26.8	243.1	27.2	298.5	64.4	108.5	31.1	424.1
Top-k	49.4	512.0	62.6	512.0	83.5	512.0	52.9	1024.0
ETE	29.9	64.3	49.0	120.0	73.7	120.1	36.4	257.1
AnF	**50.0**	31.6	**65.8**	37.6	**84.8**	65.9	**60.8**	206.8

G.1 Answer uncertainty across decoding methods

We compare denoising trajectories under AnF with $(\gamma, K, \tau) = (1, 3, 0.9)$, threshold decoding with a confidence threshold of 0.9, and top-1, top-2, and top-4 decoding. Within each task, we evaluate all methods on the same problems and use the same set of valid answers for each problem. We compute N_t exactly using the task-specific procedure. Figure 23 plots the mean of $\log N_t$ over finite states against the fraction of committed output positions. For each item, we hold N_t constant between observed commitment points and omit undefined values from the mean. For Latin Square, we keep only items for which the five outputs yield the same structural token layout. This ensures that differences at the same initial state do not arise from method-dependent token boundaries. Fewer items have finite N_t near the end of some trajectories. For example, at the final top-4 state, only 19 of 100 Shuffle items and 6 of 14 Latin Square items have finite N_t, so the endpoint means describe only these subsets.

G.2 Anchor Effects and Final-Answer Correctness

Increases in answer uncertainty are more frequent in trajectories ending in incorrect answers. Figure 24 relates each selected anchor's effect on answer uncertainty to the correctness of the final answer. For each selected anchor with finite N_t before and after anchor-conditioned refinement, we record whether N_t decreases or increases. We then group the transitions by whether the item's final answer is correct. On Graph, all 131 transitions in correctly answered items decrease N_t, whereas 29 of 82 transitions (35.4%) in incorrectly answered items increase it. On Latin Square, increases account for 103 of 368 transitions (28.0%) in correctly answered items and 196 of 319 (61.4%) in incorrectly answered items. Replace Random and Shuffle have only 3 and 8 transitions, respectively, in incorrectly answered items, which is too few for a meaningful comparison. The Latin Square cohort here (83 items) is larger than the cohort in Appendix D (17 items), because this analysis does not require the structural units to occupy the same token spans across all enumerated valid answers.

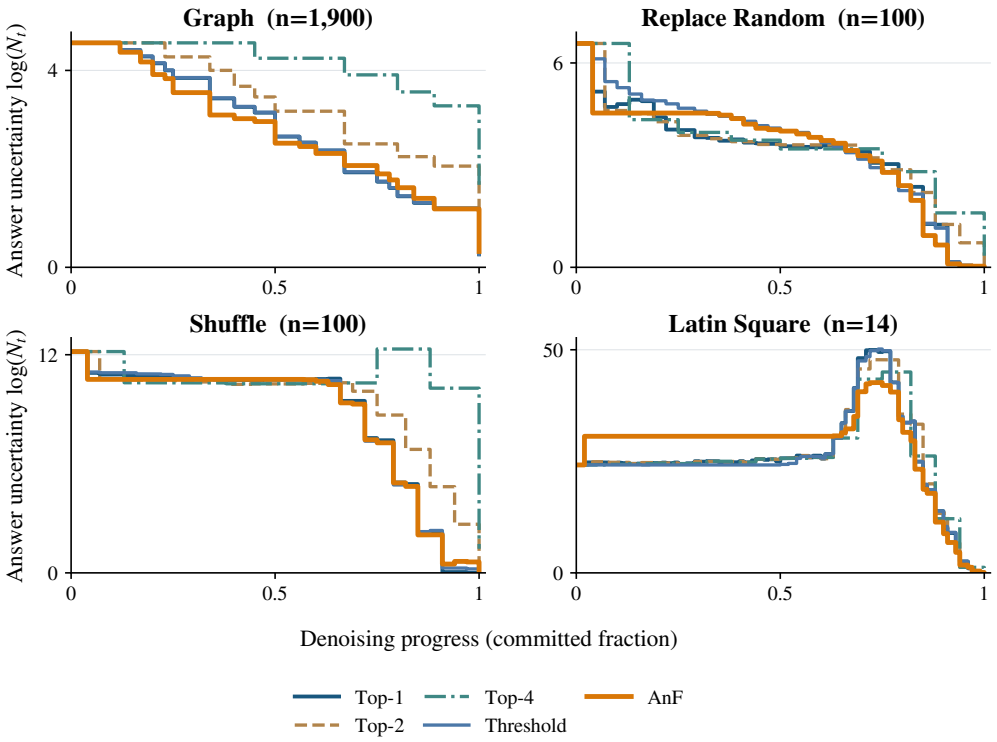

Figure 23: **Answer uncertainty during denoising under five decoding methods.** Curves show the mean of finite $\log N_t$ values over matched problems on the four controlled tasks, and panel titles give the number of included problems. Trajectories are aligned by committed fraction, and results are reported separately for each task. Latin Square includes only items with identical structural token layouts across methods.

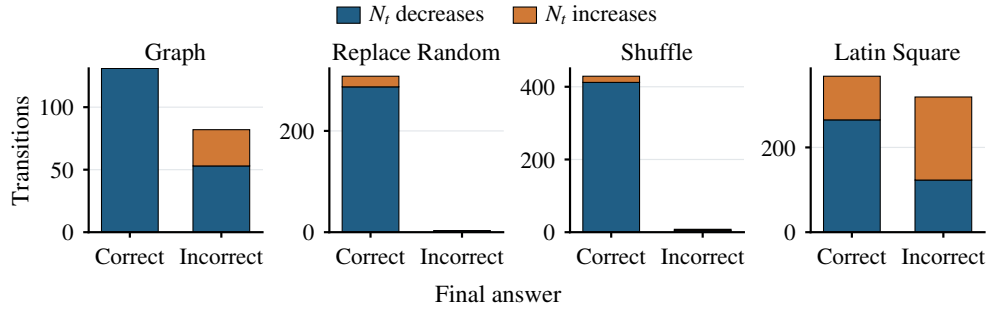

Figure 24: **Anchor effects on answer uncertainty by final correctness.** Bars count selected-anchor transitions with finite N_t before and after refinement, grouped by the direction of change and whether the final answer is correct. The groups differ in size. The plot describes an association and does not establish that changes in N_t cause changes in final-answer correctness.