FEROM: Frontier Endogenous Reveal-Order Marginal Policy Optimization for Masked Diffusion LMs

Zian Su
Purdue University
su284@purdue.edu

Ziyang Huang
Johns Hopkins University
zhuang86@jhu.edu

Kaiyuan Zhang
Rutgers University
kaiyuan.z@rutgers.edu

Xiangyu Zhang
Purdue University
xyzhang@cs.purdue.edu

Abstract

Masked diffusion language models generate text by iteratively unmasking positions, with practical samplers often selecting reveal positions from the model's own logits. The reveal order is therefore an endogenous latent variable of the rollout process with an enormous discrete latent space. Policy optimization objectives relying on pre-defined schedules or sampled trajectories introduce either bias or high variance. In this paper, we propose FEROM, *Frontier Endogenous Reveal-Order Marginal Policy Optimization*, which approximates policy gradients for the rollout-induced marginal response policy of masked diffusion LMs. FEROM derives a Rao–Blackwellized policy-gradient identity over latent reveal orders and expresses the resulting estimator as posterior edge occupancy on a reveal-state DAG. To make marginalization practical, we introduce Frontier Reveal Marginalization, a budgeted estimator that approximates this marginal edge structure by combining CDF-based computation of local probabilities of the endogenous reveal process with frontier expansion of high-mass partial states. Integrated into a GRPO-style objective, FEROM replaces single-path log-scores with a detached token-score surrogate over posterior-relevant reveal edges. Experiments on math and coding tasks show comparable or improved results over existing methods under matched scoring-state budgets. An offline proxy study further shows improved marginal-gradient approximation with increased scoring budgets.[1]

1 Introduction

Diffusion language models (dLLMs) [27, 12] and, in particular, *masked diffusion language models* (MDMs) [42, 40] have recently emerged as promising alternatives to left-to-right autoregressive (AR) language models. Instead of committing to a fixed causal factorization, an MDM generates text by iteratively refining a partially masked sequence. This decoding paradigm naturally supports bidirectional conditioning, adaptive reveal orders, and partially parallel token updates, leading to attractive latency—parallelism trade-offs at inference time [32, 51].

This flexibility, however, creates a distinctive challenge for reinforcement learning. In an AR model, a sampled response comes with a unique token-generation trajectory, so the rollout policy and the policy log-probability used in policy-gradient updates are naturally aligned. In an MDM, generation proceeds through a sequence of partially revealed states, and the sampler must repeatedly decide which masked positions to reveal next. In practical systems, these reveal decisions are often computed

[1] Code is available at: https://github.com/ziansu/ferom.

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

from the model's own logits, for example, through confidence or margins [32, 24]. Thus, the reveal order is not merely an external schedule: it is an *endogenous* latent process coupled to token prediction.

On the other hand, sequence-level rewards evaluate only the completed response, not the internal reveal trace used to produce it. The same response may be reachable through many plausible reveal orders, all equivalent from the reward function's perspective. A trajectory-level policy-gradient estimator that conditions on the single sampled reveal path is valid for the joint rollout process, but it attributes reward to latent reveal-order randomness that the reward never observes.

In this paper, we argue that the statistically appropriate object for sequence-level RL is instead the *rollout-induced marginal response policy*: the distribution over completed responses after integrating out reveal orders. Its policy gradient averages trajectory-level contributions over reveal traces consistent with the same completed response, a Rao–Blackwellization that removes nuisance pathwise variance. The difficulty is computational. Exact marginalization over endogenous reveal orders is combinatorial in sequence length, while MDM RL is already constrained by expensive transformer forwards, long denoising horizons, and limited GPU memory. This raises the central question of this paper: *how can we estimate the policy gradient of the rollout-induced marginal response policy under a realistic compute and memory budget?*

We propose `FEROM`, *Frontier Endogenous Reveal-Order Marginal Policy Optimization*, a policy-gradient framework for masked diffusion LMs under their actual rollout process. `FEROM` targets the rollout-induced marginal response policy: it first Rao–Blackwellizes trajectory policy gradients over latent reveal orders, then represents reveal trajectories as paths in a reveal-state DAG, where posterior edge occupancies weight each reveal transition's contribution to the marginal gradient. Since exact occupancy computation is combinatorial, `FEROM` introduces Frontier Reveal Marginalization (FRM). FRM approximates the marginal edge structure by maintaining a frontier of high-occupancy partial states, computing target-conditioned local probabilities of the endogenous reveal process exactly from confidence-score cumulative distribution functions (CDFs), and expanding posterior-relevant reveal edges under a fixed scoring-state budget. The resulting `FEROM`-GRPO objective uses this edge structure to replace the log-score of a single sampled reveal path with a detached token-score surrogate over posterior-relevant reveal edges.

We evaluate `FEROM` on mathematical reasoning and code generation using GSM8K [10], MATH-500 [28], HumanEval [6], and MBPP [3], with LLaDA-Instruct [32] as the base model. At the 256-token training length, `FEROM` leads the compared small-budget baselines on MATH-500 and HumanEval and ranks second on GSM8K. Our offline study further shows that larger scoring budgets allow `FEROM` to better approximate the block-level marginal gradient than single-trajectory variants.

Our contributions are fourfold. First, we formulate MDM reinforcement learning as optimizing the rollout-induced marginal response policy, with reveal orders treated as endogenous latent variables generated by the model's logits. Second, we derive a Rao–Blackwellized marginal policy-gradient identity and characterize its estimator as posterior edge occupancy on a reveal-state DAG. Third, we introduce `FEROM`, a budgeted frontier-based algorithm that approximates the token-score component of this marginal gradient under strict forward and memory constraints via exact local probabilities of the endogenous reveal process and budgeted edge coverage. Fourth, we instantiate `FEROM` in a GRPO-style objective and demonstrate benefits in low-budget training and block-level marginal-gradient approximation.

2 Preliminaries

In this section, we first review masked diffusion model basics and then introduce an alternative endogenous reveal-process view that we advocate throughout this paper.

Notation. Let $x^{1:n}$ denote a length-n token sequence, with $x^i \in \mathcal{V}$ the token at position i, and let $[n] \triangleq \{1, \dots, n\}$. $\mathbf{e}_{x^i}$ is the one-hot vector of token x^i, and $\mathbf{e}_{\texttt{[M]}}$ is the one-hot vector of the mask token. For any subset $S \subseteq [n]$, let $\tilde{x}_S$ denote a partially revealed sequence, also referred to as a state, where S is the set of currently revealed positions and the remaining positions are all masks. We use $\mathrm{Cat}(\cdot; p)$ for the categorical distribution with parameter $p \in \Delta(\mathcal{V} \cup \{\texttt{[M]}\})$.

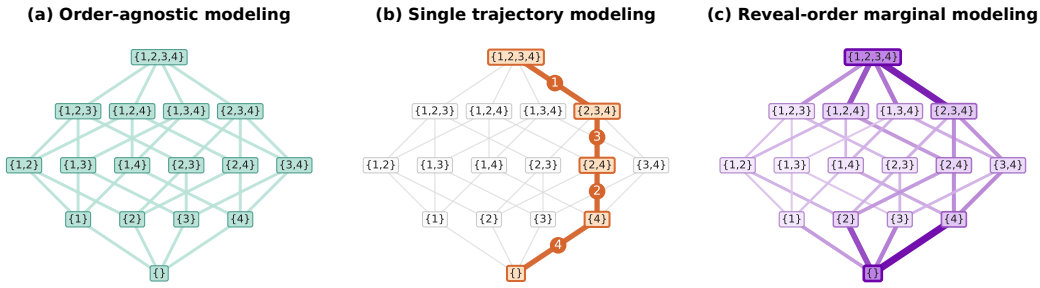

Figure 1: **Reveal-order modeling on an example 4-position reveal-state DAG.** Nodes show the still-masked sets $[n] \setminus S$, where S is the revealed set. Each edge reveals one position. Node color denotes state occupancy, and edge color/width denotes reveal-edge occupancy. **(a)** Order-agnostic denoising uses training states from an exogenous masking process rather than the adaptive rollout distribution. **(b)** Single-trajectory modeling follows one sampled reveal path, assigning credit only to visited edges and retaining latent reveal-order variance. **(c)** `FEROM` aggregates reveal-order contributions using posterior edge occupancies on the DAG.

Forward process. For a clean sequence $x_0 \sim p_{\text{data}}$ and noise level $t \sim \text{Unif}[0, 1]$, the forward corruption process *independently* masks each token according to $q_{t|0}(x_t^{1:n} \mid x_0^{1:n}) = \prod_{i=1}^{n} q_{t|0}(x_t^i \mid x_0^i)$, with

$$q_{t|0}(x_t^i \mid x_0^i) = \text{Cat}\Big(x_t^i;\ \alpha_t \mathbf{e}_{x_0^i} + (1-\alpha_t)\mathbf{e}_{\texttt{[M]}}\Big),$$

where α_t is a predefined noise schedule, commonly chosen as $\alpha_t = 1 - t$.

Reverse process and training. The corresponding reverse process $q_{s|t}(x_s^{1:n}|x_t^{1:n}, x_0^{1:n}) = \prod_{i=1}^{n} q_{s|t}(x_s^i|x_t^i, x_0^i)$, where $s < t$, reverses the forward corruption. A time-agnostic neural network $p_\theta(x_0^i|x_t^{1:n})$ approximates this process by predicting the clean token x_0^i given the randomly masked state $x_t^{1:n}$. Training minimizes the following negative evidence lower bound (NELBO) [54, 42, 39]:

$$\mathcal{L}_\theta = \int_0^1 \omega(t)\, \mathbb{E}_{\substack{x_0 \sim p_{\text{data}}, \\ x_t \sim q_{t|0}(\cdot|x_0)}} \left[\sum_{i: x_t^i = \texttt{[M]}} -\log p_\theta(x_0^i | x_t^{1:n}) \right] \mathrm{d}t \tag{1}$$

where $\omega(t) = \frac{-\alpha_t'}{1-\alpha_t}$ and $\alpha_t' = \frac{\mathrm{d}\alpha_t}{\mathrm{d}t} < 0$. The standard MDM objective samples states from the prescribed forward corruption process. In RL, using this objective as a reward-weighted surrogate optimizes a masking-state distribution that may differ from the rollout distribution [49, 23, 45].

Endogenous reveal process. At inference, an MDM alternates between token prediction and reveal-position selection [39, 32, 4]. A reveal rule is *endogenous* when it selects positions using the same logits as token prediction. LLaDA-style decoding keeps revealed tokens fixed and resamples candidates from current-state predictions. Hence the current state determines the next-update distribution for a fixed context and decoder configuration. For a one-token update,[2] let Z_j be the candidate at masked position j and I the selected position. We use p_θ for the candidate-token distribution at the rollout temperature. For a sequence x, define $r_\theta^x(i \mid \tilde{x}_S) \triangleq \Pr_\theta(I = i \mid \tilde{x}_S, Z_i = x^i)$. Only Z_i is fixed; competing candidates remain random. For a reveal order $\sigma = (\sigma_1, \ldots, \sigma_n)$, let $S_0 = \emptyset$ and $S_t = \{\sigma_1, \ldots, \sigma_t\}$. The chain rule gives the joint distribution

$$P_\theta(x, \sigma \mid \tilde{x}_{S_0}) = \prod_{t=0}^{n-1} r_\theta^x(\sigma_{t+1} \mid \tilde{x}_{S_t})\, p_\theta(x^{\sigma_{t+1}} \mid \tilde{x}_{S_t}). \tag{2}$$

Remark 2.1. Classical order-agnostic AR models marginalize over an exogenous order distribution, often uniform over permutations [46, 18, 2, 54, 34]. In contrast, an MDM's endogenous reveal process induces a state-dependent reveal distribution from the model's own predictions, so the latent order is part of the sampler whose marginal response policy is optimized.

[2] Parallel multi-token updates use a joint subset-selection probability, derived in Appendix C.

3 FEROM: Frontier Endogenous Reveal-Order Marginal Policy Optimization

We introduce FEROM through its marginal-gradient and reveal-state formulation (§3.1–3.2), budgeted frontier approximation (§3.3), and resulting GRPO-style objective (§3.4).

3.1 Theoretical Foundations: Marginalizing Latent Reveal Traces

We formalize reinforcement learning for MDMs under the endogenous reveal-process view. Let $c \sim \rho$ be a conditioning context drawn from the prompt distribution ρ, and x a generated sequence. Given a sequence-level reward $R(c, x)$, the RL objective is

$$J(\theta) = \mathbb{E}_{c\sim\rho, x\sim\pi_\theta(\cdot|c)}[R(c, x)].$$

At first glance, a trajectory-level policy-gradient estimator that samples a full reveal trace $(x, \sigma) \sim P_\theta(\cdot, \cdot \mid c)$ and uses

$$g_{\text{traj}}(x, \sigma, c) = R(c, x)\nabla \log P_\theta(x, \sigma \mid c)$$

is most straightforward since it is on-policy for the joint rollout process. However, this trajectory-level estimator conditions on the particular reveal trace sampled during rollout. Since for language models the reward $R(c, x)$ depends only on the completed sequence x and the context c, the reveal order is a nuisance latent variable for sequence-level credit assignment. Let $\pi_\theta(x \mid c) = \sum_\sigma P_\theta(x, \sigma \mid c)$ denote the reveal-order-marginal policy induced by the rollout process. Differentiating the marginal policy yields

$$\nabla \log \pi_\theta(x \mid c) = \mathbb{E}_{\sigma\sim P_\theta(\sigma|c,x)}[\nabla \log P_\theta(x, \sigma \mid c)] .$$

The sequence-level score is the posterior average of trajectory scores for the same sequence, an instance of Fisher's identity. Substituting it into the policy-gradient objective gives

$$\begin{aligned} \nabla J(\theta) &= \mathbb{E}_{\substack{c\sim\rho,\\ x\sim\pi_\theta(\cdot|c)}} [R(c, x)\nabla \log \pi_\theta(x \mid c)] \\ &= \mathbb{E}_{\substack{c\sim\rho,\\ x\sim\pi_\theta(\cdot|c)}} \left[R(c, x)\, \mathbb{E}_{\sigma\sim P_\theta(\sigma|c,x)}[\nabla \log P_\theta(x, \sigma \mid c)]\right] . \end{aligned} \tag{3}$$

This is the marginal policy gradient targeted by FEROM.

Theorem 3.1 (Rao-Blackwell dominance of marginal policy gradients)**.** *Assume that the reward depends only on the completed sequence x and context c. Assume also that differentiation can be interchanged with the trajectory sum and that the scores have finite second moments. Define*

$$g_{\text{traj}}(x, \sigma, c) = R(c, x)\nabla \log P_\theta(x, \sigma \mid c), \qquad g_{\text{marg}}(x, c) = R(c, x)\nabla \log \pi_\theta(x \mid c).$$

Then, (1) $\mathbb{E}[g_{\text{traj}}] = \mathbb{E}[g_{\text{marg}}] = \nabla J(\theta)$*; (2)* $g_{\text{marg}}(x, c) = \mathbb{E}[g_{\text{traj}}(x, \sigma, c) \mid x, c]$*; (3)* $\mathrm{Cov}(g_{\text{marg}}) \preceq \mathrm{Cov}(g_{\text{traj}})$*; (4) equality in (3) holds iff* $\mathrm{Var}(g_{\text{traj}} \mid x, c) = 0$ *almost surely.*

Theorem 3.1 motivates averaging over reveal orders: exact posterior averaging preserves the expected policy gradient and reduces reveal-order variance (proof in Appendix B).

> **Takeaway.** The reward observes the completed sequence, not its internal reveal trace. Averaging over traces consistent with that sequence removes this source of gradient variance.

3.2 Reveal-State DAG Representation and Posterior Edge Occupancy

Theorem 3.1 motivates averaging over reveal trajectories, but computing this average directly requires considering combinatorially many reveal orders. To develop a budgeted approximation, we first rewrite the marginal score on a *reveal-state directed acyclic graph (DAG)*. This representation lets us collect contributions from different orders through the transitions they share.

Reveal-state DAG. We define a DAG given a context c and a completed sequence x. Here, each node is a revealed set $S \subseteq [n]$, and each edge $e = (S, i), i \notin S$, reveals position i and moves from S to $S \cup \{i\}$. A complete path from $\emptyset$ to $[n]$ corresponds to one reveal order. For an order $\sigma = (\sigma_1, \ldots, \sigma_n)$, write $S_t(\sigma) = \{\sigma_1, \ldots, \sigma_t\}$ for the revealed set after t steps, with $S_0(\sigma) = \emptyset$. The next edge has $S = S_t(\sigma)$ and $i = \sigma_{t+1}$, so the trajectory's traversed edge set is $\mathcal{E}(\sigma) = \{(S_t(\sigma), \sigma_{t+1}) : t = 0, \ldots, n-1\}$. Orders that reach the same revealed set share the same partially revealed sequence and its possible continuations. We illustrate how different reveal-order modeling approaches use this structure in Figure 1 and provide a concrete example in Appendix E.

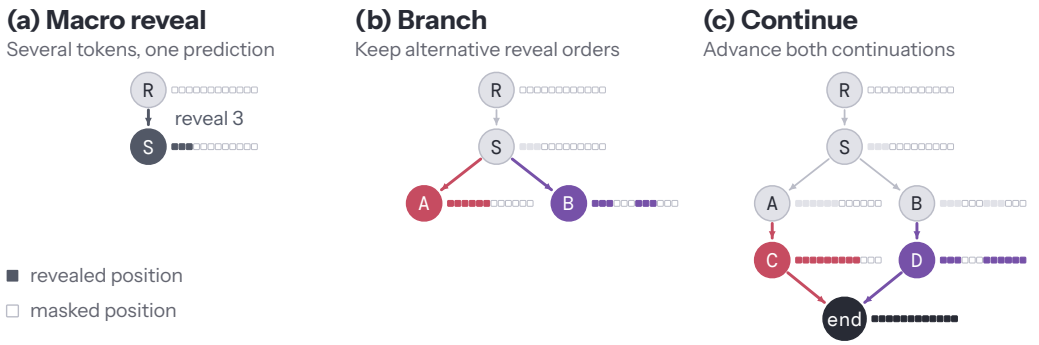

Figure 2: **Macro reveals and branching in FRM.** FRM progressively expands the reveal-state DAG for a given sequence: **(a)** a macro reveal scores several target tokens from one parent prediction; **(b)** branching creates two continuations with different leading reveals; **(c)** further macro reveals advance both continuations to the same completed sequence. Filled cells mark revealed positions; colored arrows show newly added transitions.

Posterior edge occupancy. The reveal orders consistent with x are weighted by their posterior $P_\theta(\sigma \mid c, x)$. Their combined probability of traversing an edge e defines its *posterior edge occupancy*:

$$\mu^\star_{\theta,e}(c, x) = \Pr_{\sigma \sim P_\theta(\sigma \mid c,x)} [e \in \mathcal{E}(\sigma)] .$$

Since each trajectory score is a sum of local edge scores, averaging over trajectories weights each edge by its posterior occupancy. For differentiable local edge factors, this gives the *exact marginal score*:

$$\nabla_\theta \log \pi_\theta(x \mid c) = \sum_{e=(S,i)} \mu^\star_{\theta,e}(c, x) \nabla_\theta \log \Big[r^x_\theta(i \mid c, \tilde{x}_S) p_\theta(x^i \mid c, \tilde{x}_S) \Big]. \tag{4}$$

Equation 4 averages both token-prediction and reveal-decision scores over posterior edge occupancies. Appendix D derives these occupancies and the identity.

Local reveal probabilities for LLaDA decoding. The selection factor r^x_θ in Eq. 4 depends on how the decoder chooses among sampled candidates. For a one-token LLaDA-style update, let $\mathcal{I}(S)$ be the eligible masked positions: $[n] \setminus S$, restricted to the active block for blockwise decoding. At position j, logits $\mathbf{a}_{\theta,j} = \mathrm{logits}_\theta(c, \tilde{x}_S)_j$ give the candidate distribution $p_{\theta,j} = \mathrm{softmax}(\mathbf{a}_{\theta,j}/\tau)$ at temperature $\tau > 0$. The decoder independently samples $Z_j \sim p_{\theta,j}$ at each eligible position and reveals the candidate with the highest untempered confidence $\mathrm{softmax}(\mathbf{a}_{\theta,j})_{Z_j}$.

For $i \in \mathcal{I}(S)$, condition on $Z_i = x^i$. The selection probability $r^x_\theta(i \mid c, \tilde{x}_S)$ is the probability that no competing candidate outranks this fixed target. The confidence cumulative distribution functions (CDFs) $F^{\leq}_{\theta,j}(u)$ and $F^{<}_{\theta,j}(u)$ give the probabilities of confidence at most u and strictly below u, respectively. Independence gives the following product, with lower position indices winning ties:

$$r^x_\theta(i \mid c, \tilde{x}_S) = \prod_{\substack{j \in \mathcal{I}(S) \\ j<i}} F^{<}_{\theta,j}(\mathrm{softmax}(\mathbf{a}_{\theta,i})_{x^i}) \prod_{\substack{j \in \mathcal{I}(S) \\ j>i}} F^{\leq}_{\theta,j}(\mathrm{softmax}(\mathbf{a}_{\theta,i})_{x^i}) . \tag{5}$$

Each CDF is a vocabulary sum of known candidate probabilities, requiring no sampling or additional model forward. The values r^x_θ need not sum to one over positions because each conditions on a different target event. Appendix C gives the k-token generalization and computational details.

Algorithm 1 Frontier Reveal Marginalization (FRM)

Require: context c, sequence x, initial state S_0, model p_θ, state-evaluation budget B
$\mathcal{F} \leftarrow \{S_0\}$, $\widehat{\nu}(S_0) \leftarrow 1$, $\mathcal{E}_B \leftarrow \emptyset$, $b \leftarrow 0$
while $b < B$ and $\mathcal{F} \neq \emptyset$ **do**
  Select and remove a continuation at S from $\mathcal{F}$
  Evaluate the model at $(c, \tilde{x}_S)$; $b \leftarrow b + 1$
  Compute $p_{\mathrm{loc},\theta}$ by Eq. 6; condition on allowed leading positions
  Choose macro size k using remaining positions and the pre-split budget, including this evaluation
  if alternatives warrant branching and the budget supports both branches **then**
    $i \leftarrow \arg\max_j p_{\mathrm{loc},\theta}(j \mid S, c, x)$
    Reserve the complementary continuation at S, excluding i; divide the future budget
  else
    $i \sim p_{\mathrm{loc},\theta}(\cdot \mid S, c, x)$; retain this continuation's remaining budget
  end if
  Draw ΔS, with $|\Delta S| = k$, conditional on leading reveal i
  $e \leftarrow (S, i, \Delta S)$; assign $\widehat{\mu}_e$ by Eq. 7
  Compute $\ell_{\theta,\Delta S}(S; c, x)$; append e with its score and coefficient to $\mathcal{E}_B$
  Propagate $\widehat{\mu}_e$ to $S \cup \Delta S$ by Eq. 8
  Update $\mathcal{F}$ with the unfinished child and reserved alternatives
end while
return $\mathcal{E}_B$ and $\{\widehat{\mu}_e\}_{e \in \mathcal{E}_B}$

3.3 Frontier Reveal Marginalization for Budgeted Estimation

Directly applying Eq. 4 requires evaluating exponentially many reveal states, with costly model forward and backward passes at each state. We introduce *Frontier Reveal Marginalization* (FRM) to couple planning with posterior occupancy estimation under limited computation, where a *frontier* of partially revealed states expands through *macro reveals* that score several tokens from one prediction and *branching* that explores alternative reveal orders with appreciable probability (Figure 2).

Local posterior approximation. At a revealed state S, the *parent state* of its outgoing transitions, FRM uses the local edge factors from §3.2 to prioritize reveal choices likely to have produced the given sequence x. Each choice must both propose the required token and select its position. Equation 5 gives the selection probability conditional on that proposal. Multiplying by the probability of proposing x^i and normalizing across eligible positions gives

$$p_{\mathrm{loc},\theta}(i \mid S, c, x) = \frac{r^x_\theta(i \mid c, \tilde{x}_S) p_\theta(x^i \mid c, \tilde{x}_S)}{\sum_{j \in \mathcal{I}(S)} r^x_\theta(j \mid c, \tilde{x}_S) p_\theta(x^j \mid c, \tilde{x}_S)}. \tag{6}$$

This distribution conditions on the next revealed token matching x. FRM uses this one-step distribution as a local approximation to the next-reveal posterior conditioned on the full sequence, guiding position selection and branching from current-state predictions (Appendix D).

Macro reveals: better coverage with fewer states. We measure the budget B in *state evaluations*. A *macro reveal* $\Delta S \subseteq \mathcal{I}(S)$ uses one evaluation at parent state S to score the target tokens in this set, with log-score $\ell_{\theta,\Delta S}(S; c, x) = \sum_{i \in \Delta S} \log p_\theta(x^i \mid c, \tilde{x}_S)$. To form the set, FRM selects a *leading position* i using $p_{\mathrm{loc},\theta}$ and fixes its candidate to x^i. It samples competing candidates conditional on i ranking first by confidence; the highest-ranked positions, including i, form ΔS. FRM records the macro-edge as $e = (S, i, \Delta S)$ and advances to $S \cup \Delta S$. Macro size adapts to the remaining masked positions and allocated state evaluations: fewer evaluations require larger reveals, while more evaluations permit finer state resolution. Appendix F specifies the proposal and size-selection rules.

Branching and frontier allocation. The frontier $\mathcal{F}$ stores unfinished continuations of reveal paths for the same sequence x. Each continuation specifies a current state S, the allowed leading positions for its next macro reveal, and a remaining evaluation budget. It also carries a nonnegative weight $\widehat{\nu}(S)$, representing its share of reveal-order probability under FRM's local approximation. Initially, the frontier contains only S_0, with weight one and all eligible leading positions allowed.

At a state S, FRM compares the most likely leading position i with the other allowed choices. If these alternatives carry appreciable probability and the remaining budget can advance both branches,

FRM splits the continuation *before sampling a macro-set*. One branch fixes i as its leading reveal; the other remains at S with i excluded from its allowed leading positions. The two branches divide the remaining state evaluations and advance through their own macro reveals. Otherwise, FRM samples one leading position and follows a single continuation. Expansion priority favors larger continuation weights and less concentrated local reveal distributions.

Occupancy estimation and contribution weights. Following the weighted-sum structure of Eq. 4, FRM assigns each scored macro-edge a contribution coefficient $\widehat{\mu}_e$. Here $p_{\mathrm{loc},\theta}$ is Eq. 6, restricted and renormalized over the continuation's allowed leading positions. Without a split, sampling from this distribution represents the alternatives through their sampling frequencies, so the sampled macro-edge inherits the full continuation weight. A split instead represents the selected leading position and its complement separately, so the selected branch receives only its probability share:

$$\widehat{\mu}_e = \begin{cases} \widehat{\nu}(S), & i \text{ sampled without a split,} \\ \widehat{\nu}(S)\, p_{\mathrm{loc},\theta}(i \mid S, c, x), & i \text{ fixed by a split.} \end{cases} \tag{7}$$

On a split, the complementary continuation at S retains weight $\widehat{\nu}(S) - \widehat{\mu}_e$. Each scored macro-edge passes its coefficient to the child state:

$$\widehat{\nu}(S \cup \Delta S) \leftarrow \widehat{\nu}(S \cup \Delta S) + \widehat{\mu}_e. \tag{8}$$

FRM returns the macro-edge records $\mathcal{E}_B(c, x)$ and their coefficients for the token-score objective in §3.4. Algorithm 1 summarizes the procedure; Appendix F details the sampling and allocation rules.

> **Takeaway.** Under a limited state-evaluation budget, FRM balances *token coverage along each continuation* through macro reveals with *coverage of alternative reveal orders* through selective branching.

3.4 FEROM-GRPO

We use the FRM records to construct a token-score surrogate for GRPO [13]. For each context $c \sim \rho$, the fixed rollout policy samples G responses, $x_g \overset{\text{i.i.d.}}{\sim} \pi_{\theta_{\text{old}}}(\cdot \mid c), \quad g = 1, \ldots, G$, and their sequence-level rewards determine group-relative advantages $\{\widehat{A}_g\}_{g=1}^{G}$. FRM supplies a collection of macro-edge records $\mathcal{E}_B(c, x_g)$ and contribution coefficients $\widehat{\mu}_e$ for each response. These define the surrogate log-score

$$\widehat{L}_\theta(c, x_g) = \sum_{e=(S,i,\Delta S) \in \mathcal{E}_B(c, x_g)} \mathrm{sg}(\widehat{\mu}_e)\, \ell_{\theta, \Delta S}(S; c, x_g), \tag{9}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. The selected edges and coefficients remain fixed within an update, while gradients flow through the token scores. This uses the token-score component of Eq. 4; Appendix D gives its error decomposition relative to the full marginal score.

Policy objective. Using the same FRM records and coefficients for both policies, define the surrogate score ratio $\widehat{\rho}_\theta(c, x_g) = \exp\Big(\widehat{L}_\theta(c, x_g) - \widehat{L}_{\theta_{\text{old}}}(c, x_g)\Big)$. The resulting clipped objective is

$$\mathcal{J}_{\texttt{FEROM}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{g=1}^{G} \min\Big(\widehat{\rho}_\theta(c, x_g)\widehat{A}_g,\ \mathrm{clip}\big(\widehat{\rho}_\theta(c, x_g), 1-\epsilon, 1+\epsilon\big)\widehat{A}_g\Big)\right] - \lambda_{\mathrm{KL}}\widehat{D}_{\mathrm{ref}}. \tag{10}$$

The expectation is over $c \sim \rho$ and the sampled response groups; $\epsilon > 0$ is the clipping threshold, $\lambda_{\mathrm{KL}} \geq 0$ is the regularization coefficient, and $\widehat{D}_{\mathrm{ref}}$ is the reference-policy regularizer evaluated on the same FRM records.

4 Experiments

Setup. We evaluate FEROM on mathematical reasoning and code generation using LLaDA-Instruct [32] as the base model. Following the standard LLaDA-Instruct RL evaluation protocol [33, 31], we report results on GSM8K [10], MATH-500 [28], HumanEval [6], and MBPP [3]. For

Table 1: **Accuracy (%) across sequence lengths.** Best is in bold; second-best is underlined.

Model / Seq Len	GSM8K			MATH-500			HumanEval			MBPP		
	128	**256**	**512**	**128**	**256**	**512**	**128**	**256**	**512**	**128**	**256**	**512**
LLaDA-8B-Instruct [32]	68.7	76.7	78.2	26.0	32.4	36.2	27.4	35.5	37.8	36.2	41.2	40.4
LLaDA-1.5 [56]	-	83.3	-	-	-	-	<u>29.3</u>	39.6	**51.9**	39.6	39.9	38.8
d1 [53]	73.2	81.1	82.1	33.8	38.6	40.2	-	-	-	-	-	-
wd1 [44]	77.2	80.8	82.3	33.3	34.4	39.0	-	-	-	-	-	-
d-TreeRPO [35]	-	81.2	82.6	-	37.7	38.1	-	-	-	-	-	-
ESPO [33]	<u>80.0</u>	82.3	83.7	<u>36.0</u>	39.0	<u>43.4</u>	28.1	<u>42.1</u>	<u>50.0</u>	**47.4**	<u>44.6</u>	<u>44.2</u>
GDPO [38]	78.4	82.8	<u>85.0</u>	33.2	39.6	41.4	26.2	39.6	39.0	<u>43.6</u>	**50.6**	**47.1**
SPG [48]	78.5	**86.1**	84.5	33.4	<u>40.0</u>	41.8	-	-	-	-	-	-
FEROM (Ours)	**81.0**	<u>85.7</u>	**85.5**	**38.5**	**43.9**	**46.0**	**36.6**	**46.3**	48.2	38.2	44.0	39.2

training, we use GSM8K and the math training set for reasoning, and the AceCoder-87K [52] hard split for code generation. We follow the small-budget setting of ESPO [33]. Our main-table runs use 4-bit QLoRA with bfloat16 mixed precision. Since LLaDA uses semi-autoregressive blockwise decoding, we preserve the native block order and apply FEROM independently within each block. We use the macro-reveal framework described in §3.3; Appendix G provides training and evaluation details. For evaluation, generation lengths of 128, 256, and 512 tokens use 128, 256, and 512 decoding steps, respectively, matching the baselines.

Main results. Table 1 reports results for FEROM under the ESPO-matched small-budget setting. At 256 tokens, the training length, FEROM leads the compared methods on MATH-500 and HumanEval and ranks second on GSM8K. At 128 tokens, it leads on GSM8K, MATH-500, and HumanEval; at 512, it leads on both math benchmarks but scores below ESPO on HumanEval. FEROM scores below ESPO and GDPO on MBPP. We hypothesize that LLaDA-style confidence-based decoding limits the diversity of high-reward rollouts on this task. We leave co-designing decoding and endogenous reveal-order marginal policy optimization to future work. We additionally compare full-parameter FEROM with JustGRPO, achieving similar math and HumanEval results at half the scoring budget (Appendix H).

5 Analysis

5.1 Offline Approximation Quality

End-to-end RL combines approximation error with rollout and reward noise. We isolate estimation quality at continuation states from AceCoder rollouts, marginalizing the remaining reveal orders with conditional-CDF transition probabilities. Figure 3 compares budgeted estimates against this full conditional marginal-score reference, with and without reveal-order derivatives on the same scored edges. Appendix G.3 gives the matched cases, gradient projection, and budget accounting.

Candidates. We compare FRM with one sampled path, a dense prefix, left-to-right order, block-random masking, and two independent paths when affordable. The single-path baseline evenly spaces evaluations along a macro-reveal chain; the dense prefix scores only its first one-token transitions. FRM allocates evaluations across alternative continuations. No candidate uses oracle occupancies.

Results. At small budgets, macro reveals provide coverage of the remaining sequence that a short dense prefix misses; FRM retains this benefit along a single path. In the token-only comparison, block-random masking has weaker gradient alignment across budgets, suggesting that broad mask coverage alone does not adequately represent the endogenous reveal process. Left-to-right ordering improves on random masking but generally trails model-induced path sampling, highlighting the importance of matching the reveal-order distribution. As the budget grows, frontier expansion improves advantage-weighted agreement beyond a single path; at the largest budget, it also improves agreement and lowers error relative to two independent paths. This pattern is consistent with probability concentrated

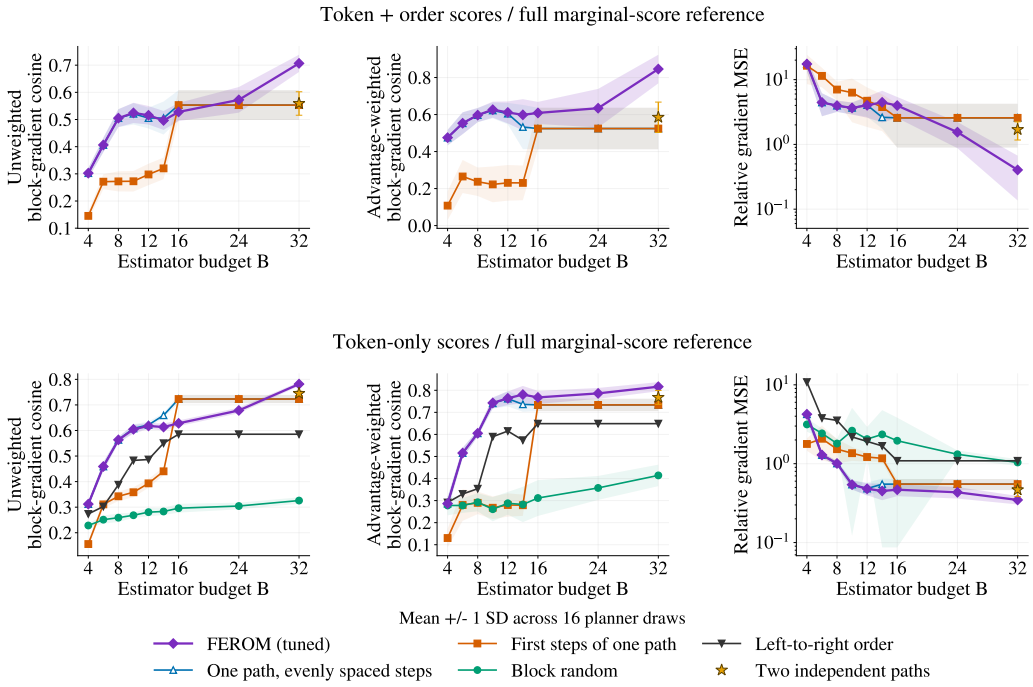

Figure 3: **Budgeted marginal-gradient approximation on AceCoder.** Token-plus-order (top) and token-only (bottom) estimates share a conditional marginal-gradient reference over 16 remaining positions. FEROM is tuned per budget on the displayed cohort, using one configuration across rows and metrics. Lines and bands show means and one standard deviation over 16 planner draws.

around a dominant reveal path: a single path captures much of the marginal contribution, while branching uses additional budget to cover alternative orders.

5.2 Low-Budget Training Dynamics

We use only two unique prompts per update, with eight responses each, to create a high-variance setting under a half-trajectory scoring budget. These full-parameter AceCoder experiments separately ablate branching and reveal-order scoring (Figure 4; Appendix G.4).

Unlike *Rollout trajectory*, *Single trajectory (macro)* constructs a fresh reveal path for the generated sequence using FRM's local probabilities. Macro reveals also change the scoring resolution, while `FEROM` additionally branches across alternative continuations.

In the branching ablation, `FEROM` improves training and validation rewards over *Single trajectory (macro)*, which uses the same macro-reveal horizon but disables branching.

In the reveal-order scoring ablation, *Rollout trajectory* uses token scores at sampled rollout states; *Rollout trajectory + order score* also differentiates the conditional reveal-order probability. The latter collapses early in training and validation, while the token-score control does not. This observed instability motivates the token-only surrogate for finite-budget training.

5.3 Computational Overhead

Figure 6 compares CDF-based FEROM with a matched block-random control, separating model computation from CDF evaluation, macro-proposal sampling, and frontier bookkeeping. Model computation dominates the measured update, while CDF evaluation contributes a small fraction. In the memory-efficient replay implementation, separate planning and scoring forwards account for most of the additional cost. Sharing these forwards reduces repeated model computation when activation memory permits. We discuss more profiling details in Appendix G.5.

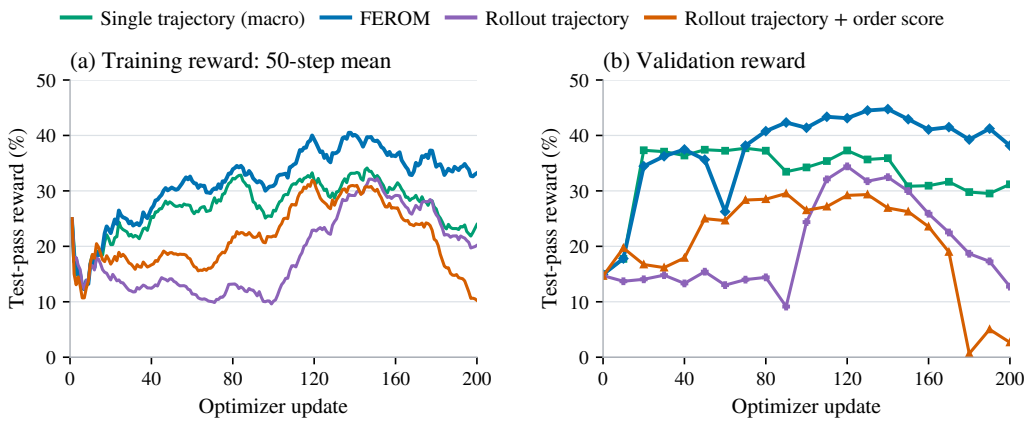

Figure 4: **Low-budget AceCoder training.** Training reward (left) and validation reward (right) for the frontier branching and reveal-order scoring ablations.

6 Related Work

Train-test mismatch of unmasking process in MDMs. Concurrent work observes that MDM generation depends on the unmasking process, so random-mask training or ELBO evaluation can mismatch the states visited by practical samplers. DUEL [45] formalizes deterministic position-selection rules and computes exact likelihood along the induced unmasking path, while PUMA [23] reduces supervised train–test mismatch through progressive, teacher-forced unmasking. Our work differs in objective and estimator: `FEROM` targets reinforcement learning. Rather than evaluating deterministic-path likelihoods or changing the supervised context distribution, we estimate the marginal response-policy gradients by approximating posterior reveal-edge occupancies under a controllable scoring-state budget.

Reinforcement learning for MDMs. Recent work has begun adapting RL and preference optimization to diffusion language models. d1 [53] introduces a post-training recipe for diffusion LMs, including a critic-free GRPO-style RL method for reasoning. SPG [48] addresses the intractability of dLLM likelihoods by constructing sandwiched policy-gradient estimators. Other methods formulate denoising as a sequential decision process, including MDPO-style denoising-trajectory optimization [16], learned unmasking policies [21], dUltra [7], which introduces an additional unmasking planner optimized with GRPO for faster parallel decoding, and TraceRL [49], introducing an explicit value model for sampled trajectories. In contrast, `FEROM` does not introduce a separate reveal-policy head or rely on a pre-defined masking objective or a single sampled reveal path. We discuss additional related work in Appendix A.

7 Conclusion

We introduced `FEROM`, a principled framework for reinforcement learning of masked diffusion language models with endogenous reveal orders. By combining a reveal-state DAG with budgeted frontier estimation, it connects the marginal-policy objective to a practical token-score surrogate.

Limitations and future work. FRM approximates the reveal-order marginal under a finite computation budget. For the confidence-based sampler, local next-reveal probabilities can be computed exactly, but finite frontier coverage, approximate continuation weights, macro reveals, and the token-score surrogate still limit marginal-gradient accuracy. Its effectiveness therefore depends on the rollout sampler and the allocation of scoring evaluations. A broader study of reveal heuristics, macro-reveal selection, and frontier allocation strategies would further clarify when marginal edge coverage is most beneficial. Finally, improving the underlying MDM architecture and memory efficiency remains important for scaling marginal-policy optimization to longer responses and larger models.

Acknowledgments and Disclosure of Funding

This work was funded in part by the National Science Foundation (NSF) Awards IIS-2416835, DARPA I2O CARVE, NIST 60NANB26D035, Cisco and Amazon. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

The authors declare no competing interests.

References

[1] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

[2] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[3] 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.

[4] Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, Chengxi Li, Chongxuan Li, Jianguo Li, Zehuan Li, Huabin Liu, Lin Liu, Guoshan Lu, Xiaocheng Lu, Yuxin Ma, Jianfeng Tan, Lanning Wei, Ji-Rong Wen, Yipeng Xing, Xiaolu Zhang, Junbo Zhao, Da Zheng, Jun Zhou, Junlin Zhou, Zhanchao Zhou, Liwang Zhu, and Yihong Zhuang. LLaDA2.0: Scaling up diffusion language models to 100B. *arXiv preprint arXiv:2512.15745*, 2025.

[5] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *International Conference on Machine Learning*, pages 5453–5512. PMLR, 2024.

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[7] Shirui Chen, Jiantao Jiao, Lillian J Ratliff, and Banghua Zhu. dUltra: Ultra-fast diffusion language models via reinforcement learning. *arXiv preprint arXiv:2512.21446*, 2025.

[8] Ziyu Chen, Xinbei Jiang, Peng Sun, and Tao Lin. Optimizing decoding paths in masked diffusion models by quantifying uncertainty. *arXiv preprint arXiv:2512.21336*, 2025.

[9] Shuang Cheng, Yihan Bian, Dawei Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, et al. SDAR: A synergistic diffusion-autoregression paradigm for scalable sequence generation. *arXiv preprint arXiv:2510.06303*, 2025.

[10] 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.

[11] Nima Fathi, Torsten Scholak, and Pierre-André Noël. Unifying autoregressive and diffusion-based sequence generation. *arXiv preprint arXiv:2504.06416*, 2025.

[12] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[14] Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11575–11596, 2023.

[15] Marton Havasi, Brian Karrer, Itai Gat, and Ricky TQ Chen. Edit flows: Flow matching with edit operations. *arXiv preprint arXiv:2506.09018*, 2025.

[16] Haoyu He, Katrin Renz, Yong Cao, and Andreas Geiger. MDPO: Overcoming the training-inference divide of masked diffusion language models. *arXiv preprint arXiv:2508.13148*, 2025.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[18] Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021.

[19] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[20] Zheng Huang, Kiran Ramnath, Yueyan Chen, Aosong Feng, Sangmin Woo, Balasubramaniam Srinivasan, Zhichao Xu, Kang Zhou, Shuai Wang, Haibo Ding, and Lin Lee Cheong. Diffusion language model inference with Monte Carlo tree search. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 3493–3512, Rabat, Morocco, March 2026. Association for Computational Linguistics.

[21] Metod Jazbec, Theo X Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning unmasking policies for diffusion language models. *arXiv preprint arXiv:2512.09106*, 2025.

[22] Jaeyeon Kim, Lee Cheuk-Kit, Carles Domingo-Enrich, Yilun Du, Sham Kakade, Timothy Ngotiaoco, Sitan Chen, and Michael Albergo. Any-order flexible length masked diffusion. *arXiv preprint arXiv:2509.01025*, 2025.

[23] Jaeyeon Kim, Jonathan Geuter, David Alvarez-Melis, Sham Kakade, and Sitan Chen. Stop training for the worst: Progressive unmasking accelerates masked diffusion training. *arXiv preprint arXiv:2602.10314*, 2026.

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

[25] Fanheng Kong, Jingyuan Zhang, Yahui Liu, Zirui Wu, Yu Tian, Victoria W., and Guorui Zhou. Accelerating diffusion LLM inference via local determinism propagation. *arXiv preprint arXiv:2510.07081*, 2025.

[26] Sanghyun Lee, Seungryong Kim, Jongho Park, and Dongmin Park. Lookahead unmasking elicits accurate decoding in diffusion language models. *arXiv preprint arXiv:2511.05563*, 2025.

[27] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-LM improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

[28] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

[29] Yangzhou Liu, Yue Cao, Hao Li, Gen Luo, Zhe Chen, Weiyun Wang, Xiaobo Liang, Biqing Qi, Lijun Wu, Changyao Tian, et al. Sequential diffusion language models. *arXiv preprint arXiv:2509.24007*, 2025.

[30] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, pages 32819–32848. PMLR, 2024.

[31] Zanlin Ni, Shenzhi Wang, Yang Yue, Tianyu Yu, Weilin Zhao, Yeguo Hua, Tianyi Chen, Jun Song, Cheng Yu, Bo Zheng, and Gao Huang. The flexibility trap: Rethinking the value of arbitrary order in diffusion language models. *arXiv preprint arXiv:2601.15165*, 2026.

[32] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[33] Jingyang Ou, Jiaqi Han, Minkai Xu, Shaoxuan Xu, Jianwen Xie, Stefano Ermon, Yi Wu, and Chongxuan Li. Principled RL for diffusion LLMs emerges from a sequence-level perspective. *arXiv preprint arXiv:2512.03759*, 2025.

[34] Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024.

[35] Leyi Pan, Shuchang Tao, Yunpeng Zhai, Zheyu Fu, Liancheng Fang, Minghua He, Lingzhe Zhang, Zhaoyang Liu, Bolin Ding, Aiwei Liu, et al. d-TreeRPO: Towards more reliable policy optimization for diffusion language models. *arXiv preprint arXiv:2512.09675*, 2025.

[36] Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. Path planning for masked diffusion model sampling. *arXiv preprint arXiv:2502.03540*, 2025.

[37] Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. CANDI: Hybrid discrete-continuous diffusion models. *arXiv preprint arXiv:2510.22510*, 2025.

[38] Kevin Rojas, Jiahe Lin, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, Molei Tao, and Wei Deng. Improving reasoning for diffusion language models via group diffusion policy optimization. *arXiv preprint arXiv:2510.08554*, 2025.

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

[40] Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. *arXiv preprint arXiv:2506.10892*, 2025.

[41] Yangyi Shen, Tianjian Feng, Jiaqi Han, Wen Wang, Tianlang Chen, Chunhua Shen, Jure Leskovec, and Stefano Ermon. Improving diffusion language model decoding through joint search in generation order and token space. *arXiv preprint arXiv:2601.20339*, 2026.

[42] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in Neural Information Processing Systems*, 37:103131–103167, 2024.

[43] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[44] Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. *arXiv preprint arXiv:2507.08838*, 2025.

[45] Gilad Turok, Yair Schiff, Chris De Sa, and Volodymyr Kuleshov. DUEL: Exact likelihood reshapes masked diffusion scaling laws and beyond. *arXiv preprint arXiv:2603.01367*, 2026.

[46] Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *International Conference on Machine Learning*, pages 467–475. PMLR, 2014.

[47] Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion. *arXiv preprint arXiv:2503.04482*, 2025.

[48] Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, et al. SPG: Sandwiched policy gradient for masked diffusion language models. *arXiv preprint arXiv:2510.09541*, 2025.

[49] Yinjie Wang, Ling Yang, Bowen Li, Ye Tian, Ke Shen, and Mengdi Wang. Revolutionizing reinforcement learning framework for diffusion large language models. *arXiv preprint arXiv:2509.06949*, 2025.

[50] Shutong Wu and Jiawei Zhang. Free draft-and-verification: Toward lossless parallel decoding for diffusion large language models. *arXiv preprint arXiv:2510.00294*, 2025.

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

[52] Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. ACECODER: Acing coder RL via automated test-case synthesis. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12023–12040, Vienna, Austria, July 2025. Association for Computational Linguistics.

[53] Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[54] Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024.

[55] Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation. *arXiv preprint arXiv:2302.05737*, 2023.

[56] Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, et al. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. *arXiv preprint arXiv:2505.19223*, 2025.

A Additional Related Work

Diffusion Language Models. Continuous diffusion models for image generation established the foundation of iterative refinement via noise reversal [43, 17], while discrete diffusion models adapt these stochastic processes to categorical data domains [2, 42, 5]. This progression demonstrates the effectiveness of diffusion mechanisms across different data modalities. Continuous diffusion models in the text embedding space address the challenge of discrete text generation by mapping tokens to a continuous latent space for diffusion processes [27, 12, 14, 40]. By operating on embeddings, these methods utilize the well-established mathematical framework of continuous diffusion while avoiding direct operations on discrete variables. Masked diffusion language models (MDLMs) operate directly on discrete tokens by replacing the Gaussian noise process with a corruption process based on masking [30, 55, 39, 32, 51]. This approach simplifies the learning objective to recovering masked information, effectively modeling the joint probability distribution of the text. Combining continuous and discrete diffusion for text generation seeks to integrate the optimization benefits of continuous spaces with the semantic integrity of discrete modeling [47, 37]. Such hybrid methodologies aim to enhance generation fidelity and convergence stability. Block-diffusion interpolates autoregressive and masked diffusion generation, allowing for the simultaneous prediction of token blocks while maintaining sequential dependencies between blocks [1, 9]. Another way to view block-diffusion is semi-autoregressive generation, unifying autoregressive and diffusion generation to bridge the gap between purely sequential and fully parallel decoding strategies [11, 29]. Another notable trend in diffusion LMs is to generate flexible length, which is more intuitive and implies more powerful abilities of diffusion models by decoupling generation from a predetermined sequence length [15, 22]. Enabling variable-length outputs allows these models to better accommodate the dynamic nature of natural language sequences.

Inference-Time Reveal-Order Search and Planning. A complementary line of work improves masked diffusion LM generation by modifying the inference-time unmasking path. Several methods use confidence or uncertainty to accelerate decoding or select better revealing trajectories. LocalLeap [25] exploits local determinism around high-confidence anchors for adaptive parallel decoding, while FreeDave [50] uses draft-and-verification to reduce the number of diffusion forwards. Other methods treat decoding more explicitly as a path-selection problem: Lookahead Unmasking [26] evaluates candidate unmasking paths with uncertainty-based verification, Denoising Entropy [8] scores generation paths by cumulative predictive uncertainty, and Path Planning [36] separates planning from denoising at each step.

Recent work also explores broader search over both orders and token values. Order-Token Search [41] performs beam-style search jointly over generation order and token assignments, and MEDAL [20] applies Monte Carlo tree search to explore promising denoising trajectories. These inference-time methods are closely related to our reveal-state DAG perspective: they also recognize that generation quality depends on the path through partially revealed states. However, their goal is to choose or verify a good decoding trajectory at test time. In contrast, `FEROM` uses reveal-order structure during RL training to approximate the marginal response-policy gradient. FRM does not search for a single best inference path; it constructs a budgeted set of posterior-relevant reveal edges for policy-gradient estimation.

B Theory

B.1 Proof of Theorem 3.1

We prove Theorem 3.1 in three steps. Throughout this section, expectations are taken over

$$c \sim \rho, \qquad (x, \sigma) \sim P_\theta(\cdot, \cdot \mid c),$$

unless otherwise specified. We assume that the reward $R(c, x)$ is fixed with respect to θ, and that differentiation can be exchanged with the finite sum over responses and reveal orders. The marginal response policy is

$$\pi_\theta(x \mid c) = \sum_\sigma P_\theta(x, \sigma \mid c).$$

Value depends only on the marginal response policy. Since the reward depends only on the context c and completed response x, not on the latent reveal order σ, we have

$$J(\theta) = \mathbb{E}_{c\sim\rho} \sum_{x,\sigma} P_\theta(x, \sigma \mid c) R(c, x) \tag{11}$$

$$= \mathbb{E}_{c\sim\rho} \sum_{x} \left(\sum_{\sigma} P_\theta(x, \sigma \mid c) \right) R(c, x) \tag{12}$$

$$= \mathbb{E}_{c\sim\rho} \sum_{x} \pi_\theta(x \mid c) R(c, x). \tag{13}$$

Thus, for sequence-level rewards, the conditional reveal-order distribution $P_\theta(\sigma \mid c, x)$ describes the latent execution trace. The reveal process affects how the response is generated, but the objective depends on it only through the induced marginal response policy $\pi_\theta(x \mid c)$.

Marginal score as a Rao–Blackwellized path score. For any c, x with $\pi_\theta(x \mid c) > 0$,

$$\mathbb{E}_{\sigma\sim P_\theta(\cdot|c,x)} \left[\nabla_\theta \log P_\theta(x, \sigma \mid c)\right] = \sum_{\sigma} P_\theta(\sigma \mid c, x) \frac{\nabla_\theta P_\theta(x, \sigma \mid c)}{P_\theta(x, \sigma \mid c)} \tag{14}$$

$$= \sum_{\sigma} \frac{P_\theta(x, \sigma \mid c)}{\pi_\theta(x \mid c)} \frac{\nabla_\theta P_\theta(x, \sigma \mid c)}{P_\theta(x, \sigma \mid c)} \tag{15}$$

$$= \frac{\sum_\sigma \nabla_\theta P_\theta(x, \sigma \mid c)}{\pi_\theta(x \mid c)} \tag{16}$$

$$= \frac{\nabla_\theta \pi_\theta(x \mid c)}{\pi_\theta(x \mid c)} \tag{17}$$

$$= \nabla_\theta \log \pi_\theta(x \mid c). \tag{18}$$

Therefore, the marginal response-policy score is the conditional expectation of the joint path score:

$$\nabla_\theta \log \pi_\theta(x \mid c) = \mathbb{E}\left[\nabla_\theta \log P_\theta(x, \sigma \mid c) \mid x, c\right].$$

Now define

$$g_{\mathrm{traj}}(x, \sigma, c) = R(c, x) \nabla_\theta \log P_\theta(x, \sigma \mid c),$$

and

$$g_{\mathrm{marg}}(x, c) = R(c, x) \nabla_\theta \log \pi_\theta(x \mid c).$$

Since $R(c, x)$ is measurable with respect to (c, x), the previous identity implies

$$g_{\mathrm{marg}}(x, c) = \mathbb{E}\left[g_{\mathrm{traj}}(x, \sigma, c) \mid x, c\right].$$

This proves the Rao–Blackwell relation in Theorem 3.1.

Unbiasedness. The trajectory estimator is unbiased:

$$\mathbb{E}[g_{\mathrm{traj}}] = \mathbb{E}_{c\sim\rho} \sum_{x,\sigma} P_\theta(x, \sigma \mid c) R(c, x) \nabla_\theta \log P_\theta(x, \sigma \mid c) \tag{19}$$

$$= \mathbb{E}_{c\sim\rho} \sum_{x,\sigma} R(c, x) \nabla_\theta P_\theta(x, \sigma \mid c) \tag{20}$$

$$= \nabla_\theta \mathbb{E}_{c\sim\rho} \sum_{x,\sigma} P_\theta(x, \sigma \mid c) R(c, x) \tag{21}$$

$$= \nabla_\theta J(\theta). \tag{22}$$

Similarly, using the marginal response policy,

$$\mathbb{E}[g_{\mathrm{marg}}] = \mathbb{E}_{c\sim\rho} \sum_{x} \pi_\theta(x \mid c) R(c, x) \nabla_\theta \log \pi_\theta(x \mid c) \tag{23}$$

$$= \mathbb{E}_{c\sim\rho} \sum_{x} R(c, x) \nabla_\theta \pi_\theta(x \mid c) \tag{24}$$

$$= \nabla_\theta \mathbb{E}_{c\sim\rho} \sum_{x} \pi_\theta(x \mid c) R(c, x) \tag{25}$$

$$= \nabla_\theta J(\theta). \tag{26}$$

Thus,

$$\mathbb{E}[g_{\mathrm{traj}}] = \mathbb{E}[g_{\mathrm{marg}}] = \nabla_\theta J(\theta).$$

Variance dominance. Since

$$g_{\mathrm{marg}} = \mathbb{E}[g_{\mathrm{traj}} \mid x, c],$$

the law of total covariance gives

$$\mathrm{Cov}(g_{\mathrm{traj}}) = \mathrm{Cov}\left(\mathbb{E}[g_{\mathrm{traj}} \mid x, c]\right) + \mathbb{E}\left[\mathrm{Cov}(g_{\mathrm{traj}} \mid x, c)\right].$$

Using the Rao–Blackwell relation,

$$\mathrm{Cov}(g_{\mathrm{traj}}) = \mathrm{Cov}(g_{\mathrm{marg}}) + \mathbb{E}\left[\mathrm{Cov}(g_{\mathrm{traj}} \mid x, c)\right].$$

The conditional covariance term is positive semidefinite, so

$$\mathrm{Cov}(g_{\mathrm{marg}}) \preceq \mathrm{Cov}(g_{\mathrm{traj}}).$$

Equivalently, for any test direction u,

$$\mathrm{Var}(u^\top g_{\mathrm{traj}}) - \mathrm{Var}(u^\top g_{\mathrm{marg}}) = \mathbb{E}\left[\mathrm{Var}(u^\top g_{\mathrm{traj}} \mid x, c)\right] \geq 0.$$

Since

$$g_{\mathrm{traj}} = R(c, x) \nabla_\theta \log P_\theta(x, \sigma \mid c),$$

this conditional variance can also be written as

$$\mathbb{E}\left[R(c, x)^2 \,\mathrm{Var}\left(u^\top \nabla_\theta \log P_\theta(x, \sigma \mid c) \mid x, c\right)\right].$$

Therefore, equality holds if and only if

$$\mathrm{Var}(g_{\mathrm{traj}} \mid x, c) = 0$$

almost surely. When $R(c, x) \neq 0$, this is equivalent to

$$\nabla_\theta \log P_\theta(x, \sigma \mid c)$$

being almost surely constant over reveal orders σ conditional on x, c.

C Endogenous Reveal Probabilities for Confidence-Based Decoding

This section describes LLaDA's low-confidence remasking rule [32] with the temperature-controlled token sampling used in its released implementation.[3] We derive the probabilities of revealing specified response tokens in a general k-token update and its one-token special case, then describe their computation and cost.

C.1 Confidence-Based Decoding

Given a context c, a response x, and a revealed set S, the partial response $\tilde{x}_S$ contains x^i for $i \in S$ and masks at the remaining response positions. Let $\mathcal{I}(S)$ be the masked positions eligible for the next update: those in the active block for blockwise decoding, or all masked response positions otherwise.

One model evaluation gives logits $\mathbf{a}_{\theta,j} = \mathrm{logits}_\theta(c, \tilde{x}_S)_j$ over vocabulary $\mathcal{V}$. At temperature $\tau > 0$, candidates at eligible positions are drawn independently:

$$p_{\theta,j}(v) = \mathrm{softmax}(\mathbf{a}_{\theta,j}/\tau)_v, \qquad Z_j \sim \mathrm{Cat}(\cdot; p_{\theta,j}). \tag{27}$$

The confidence assigned to Z_j is $\mathrm{softmax}(\mathbf{a}_{\theta,j})_{Z_j}$, its probability under the untempered model output. Temperature therefore changes which tokens are sampled, but not how their confidence is measured. At $\tau = 0$, the decoder chooses an argmax token deterministically.

Rank the candidates by decreasing confidence, with lower position indices winning ties. Let $\mathrm{Top}_k(Z)$ be the set of the first k positions. For a scheduled reveal count $1 \leq k \leq |\mathcal{I}(S)|$, the decoder reveals the positions ΔS and updates the state as

$$\Delta S = \mathrm{Top}_k(Z), \qquad S' = S \cup \Delta S. \tag{28}$$

Using the predictions at $\tilde{x}_S$, the decoder fills each selected position $j \in \Delta S$ with its sampled token Z_j, leaving the remaining eligible positions masked.

[3] `https://github.com/ML-GSAI/LLaDA/blob/main/generate.py`

C.2 Target-Conditioned Reveal Probabilities

Revealing specified response tokens requires both proposing those tokens and selecting their positions. Once their candidate values are fixed, selection depends on the remaining sampled candidates. The confidence-based rule turns this event into independent confidence comparisons, whose probabilities are obtained from cumulative distribution functions (CDFs).

k-token reveals. For a queried reveal set $\Delta S \subseteq \mathcal{I}(S)$ with $|\Delta S| = k$, we condition on its candidates matching the response tokens, $Z_{\Delta S} = x_{\Delta S}$. The *competing candidates* are the sampled tokens Z_j at the remaining eligible positions $j \in \mathcal{I}(S) \setminus \Delta S$. This conditioning fixes the target candidates' confidences, while the competing candidates remain random and independent.

Let $i_{\Delta S}$ be the position of the lowest-ranked target token in ΔS. The queried set is selected precisely when every competing candidate ranks below this token. We therefore compute the probability of each such comparison and multiply the results.

At each competing position j, confidence values are fixed by the logits, while the candidate Z_j is random. The strict and non-strict CDFs of its confidence at threshold u are

$$\begin{aligned} F^{<}_{\theta,j}(u) &= \sum_{v \in \mathcal{V}} p_{\theta,j}(v) \mathbf{1}\{\mathrm{softmax}(\mathbf{a}_{\theta,j})_v < u\}\,, \\ F^{\leq}_{\theta,j}(u) &= \sum_{v \in \mathcal{V}} p_{\theta,j}(v) \mathbf{1}\{\mathrm{softmax}(\mathbf{a}_{\theta,j})_v \leq u\}\,. \end{aligned} \tag{29}$$

Each sum weights the possible candidate tokens by their sampling probabilities. For example, $F^{\leq}_{\theta,j}(u)$ is the probability that the sampled token at position j has confidence at most u.

The conditional selection probability and joint transition probability are

$$\begin{aligned} r^x_\theta(\Delta S \mid c, \tilde{x}_S) &= \Pr(\mathrm{Top}_k(Z) = \Delta S \mid c, \tilde{x}_S, Z_{\Delta S} = x_{\Delta S})\,, \\ \psi_\theta(S, \Delta S; c, x) &= \left(\prod_{i \in \Delta S} p_{\theta,i}(x^i) \right) r^x_\theta(\Delta S \mid c, \tilde{x}_S). \end{aligned} \tag{30}$$

The second line multiplies the target-token sampling probabilities by the conditional selection probability. Evaluating every competing candidate's CDF at the confidence of $x^{i_{\Delta S}}$ gives

$$\begin{aligned} r^x_\theta(\Delta S \mid c, \tilde{x}_S) = &\prod_{\substack{j \in \mathcal{I}(S) \setminus \Delta S \\ j < i_{\Delta S}}} F^{<}_{\theta,j}(\mathrm{softmax}(\mathbf{a}_{\theta,i_{\Delta S}})_{x^{i_{\Delta S}}}) \\ &\cdot \prod_{\substack{j \in \mathcal{I}(S) \setminus \Delta S \\ j > i_{\Delta S}}} F^{\leq}_{\theta,j}(\mathrm{softmax}(\mathbf{a}_{\theta,i_{\Delta S}})_{x^{i_{\Delta S}}})\,. \end{aligned} \tag{31}$$

If $\Delta S = \mathcal{I}(S)$, there are no competitors and the selection probability is one. These conditional probabilities assume that all queried target tokens have positive sampling probability; otherwise the joint transition probability is zero.

One-token reveals. For $k = 1$, set $\Delta S = \{i\}$ and let I denote the highest-ranked position. Writing the singleton query by its position index gives

$$\begin{aligned} r^x_\theta(i \mid c, \tilde{x}_S) &= \Pr(I = i \mid c, \tilde{x}_S, Z_i = x^i) \\ &= \prod_{\substack{j \in \mathcal{I}(S) \\ j < i}} F^{<}_{\theta,j}(\mathrm{softmax}(\mathbf{a}_{\theta,i})_{x^i}) \prod_{\substack{j \in \mathcal{I}(S) \\ j > i}} F^{\leq}_{\theta,j}(\mathrm{softmax}(\mathbf{a}_{\theta,i})_{x^i})\,. \end{aligned} \tag{32}$$

This is the one-token selection probability in Eq. 5. Multiplying by the probability of sampling x^i gives the corresponding edge factor,

$$\psi_\theta(S, i; c, x) = p_{\theta,i}(x^i)\, r^x_\theta(i \mid c, \tilde{x}_S). \tag{33}$$

Each $r^x_\theta(i \mid c, \tilde{x}_S)$ conditions on a different event, $Z_i = x^i$. These probabilities therefore need not sum to one over positions, and the edge factor uses them without cross-position normalization.

Remark C.1 (Parallel decoding). For parallel decoding that reveals $k > 1$ tokens per update, the corresponding transition is the k-reveal kernel in Eq. 30, with all selected tokens predicted at the same state. The current FEROM formulation uses one-token transitions; extending frontier marginalization directly to k-reveal transitions is left to future work.

C.3 Computation and Cost

At each eligible position, sort vocabulary tokens by increasing confidence and store cumulative sums of their sampling probabilities. A binary search then gives the total probability below, or at most, a supplied confidence threshold. For a reveal-set query, use the confidence of its lowest-ranked target as this threshold at each competing position. Summing the log-CDF values evaluates the product stably. Algorithm 2 summarizes the calculation.

Let $m = |\mathcal{I}(S)|$ and $V = |\mathcal{V}|$. Building the sorted tables takes $O(mV \log V)$ time and $O(mV)$ storage. Once built, a query costs $O(|\Delta S| + (m - |\Delta S|) \log V)$: find the lowest-ranked target, then look up the competitors' CDFs. Computing all m singleton queries therefore takes $O(mV \log V + m^2 \log V)$ total time. If their CDF factors are cached, storage becomes $O(mV + m^2)$. These calculations reuse the existing logits and require no additional model evaluation; processing positions in chunks reduces peak table memory.

Algorithm 2 Computing Target-Conditioned Reveal-Set Probabilities

Require: Current-state logits, response x, eligible positions $\mathcal{I}(S)$, temperature τ, and nonempty reveal sets whose target tokens have positive sampling probability

for $j \in \mathcal{I}(S)$ **do**
    Compute token-sampling probabilities $p_{\theta,j}$ and confidences $\mathrm{softmax}(\mathbf{a}_{\theta,j})$
    Sort tokens by increasing confidence and store cumulative sampling probabilities
end for
for each queried set $\Delta S \subseteq \mathcal{I}(S)$ **do**
    Find the lowest-ranked target position $i_{\Delta S}$
    Look up the competing positions' CDFs at the confidence of $x^{i_{\Delta S}}$
    Compute conditional selection probability $r^x_\theta(\Delta S \mid c, \tilde{x}_S)$ by Eq. 31
    Compute joint transition probability $\psi_\theta(S, \Delta S; c, x)$ by Eq. 30
end for
return conditional selection and joint transition probabilities for the queried sets

D Computing Posterior Occupancies on the Reveal-State DAG

This section derives the backward recursion for completion mass and the forward propagation of node and edge occupancies on the reveal-state DAG. These recursions relate the local edge factors to the response-conditioned next-reveal posterior and marginal-score decomposition used in §3.2.

Setup. We consider a context c and a sequence x for which $\pi_\theta(x \mid c) > 0$. Let $S_0 \subseteq [n]$ be the initial revealed set, with any initially fixed tokens included in the conditioning context c; ordinary response generation uses $S_0 = \emptyset$. A reveal order $\sigma = (\sigma_1, \ldots, \sigma_{n-|S_0|})$ is a permutation of the initially masked positions $[n] \setminus S_0$. Define the revealed set after t reveal steps as

$$S_t(\sigma) = S_0 \cup \{\sigma_1, \ldots, \sigma_t\}.$$

Thus, $S_0(\sigma) = S_0$, and $S_{n-|S_0|}(\sigma) = [n]$.

For a revealed set S and an eligible next position $i \in \mathcal{I}(S)$, define the local edge factor

$$\psi_\theta(S, i; c, x) \triangleq r^x_\theta(i \mid c, \tilde{x}_S)\, p_\theta(x^i \mid c, \tilde{x}_S).$$

The first term is the probability of selecting i conditional on its proposal being x^i, with competitors marginalized. The second is the probability of proposing that token under the actual rollout distribution. Set $\psi_\theta(S, i; c, x) = 0$ for ineligible positions, so the following sums also cover restricted schedules such as blockwise decoding. For a full reveal order σ, the joint path probability of

generating x through σ is

$$P_\theta(x, \sigma \mid c) = \prod_{t=0}^{n-|S_0|-1} \psi_\theta(S_t(\sigma), \sigma_{t+1}; c, x).$$

Completion mass. For any revealed set $S \supseteq S_0$, define the completion mass

$$\mathcal{Z}_\theta(S; c, x)$$

as the total probability mass of all reveal continuations that complete the response x from state S. Equivalently,

$$\mathcal{Z}_\theta(S; c, x) = \sum_{\text{reveal orders from } S \text{ to } [n]} \prod_{\text{edges } (S', i) \text{ on the continuation}} \psi_\theta(S', i; c, x).$$

This is a path-sum normalizer, not a learned value function. It satisfies the backward recursion

$$\mathcal{Z}_\theta(S; c, x) = \sum_{i \notin S} \psi_\theta(S, i; c, x)\, \mathcal{Z}_\theta(S \cup \{i\}; c, x), \qquad \mathcal{Z}_\theta([n]; c, x) = 1.$$

The marginal response policy is the completion mass at the initial state:

$$\pi_\theta(x \mid c) = \sum_\sigma P_\theta(x, \sigma \mid c) = \mathcal{Z}_\theta(S_0; c, x).$$

Response-conditioned next-reveal posterior. The target-conditioned selection probability $r^x_\theta(i \mid c, \tilde{x}_S)$ conditions on one proposed token, not on producing the entire response. The response-conditioned next-reveal posterior instead asks: among all paths that generate x and have reached S, how likely is the next edge to reveal position i?

For any nonterminal state S reached with positive probability under $P_\theta(\sigma \mid c, x)$, this posterior is

$$\eta_\theta(i \mid S, c, x) = \Pr_\theta \left[\sigma_{|S|-|S_0|+1} = i \mid S_{|S|-|S_0|}(\sigma) = S,\ c,\ x\right],$$

and it has the closed form

$$\eta_\theta(i \mid S, c, x) = \frac{\psi_\theta(S, i; c, x)\, \mathcal{Z}_\theta(S \cup \{i\}; c, x)}{\mathcal{Z}_\theta(S; c, x)}.$$

Equivalently,

$$\eta_\theta(i \mid S, c, x) \propto r^x_\theta(i \mid c, \tilde{x}_S)\, p_\theta(x^i \mid c, \tilde{x}_S)\, \mathcal{Z}_\theta(S \cup \{i\}; c, x).$$

Thus the posterior combines the conditional selection probability, the likelihood of proposing the target token, and the future completion mass. Local conditioning alone does not recover this posterior. Values of η_θ at states with zero posterior occupancy do not contribute to the marginal score.

Node and edge occupancies. The response-conditioned posterior over reveal trajectories induces occupancies on the reveal-state DAG. The oracle node occupancy is

$$\nu^\star_\theta(S; c, x) = \Pr_{\sigma \sim P_\theta(\sigma \mid c, x)} \left[S_{|S|-|S_0|}(\sigma) = S\right],$$

for $S \supseteq S_0$, and $\nu^\star_\theta(S; c, x) = 0$ otherwise. The initial occupancy is

$$\nu^\star_\theta(S_0; c, x) = 1.$$

For an edge $e = (S, i)$, the oracle edge occupancy is

$$\mu^\star_{\theta,(S,i)}(c, x) = \Pr_{\sigma \sim P_\theta(\sigma \mid c, x)} \left[S_{|S|-|S_0|}(\sigma) = S,\ \sigma_{|S|-|S_0|+1} = i\right].$$

Using the response-conditioned next-reveal posterior, the edge occupancy factorizes as

$$\mu^\star_{\theta,(S,i)}(c, x) = \nu^\star_\theta(S; c, x)\, \eta_\theta(i \mid S, c, x).$$

Node occupancies can also be propagated forward through incoming edges. For $S \neq S_0$,

$$\nu^\star_\theta(S; c, x) = \sum_{j \in S \setminus S_0} \mu^\star_{\theta,(S\setminus\{j\},j)}(c, x).$$

This recurrence is the exact counterpart of the approximate node-occupancy propagation used by FRM.

Edge-occupancy form of the marginal score. Assume the relevant edge factors are differentiable and positive on the support of the posterior. The marginal score can be written as an edge-occupancy weighted sum. Starting from

$$\pi_\theta(x \mid c) = \sum_\sigma P_\theta(x, \sigma \mid c),$$

we have

$$\nabla_\theta \log \pi_\theta(x \mid c) = \sum_\sigma P_\theta(\sigma \mid c, x) \nabla_\theta \log P_\theta(x, \sigma \mid c) \tag{34}$$

$$= \sum_\sigma P_\theta(\sigma \mid c, x) \sum_{t=0}^{n-|S_0|-1} \nabla_\theta \log \psi_\theta(S_t(\sigma), \sigma_{t+1}; c, x) \tag{35}$$

$$= \sum_{e=(S,i)} \mu^\star_{\theta,e}(c, x) \nabla_\theta \log \psi_\theta(S, i; c, x) \tag{36}$$

$$= \sum_{e=(S,i)} \mu^\star_{\theta,e}(c, x) \nabla_\theta \log \left[r^x_\theta(i \mid c, \tilde{x}_S) p_\theta(x^i \mid c, \tilde{x}_S) \right]. \tag{37}$$

This is the exact reveal-state DAG form of the Rao–Blackwellized marginal policy score. The posterior occupancies here are coefficients from differentiating the original path sum (Fisher's identity). One may detach them when evaluating this weighted score; no extra derivative of occupancy is required. The score of r^x_θ, however, is part of the differentiated edge factor.

Relation to the token-score surrogate. The practical estimator in the main paper uses the reveal process to estimate posterior edge occupancies, but applies gradients only to the token-likelihood component. For edge $e = (S, i)$, define

$$h_e = \nabla_\theta \log p_\theta(x^i \mid c, \tilde{x}_S).$$

The oracle token-score target is

$$g^\star_{\mathrm{tok}}(c, x) = \sum_e \mu^\star_{\theta,e}(c, x) h_e.$$

This omits the reveal-kernel score, so it need not equal the full marginal score even with exact occupancies. Rather, it is the token-score component of the edge-occupancy decomposition, and it is the target approximated by the budgeted FRM estimator in the main text.

To separate the errors without assuming that macro reveals share the one-token edge basis, let $\widehat{g}_B$ be the practical token-score estimate and $k_e = \nabla_\theta \log r^x_\theta(i \mid c, \tilde{x}_S)$. Then

$$\widehat{g}_B - \nabla_\theta \log \pi_\theta(x \mid c) = \left(\widehat{g}_B - g^\star_{\mathrm{tok}}\right) - \sum_e \mu^\star_{\theta,e} k_e.$$

The first term includes finite-frontier coverage, approximate continuation weighting, and the shared-parent approximation in macro reveals. The one-token local probabilities are computed exactly, but the macro proposals and response-conditioned occupancies remain approximate. The second term is the omitted reveal score. This is an error decomposition, not a bound: exact local probabilities alone do not guarantee a small total error. Theorem 3.1 concerns exact complete scores and exact posterior averaging, not variance dominance of $\widehat{g}_B$.

E Example of Oracle Marginal Distribution

Figure 5 shows a reveal-state DAG for a 12-position block from an AceCoder completion. The distribution is concentrated but supports several plausible reveal trajectories, motivating marginalization over reveal orders.

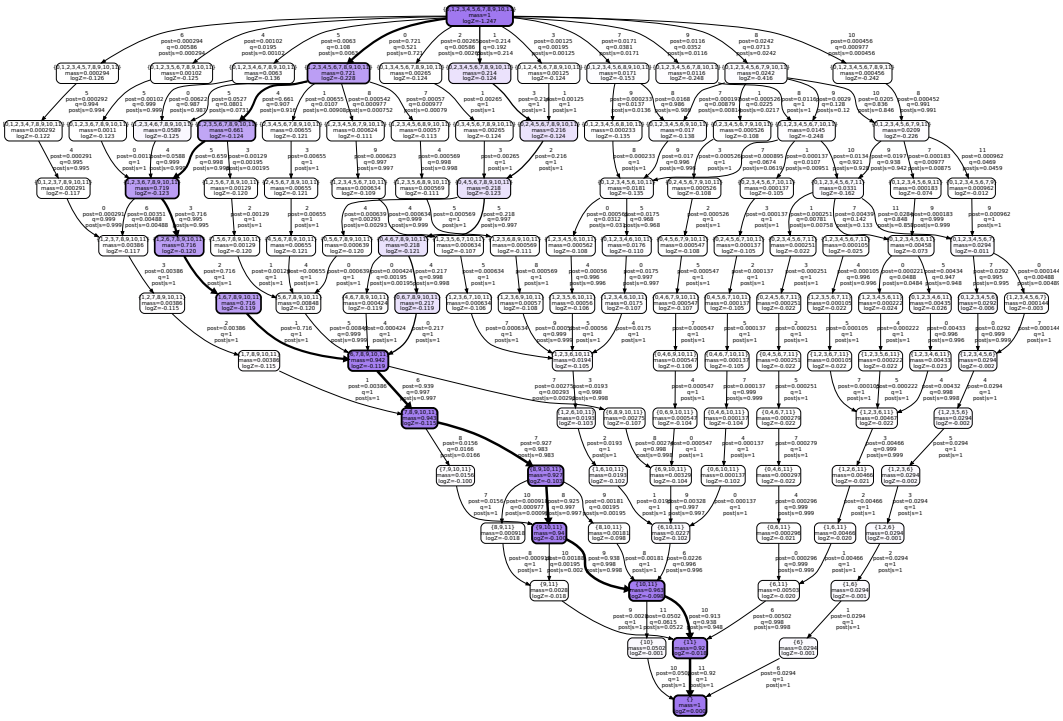

Figure 5: Reveal-state DAG for a 12-position block from an AceCoder completion.

F Frontier Reveal Marginalization Details

This appendix specifies how FRM constructs macro reveals, allocates state evaluations across continuations, and combines their token scores. It uses the local planning distribution in Eq. 6; numerical settings for the experiments are collected in Appendix G.

F.1 Macro-Reveal Construction

Budget-aware reveal size. Let $b_S \geq 1$ be the number of state evaluations assigned to a continuation, including the current evaluation. The nominal reveal count is

$$k_0(S) = \left\lceil \frac{|\mathcal{I}(S)|}{b_S} \right\rceil . \tag{38}$$

This horizon spreads token coverage across the available states. For example, two state evaluations can cover 32 positions with two 16-token macro reveals. Each continuation updates its horizon as positions and evaluations are consumed.

Conditional confidence rankings. After selecting a leading position i, FRM forms a macro reveal from positions likely to follow it in the same parent prediction. For a supported leading position, we condition on both $Z_i = x^i$ and $I = i$. Each competing candidate is drawn independently from its token distribution restricted to candidates that rank below this target, using the CDFs and tie convention in Appendix C. The first k positions of the resulting confidence ranking form the conditional prefix; for $k = 1$, it contains only i.

For an unsplit continuation, the leading position follows $p_{\mathrm{loc},\theta}$. Discarding the sampled prefix's order gives a macro-set with distribution

$$\begin{aligned} q_{k,\theta}(\Delta S \mid S, c, x) = \sum_{i \in \Delta S} & p_{\mathrm{loc},\theta}(i \mid S, c, x) \\ & \cdot \Pr_{\theta}\big(\mathrm{Top}_k(Z) = \Delta S \mid c, \tilde{x}_S, I = i, Z_i = x^i\big) , \qquad |\Delta S| = k, \end{aligned} \tag{39}$$

where the sum includes supported leading positions. We approximate the conditional set probabilities with empirical frequencies from sampled rankings, retaining $p_{\mathrm{loc},\theta}$, renormalized over represented leading positions, as the mixture weights. The CDF calculation supplies the one-token selection probabilities, while the ranking samples supply the additional macro positions.

Coverage and concentration. The fixed-horizon rule uses $k_0(S)$ directly. An optional size controller compares this count with neighboring counts using

$$\mathrm{score}(k) = k \max_{\Delta S:\, |\Delta S| = k} \widehat{q}_k(\Delta S \mid S, c, x), \tag{40}$$

where $\widehat{q}_k$ is the finite-pool version of Eq. 39. The factor k rewards token coverage; the largest set probability favors sizes for which candidate rankings agree. This criterion chooses the size, not the particular set: the macro-set is subsequently sampled. A search radius of one considers $k_0 - 1, k_0, k_0 + 1$, clipped to the eligible range; ties favor the horizon and then the smaller count. Radius zero retains the fixed-horizon rule.

F.2 Frontier Expansion and Contribution Weights

Continuations and allowed reveals. A frontier continuation carries its state S, allowed leading positions, incoming weight $\widehat{\nu}(S)$, and remaining state evaluations b_S. The support restriction is suppressed in this notation. A residual continuation excludes leading choices already represented by other branches; $p_{\mathrm{loc},\theta}$ is restricted and renormalized over that support. This restriction changes which position can lead, not which candidates participate in the parent confidence competition.

For blockwise decoding, each active block receives a share of the response-level budget and starts with incoming weight one. Root branching separates alternative leading reveals at a block's initial state. Local branching splits a continuation at an intermediate state S, retaining the already-revealed positions in both branches. Both use the incoming weight $\widehat{\nu}(S)$ in the contribution rule below, without duplicating earlier scored contributions. The split divides only the continuation's future evaluation budget; branch-count limits and secondary-budget caps remain block-level and do not reset at intermediate states.

Branch admission and budget allocation. Before sampling the scored macro-set, FRM considers separating the most likely allowed leading position i from its complement. With b_S state evaluations assigned, including the current evaluation, the future allocations satisfy

$$b_{\mathrm{main}} + b_{\mathrm{res}} = b_S - 1. \tag{41}$$

For $b_S > 1$, write $p_i = p_{\mathrm{loc},\theta}(i \mid S, c, x)$. The mass-proportional allocation rounds $(b_S - 1)(1 - p_i)$ to the nearest integer and raises it to at least one. This tentative secondary allocation is capped by the unused block-level secondary allowance and the per-split limit, yielding b_{res}. The per-split limit is a configured fraction of $b_S - 1$, rounded down with a minimum of one. The main continuation receives the remaining future evaluations, as in Eq. 41.

A split must respect the branch-count limit, give the residual its configured minimum number of evaluations, and leave at least one evaluation for the main child if it remains unfinished. The residual must have positive weight. Using an absolute threshold in expected state evaluations, admission additionally requires

$$\widehat{\nu}(S)(1 - p_i) b_{\mathrm{res}} \geq \max\{\tau_{\mathrm{branch}}, \lambda\, \widehat{\nu}(S) p_i b_{\mathrm{main}}\}, \tag{42}$$

where $\tau_{\mathrm{branch}} \geq 0$ is the absolute threshold and $\lambda \geq 0$ sets the required fraction of the main continuation's weight-times-budget value. These tests use the evaluated parent probabilities before drawing the scored prefix. If the split is rejected, all $b_S - 1$ future evaluations remain with one sampled continuation.

Weight accounting. Without a split, the leading position and its conditional prefix are sampled from the local proposal. Their selection probabilities are already represented by sampling frequency, so the macro-edge inherits $\widehat{\nu}(S)$, as in Eq. 7. For an admitted split, the leading choice i is explicitly retained and only its conditional prefix is sampled. Its contribution and the complementary continuation's weight are

$$\begin{aligned} \widehat{\mu}_e &= \widehat{\nu}(S)\, p_i, \\ \widehat{\nu}(S) - \widehat{\mu}_e &= \widehat{\nu}(S)\, (1 - p_i). \end{aligned} \tag{43}$$

The residual samples its next leading position from the complementary support. These two contributions partition the incoming weight. Multiplying a sampled macro-edge by its proposal probability again would count that selection probability twice.

Expansion and merging. The child inherits the scored edge's coefficient by Eq. 8; the residual remains a separate continuation at the parent. Pending continuations are prioritized by represented weight and, when a parent distribution is available, its spread:

$$\text{priority}(S) = \frac{\widehat{\nu}(S)}{\sqrt{\sum_{j \in \mathcal{I}(S)} p_{\text{loc},\theta}(j \mid S, c, x)^2}}.$$

Residual continuations retain their original parent distribution and use remaining weight as priority. Continuations with the same state, allowed leading positions, and parent distribution can merge by adding their weights and remaining budgets. Each represented contribution is propagated once. Planning ends when the budget is exhausted or no unfinished continuation remains.

F.3 Scoring and Training Conventions

Shared-parent token scores. A macro-edge records $(S, i, \Delta S)$, retaining the leading position for weight accounting. Every target token in ΔS is scored at the same parent, as defined in §3.3; the next evaluated state includes the whole set. Macro reveals exchange intermediate-state resolution for token coverage. Appendix D separates this approximation and finite frontier coverage from the omitted reveal-order score.

Aggregation and normalization. The token-score objective sums contributions with their raw coefficients from Eq. 7, without a softmax over selected edges. Coefficients are detached during differentiation. Training normalizes the loss by the configured generation length, using the same constant for all responses.

Scoring budget and runtime. The budget B counts scored state evaluations. We report the corresponding runtime and its breakdown in Appendix G.5.

G Experimental Protocols

We summarize the settings for the main-table experiments, offline study, training ablations, and runtime profile. For confidence-based generation, the number of decoding steps equals the generation length.

G.1 Main-Table Training

We train LLaDA-8B-Instruct [32] with 4-bit QLoRA and bfloat16 mixed precision on eight NVIDIA H200 GPUs. The LoRA adapters [19] use rank 128, scaling $\alpha = 64$, and dropout 0.05. Table 2 gives the optimizer settings. We use the training splits of `openai/gsm8k` (`main`) and `ankner/math-500` for GSM8K and MATH, respectively, and the ESPO-preprocessed hard subset of AceCoder-87K for coding. Training rollouts use confidence-based decoding with temperature 0.9, generation length 256, and block length 32. Each prompt has 16 sampled responses; updates average four unique prompts for GSM8K and MATH, and two to four for coding, using gradient accumulation.

GSM8K uses the question directly as its user prompt; MATH requests step-by-step reasoning and a boxed final answer. Both use final-answer correctness rewards without a separate format bonus. Coding follows the ESPO prompt and reward recipe: a system instruction requests a single final Python code block, the user prompt supplies the problem and an example test, and the reward combines a code-format score with the fraction of supplied tests passed. Execution credit requires correctly formatted, syntactically valid code.

The scoring allocation is two evaluated parent states per block for GSM8K and MATH, and 16 for AceCoder, matching the random-masking scoring budget of ESPO.

Table 2: Optimizer settings for the main-table FEROM runs.

	Value
Optimizer	AdamW, $\beta = (0.9, 0.99)$, weight decay 0.1
LR schedule	Constant after warmup; warmup ratio 10^{-4}
Learning rate	3×10^{-6}
Gradient-norm clipping	0.2
KL coefficient	0.0
Clipping threshold ϵ	0.5

G.2 Main-Table Evaluation

We use the JustGRPO-style evaluation pipeline [31] with confidence-based decoding, block length 32, temperature zero, and no classifier-free guidance. Adapters are evaluated on an unquantized bfloat16 base model. We generate one completion per problem and report math accuracy and code pass@1. Baseline scores are taken from their source papers.

Each problem uses the model's chat template. GSM8K supplies the question, and MATH-500 additionally requests a boxed final answer. Code prompts request a complete Python function from the HumanEval specification or the MBPP problem, function stub, and example tests. We grade math answers with the task grader and code by functional correctness, evaluating 1,319 GSM8K, 500 MATH-500, 164 HumanEval, and 500 MBPP problems.

G.3 Offline Gradient-Alignment Study

Data and continuation states. We generate eight 256-token responses per AceCoder prompt for 32 prompts, using LLaDA-8B-Instruct with confidence-based decoding, block length 32, and temperature 0.9. Rewards are supplied-test pass fractions. For prompts with reward variation, we uniformly select a four-response subset whose recomputed group-relative advantages are nonzero. We uniformly select a common block index that is complete before EOS in all source responses, taking each selected response's halfway state with 16 masked positions. This gives 56 continuation cases from 14 prompt groups. All methods share these states, targets, and advantages; zero-advantage prompts remain in aggregate averages.

Marginal reference and scoring. Dynamic programming over all subsets of the remaining positions gives the conditional marginal-gradient reference from the joint token/CDF transition probabilities. It includes both token-prediction and reveal-selection derivatives and is normalized by continuation length. Candidate scores include both components or only token derivatives, with detached coefficients. All gradients use the same LoRA initialization and random projection.

Budget and comparisons. Budgets range from 4 to 32 state evaluations. We compare FEROM with a single path using evenly spaced evaluations, the initial one-token steps of a path, left-to-right scoring, random masking, and two independent complete paths at the largest budget. Planning uses predictions only at evaluated states. Token-plus-order comparisons include methods with supported endogenous reveal transitions.

Metrics and parameter selection. Unweighted cosine averages case-wise gradient cosines. Advantage-weighted cosine compares gradients aggregated first within response groups using their advantages, then across prompts. Relative MSE divides aggregate squared error by the reference's squared norm. At each budget, we tune one FEROM configuration on the displayed cases using token-only advantage-weighted cosine, then reuse it across scoring choices and metrics. Results average 16 paired planner draws; bands show one standard deviation.

G.4 Low-Budget Training Ablations

Training and validation. We train all LLaDA-8B-Instruct parameters in bfloat16 on 17,174 AceCoder examples and validate on 256 disjoint problem families. Each update uses two unique prompts with eight responses each. Rollouts use ESPO-style prompts without truncation, 256

generated tokens, block length 32, and temperature 0.9. Training reward combines supplied-test pass fractions with a format bonus. The scoring budget is 128 state evaluations per response, half of a full trajectory, and the loss is normalized by 256. All arms use conditional-CDF reveal probabilities. AdamW uses the betas and weight decay in Table 2, with constant learning rate 3×10^{-6} and gradient clipping at 0.2.

Validation runs before training and every ten updates using greedy confidence-based decoding at the training generation length. The curves report execution test-pass fractions without the format bonus; training curves use a trailing 50-step mean.

Branching ablation. *Single trajectory (macro)* and `FEROM` use the same fixed-horizon macro reveals, 4,096 conditional ranking samples per evaluated state, and token-only scores. The control follows one path; `FEROM` permits intermediate-state branching with at most two continuations per block and at most 25% of the block's scoring budget assigned to secondary continuations.

Reveal-order scoring ablation. We separately compare *Rollout trajectory* and *Rollout trajectory + order score* on actual one-token rollout transitions. Both sample states across trajectory-depth strata and apply inverse-inclusion weights. Only the second arm adds the conditional reveal-order score to the token score.

G.5 Runtime Measurement Scope

Figure 6 profiles LLaDA-8B-Instruct updates on one RTX A6000 using 4-bit NF4 QLoRA and bfloat16 mixed precision. Each update uses one GSM8K prompt with two 256-token responses, block length 32, temperature 0.9, and a scoring budget of 16 state evaluations per response. FEROM uses CDF-based reveal probabilities and 1,024 conditional ranking samples. All methods use fused token scoring, activation checkpointing, and zero LoRA dropout.

All methods use identical rollouts, and both FEROM implementations share their scoring states and coefficients. We hold the policy fixed with zero learning rate while measuring the complete update, including backward passes, clipping, and optimizer steps. Timings average three updates after one warm-up, with CUDA synchronization and non-overlapping phase measurements; setup, I/O, logging, and communication are excluded. Memory-efficient replay uses separate planning and scoring forwards; shared-forward execution reuses their activations.

H Additional Comparisons with JustGRPO

We additionally compare with JustGRPO [31], following its full-parameter training setup in bfloat16 rather than the 4-bit QLoRA setup of the main-table experiments. We use the released JustGRPO evaluation pipeline with 256-token confidence-based generation.

H.1 End-to-End Training from LLaDA-Instruct

We initialize FEROM from LLaDA-Instruct and train all model parameters with 64 unique prompts per update and a half-trajectory scoring budget, $B/T = 0.5$, where T is the number of one-token updates in a full trajectory. Table 3 compares these results with JustGRPO.

Table 3: **End-to-end training from LLaDA-Instruct.** Accuracy/pass@1 (%) at generation length 256; JustGRPO results are from [31]. Bold marks the best value in each benchmark column.

Method	B/T	GSM8K	MATH-500	HumanEval	MBPP
`FEROM`	0.5	87.12	44.80	48.78	43.40
JustGRPO	1.0	**89.10**	**45.10**	**49.40**	**52.40**

With half a trajectory's scoring budget, FEROM achieves performance close to JustGRPO on both math benchmarks and HumanEval. These runs also improve accuracy on the math benchmarks over the main-table results.

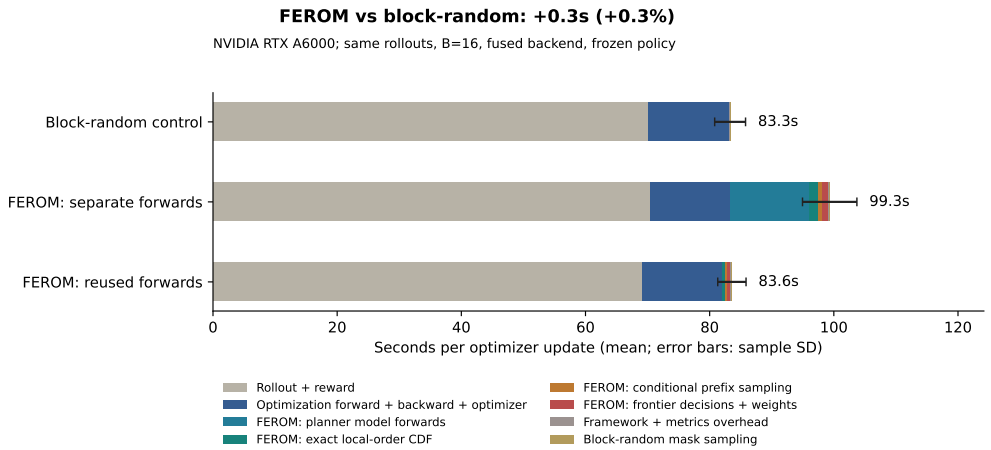

Figure 6: **Computational cost of CDF-based FEROM.** Matched block-random, memory-efficient replay (separate forwards), and shared-forward profiles. Stacked bars separate model work from CDF evaluation, macro-prefix sampling, and frontier bookkeeping. Error bars show sample standard deviations over three measured updates; the shared-forward timing difference is smaller than this variability.

H.2 Continual Training from JustGRPO-Code

Starting from the released JustGRPO-Code checkpoint, we compare paired continuations with native JustGRPO and FEROM using either confidence-based or left-to-right rollouts. All continuations use full-parameter updates, 16 unique prompts with 16 responses each, and a full-trajectory scoring budget. Table 4 reports the completed evaluations.

Table 4: **Continual training from JustGRPO-Code.** Pass@1 (%) at generation length 256. Bold marks the best continuation at each update count.

Continuation	Updates	HumanEval	MBPP
Released checkpoint	0	48.78	53.00
JustGRPO	10	48.17	**53.80**
`FEROM`, confidence rollout	10	50.00	53.20
`FEROM`, left-to-right rollout	10	**50.61**	53.20
JustGRPO	20	**51.22**	**53.80**
`FEROM`, confidence rollout	20	**51.22**	52.20
`FEROM`, left-to-right rollout	20	50.61	53.20

After ten updates, both FEROM continuations improve HumanEval and MBPP over the released checkpoint, with left-to-right rollout yielding the stronger HumanEval result. After twenty updates, confidence-rollout FEROM matches native JustGRPO on HumanEval. These results show that FEROM can further improve the HumanEval performance of a policy already trained with JustGRPO.

Discussion and future work. We hypothesize that the remaining performance gap to JustGRPO mainly reflects limited diversity of high-reward rollouts under LLaDA-style confidence-based decoding. FEROM targets the reveal-order marginal induced by a given decoder; improving the diversity and quality of its rollout proposals calls for co-design of decoding and policy optimization. This co-design is beyond the scope of the present work and will be explored in future work.

I Broader Impact

FEROM is foundational research on policy optimization for masked diffusion language models, with potential benefits for reasoning and code generation. It shares the general downstream benefits and risks of language-model research; this work does not introduce a new application or deployment.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction summarize the marginal-policy formulation, finite-budget estimator, and empirical findings. Sections 3.1–3.4 distinguish the theoretical identity from the practical surrogate.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in §7.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: Theorem 3.1 states the assumptions of the marginal policy-gradient identity; Appendix B provides its proof and variance analysis. Appendices C and D give the reveal-probability and occupancy derivations.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Sections 3.3–3.4 and Appendix F describe the estimator and objective. Appendix G specifies the datasets, training settings, evaluation protocol, and study procedures.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: We provide the code and usage instructions at `https://github.com/ziansu/ferom`, also linked in the abstract. We use public datasets, with their sources and preparation described in Appendix G and the repository.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Appendix G gives data splits, training and rollout settings, optimizer hyperparameters, and evaluation procedures; Appendix H describes the additional JustGRPO comparisons.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Figure 3 reports one standard deviation over repeated planner draws, and Figure 6 reports timing standard deviations over measured updates. Appendices G.3 and G.5 define these repetitions; end-to-end training results do not include independent-run error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix G.1 describes the main-table hardware and training precision. Appendix G.5 and Figure 6 give the profiling hardware, measurement settings, and computational overhead.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our experiments use public models and benchmarks and do not involve human participants or new personal-data collection.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: Appendix I briefly discusses potential benefits and general downstream impacts of this algorithmic work.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: This work studies an optimization algorithm on existing public models and benchmarks, without introducing a new scraped dataset or deployment.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We properly cite the assets we use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We document the FEROM implementation at `https://github.com/ziansu/ferom`, including installation, training configurations, data preparation, and usage instructions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: Crowdsourcing or human subjects are not involved.

    Guidelines:

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

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

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

    Answer: [N/A]

Justification: Crowdsourcing or human subjects are not involved.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLM assistance is limited to routine writing, editing, and coding support, rather than an additional component of the proposed method.

Guidelines:

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