PATH PLANNING FOR MASKED DIFFUSION MODELS WITH APPLICATIONS TO BIOLOGICAL SEQUENCE GENERATION

Fred Zhangzhi Peng,[1*] **Zachary Bezemek,**[2*] **Sawan Patel,**[3] **Sherwood Yao,**[3]
Jarrid Rector-Brooks,[4,5] **Alexander Tong,**[4,5†] **Pranam Chatterjee**[1,6†]

[1]Duke University, Durham, NC, USA
[2]Department of Mathematics, Duke University, Durham, NC, USA
[3]Atom Bioworks, Cary, NC, USA
[4]Mila – Quebec AI Institute, Montréal, Canada
[5]Université de Montréal, Montréal, Canada
[6]Department of Computer Science, Duke University, Durham, NC, USA

*Equal contribution, †Supervising authors

Emails: alexander.tong@mila.quebec, pranam.chatterjee@duke.edu

ABSTRACT

In this paper, we investigate how the order in which tokens are unmasked during masked diffusion models (MDMs) inference affects generative quality. We derive an expanded evidence lower bound (ELBO) that introduces a *planner*, responsible for selecting which tokens to unmask at each step. Our analysis suggests that alternative unmasking strategies can improve generative performance. Based on these insights, we propose *Path Planning (P2)*, a sampling framework that leverages pre-trained BERT or the denoiser itself to guide unmasking decisions. P2 generalizes all known MDM sampling strategies and enables significant improvements across diverse domains including language generation (in-context learning, code generation, story infilling, mathematical reasoning, reverse curse correction) and biological sequence generation (protein and RNA sequences).

1 INTRODUCTION

Inspired by the success of diffusion models in continuous space, recent work has focused on developing effective training algorithms for discrete diffusion models, with absorbing state diffusion emerging as the dominant approach (Austin et al., 2021). This has led to scalable masked diffusion models (MDMs) with simplified objectives (Sahoo et al., 2024; Shi et al., 2024; Gat et al., 2024). However, less attention has been given to inference strategies, raising the question: can novel inference techniques improve generative quality? We answer affirmatively by examining how token unmasking order during MDM inference impacts performance. While uniform unmasking is optimal for a perfect denoiser (see Appendix D.3), empirical evidence shows it is suboptimal for imperfect models (Ou et al., 2024; Shih et al., 2022; Li et al., 2021).

We propose *Path Planning (P2)*, a training-free inference method derived from an expanded ELBO that incorporates a "planner" to select tokens for unmasking—and optionally remasking—at each step. Although uniform unmasking remains optimal for ideal denoisers, non-uniform planners improve generation for real-world models by reweighting the MLM objective based on denoiser performance. P2 leverages pre-trained BERT models or the denoiser itself as effective planners without additional training. Notably, P2 generalizes existing MDM sampling strategies (Table 4) and achieves state-of-the-art results across domains: outperforming a 7B Llama model in math reasoning with a 1B MDM, surpassing ARMs in code generation, enhancing protein design with DPLM (Wang et al., 2024a), and generating RNA sequences with greater structural plausibility than natural sequences.

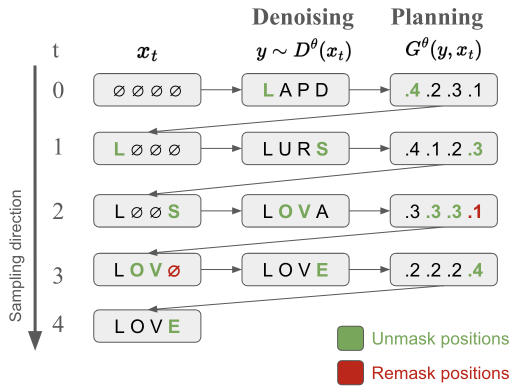

Figure 1: An example of the P2 sampling strategy (see Algorithm D.7). In each step, the denoiser $D^\theta(\cdot)$ makes predictions and the planner $G^\theta(\cdot)$ ranks and selects positions to unmask (green colored) and remask (red colored).

2 Background

Notation We will denote by $S = \{1, ..., N\}$ a finite dictionary of tokens, by $\bar{S} = S \cup \{M\}$ the extension of this dictionary via the addition of some masked state M. For a metric space X, we define by $P(X)$ the space of Borel probability measures on X. When X is finite we endow X with the discrete metric and let $|X|$ denote the cardinality of X. With some abuse of notation we freely identify $\mu \in P(X)$ with a column vector in $[0,1]^{|X|}$ corresponding to the associated probability mass function. We denote by $\delta_x \in P(X)$ the probability measure such that $\delta_x(y) := 1$ if $x = y$ and 0 otherwise and by $\text{Unif}(X) \in P(X)$ the uniform probability measure on X. We suppose that we are interested in generating sequences of length L comprised of elements of S from some data distribution $p_{\text{data}} \in P(S^L)$. We use x_i to denote the i'th coordinate of an elements $x \in \bar{S}^L$, and x^{-i} to denote the element in $\bar{S}^{L-1}$ which is the same as x but with the i'th token removed. For $x \in \bar{S}^L$ and $y \in \bar{S}$, we denote by $x^{-i,y}$ the element in $\bar{S}^L$ which is the same as x but with the i'th token replaced by y. We denote by $M^L := (M, \ldots, M) \in \bar{S}^L$.

2.1 Masked Diffusion Models

In a masked diffusion model, one starts with a a collection of probability mass functions given by, for $y \in \bar{S}^L$ and $t \in [0,1]$:

$$P_t(y; p_{\text{data}}) := \alpha(t) p_{\text{data}}(y) + (1 - \alpha(t))\delta_{M^L}(y) \tag{1}$$

for a monotone-decreasing, continuously differentiable noise scheduler $\alpha : [0,1][0,1]$ with $\alpha(0) = 1$ and $\alpha(1) = 0$, and finds continuous time Markov chain $\overleftarrow{X}_t$ such that $(X_t = x) = P_{1-t}(x; p_{\text{data}})$.

A rate matrix generating $\overleftarrow{X}_t$ is given for $x \neq y \in \bar{S}^L$, by:

$$\overleftarrow{Q}(y, x) = -\frac{\dot{\alpha}(1-t)}{1 - \alpha(1-t)} \sum_{i=1}^{L} \delta_M(x_i) p^i_{data}(y_i | x_{\neq M}) \delta_{y^{-i}}(x^{-i})$$

and $\overleftarrow{Q}(x, x) = \frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)} \sum_{i=1}^{L} \delta_M(x_i)$ (see e.g. Ou et al. (2024) Theorem 1). Here for $z \in \bar{S}^L$, $z_{\neq M}$ denotes the coordinates of z which are not equal to M, and for $i \in \{1, \ldots, L\}$, and $j \in S$:

$$p^i_{data}(j | z_{\neq M}) := p_{data}(\{x : x_i = j\} | z_{\neq M}).$$

One then attempts to approximate $\overleftarrow{X}_t$ with $X_t^{\theta,\text{mask}}$ with transition matrix given for $x \neq y$ by:

$$Q_t^{\theta,\text{mask}}(y, x) := -\frac{\dot{\alpha}(1-t)}{1 - \alpha(1-t)} \sum_{i=1}^{L} \delta_M(x_i) D^\theta_{i,y_i}(x) \delta_{y^{-i}}(x^{-i}). \tag{2}$$

Here we are using the "mean parametrization" of the approximate backwards matrix. That is, we have a neural network parameterized by θ which gives a "denoising function" $D^\theta : \bar{S}^L P(S)^L$, with the hope that

$$D^\theta_{i,\cdot}(x) \approx p^i_{data}(\cdot|x_{\neq M}) \in P(S). \tag{3}$$

In particular, one enforces during inference that $D^\theta_{i,y_i}(x) = \delta_{x_i}(y_i)$ if $x_i \neq M$.

Approximate samples from the data distribution are then obtained via simulating the Markov chain $X^{\theta,\text{mask}}$ with $X_0^{\theta,\text{mask}} = M^L$ to time 1.

3 Path Planning

3.1 Brief Mathematical Formulation

The complete mathematical description is provided in Appendix D.1.

To formulate P2, we modify the jump matrix for the approximate backward process by introducing a planner function $G^\theta : S^L \times \bar{S}^L \to [0,1]^L$, which guides the (re)sampling of tokens during inference. Specifically, $G^\theta_j(y,x)$ estimates the likelihood that the j-th token in a partially denoised sequence x should be (re)sampled based on the denoiser D^θ.

We define:

$$\begin{aligned} F^\theta_j(y,x) \coloneqq\ & \delta_M(x_j)\mathbb{E}_{Y\sim D^\theta(x)}[G^\theta_j(Y^{-j,y_j},x)] \\ & + (1-\delta_M(x_j))\mathbb{E}_{Y\sim D^\theta(x)}[G^\theta_j(Y^{-j,x_j},x)], \end{aligned}$$

which represents the probability of (re)sampling the j-th position, considering both masked and unmasked states.

Next, we define:

$$\hat{D}^\theta_{i,y_i}(x) = D^\theta_{i,y_i}(x)\delta_M(x_i) + \frac{D^\theta_{i,y_i}(x^{-i,M})}{1-D^\theta_{i,x_i}(x^{-i,M})}(1-\delta_M(x_i)),$$

which approximates the probability of unmasking or resampling a token based on the denoiser's output.

The updated transition rate matrix is:

$$Q^\theta_t(y,x) \coloneqq -\frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)}\sum_{i=1}^L F^\theta_i(y,x)\hat{D}^\theta_{i,y_i}(x)\delta_{y^{-i}}(x^{-i}). \tag{4}$$

This formulation leads to an expanded Evidence Lower Bound (ELBO):

$$\begin{aligned} E(x^0) &= E_{MP}(x^0) + E_{UP}(x^0) + E_D(x^0), \\ E_{MP}(x^0) &= -\int_0^1 \frac{\dot{\alpha}(t)}{1-\alpha(t)}\mathbb{E}_{X_t}\Bigg[\sum_{i=1}^L \delta_M([X_t]_i) \\ &\qquad \times \mathbb{E}_Y[\log(G^\theta_i(Y^{-i,x^0_i},X_t))]\Bigg]dt, \\ E_{UP}(x^0) &= -\int_0^1 \frac{\dot{\alpha}(t)}{1-\alpha(t)}\mathbb{E}_{X_t}\Bigg[\sum_{i=1}^L (1-\delta_M([X_t]_i)) \\ &\qquad \times \mathbb{E}_Y[\log(1-G^\theta_i(Y^{-i,x^0_i},X_t))]\Bigg]dt, \\ E_D(x^0) &= -\int_0^1 \frac{\dot{\alpha}(t)}{1-\alpha(t)}\mathbb{E}_{X_t}\Bigg[\sum_{i=1}^L \delta_M([X_t]_i)\log(D^\theta_{i,x^0_i}(X_t))\Bigg]dt. \end{aligned}$$

While $E_D(x^0)$ aligns with the standard masked diffusion ELBO, $E_{MP}(x^0)$ and $E_{UP}(x^0)$ capture the planner's role in guiding the denoising process. The full derivation and additional theoretical insights are provided in Appendix C.5.

3.2 A FAMILY OF PLANNERS: THE P2 SAMPLING STRATEGY

We introduce the P2 sampling strategy, designed for controllable generation by leveraging both the planner G^θ and denoiser D^θ, ensuring convergence to a fully unmasked sequence. The planner is decomposed as:

$$G_j^\theta(y, x) = \delta_M(x_j) G_j^{\theta,M}(y, x) + (1 - \delta_M(x_j))(1 - G_j^{\theta,U}(y, x)),$$

where $G_j^{\theta,M}(y, x)$ predicts the likelihood of unmasking a masked token, and $G_j^{\theta,U}(y, x)$ predicts the likelihood of retaining an unmasked token.

To enhance efficiency, we apply a modified top-k sampling strategy controlled by an unmasking scheduler $\kappa : \{1, \ldots, L\} \to \{1, \ldots, L\}$, a monotone non-decreasing function with $\kappa(L) = L$, ensuring that $\kappa(t)$ tokens are unmasked at iteration t.

We further introduce a stochasticity parameter η to control the remasking frequency:

$$\tilde{G}_j^\eta(x, y) \propto \eta \delta_M(x_j) G_j^M(y, x) + (1 - \delta_M(x_j)) G_j^U(y, x), \tag{5}$$

where increasing η promotes higher remasking rates. This tunable stochasticity aligns with principles from DFM (Campbell et al., 2024), addressing gaps in prior sampling strategies (Shi et al., 2024; Gong et al., 2024; Zheng et al., 2023; Wang et al., 2024a;b; Liu et al., 2024).

The full P2 sampling algorithm is detailed in Algorithm D.7 (Appendix), with an illustrative example in Figure 1. The complete formulation of the P2 sample strategy can be found in Appendix D.2.

4 EXPERIMENTS

In the main text, we show our main protein and RNA generation results. Please find extensive additional results on protein/RNA generation, language generation, code generation, as well as ablation studies in the Appendix H.

4.1 PROTEIN SEQUENCE GENERATION

Setup and Evaluation. We benchmark our P2 method against state-of-the-art protein sequence generation models, including discrete diffusion models (DPLM (Wang et al., 2024a), EvoDiff (Alamdari et al., 2024), and ESM3 (Hayes et al., 2025)), an autoregressive model (ProGen2 (Nijkamp et al., 2022)), and masked language models (ESM2 (Lin et al., 2023)). Each model generates 100 sequences across lengths in $[200, 300, \ldots, 800]$, following their respective sampling strategies, with modifications ensuring fair evaluation. Protein sequence quality is assessed using ESMFold (Lin et al., 2023), measuring foldability through pLDDT, pTM, and pAE scores. We define foldability as the percentage of sequences satisfying pLDDT > 80, pTM > 0.7, and pAE < 10. Additionally, we analyze token entropy and sequence diversity to detect mode collapse. Further details on experimental settings and evaluation metrics are provided in the appendix F.2.

Results. As summarized in Table 1, our P2 algorithm applied to DPLM (150M and 650M) consistently improves all folding metrics—pLDDT, pTM, and pAE—outperforming the default RDM sampling strategy (Zheng et al., 2023). Importantly, this improvement does not compromise token entropy or sequence diversity, highlighting P2's ability to maintain diversity while enhancing quality.

When compared to baselines, including the 2.7B ProGen2-Large autoregressive model and discrete diffusion counterparts ESM3 and EvoDiff, P2 demonstrates remarkable foldability improvements. Visualizations of predicted structures for generated sequences are shown in Figure 2, illustrating P2's ability to generate highly foldable, structurally plausible proteins. These results underscore P2's potential to enable de novo protein design directly from sequence alone. Detailed performance comparisons across sequence lengths are provided in Appendix Figure 6.

4.2 RNA SEQUENCE GENERATION

Experimental Setup. We train a 150M Masked Diffusion Model (MDM) trained on 27M RNA sequences from RNACentral (Petrov, 2021) over 100K steps with a batch size of 320K tokens.

Table 1: Protein Sequence Generation Benchmark. The arrows indicate whether higher (↑) or lower (↓) values are better. See Appendix Figure 6 for a detailed comparison.

Model Name	pLDDT (↑)	pTM (↑)	pAE (↓)	Foldability (%) (↑)	Entropy (↑)	Diversity (%) (↑)
EvoDiff	31.84	0.21	24.76	0.43	**4.05**	93.19
ESM3	34.13	0.23	24.65	1.50	3.99	**93.44**
Progen2-small	49.38	0.28	23.38	4.48	2.55	89.31
Progen2-large	55.07	0.35	22.00	11.87	2.73	91.48
Progen2-medium	57.94	0.38	20.81	12.75	2.91	91.45
DPLM-150M	80.23	0.65	12.07	48.14	3.14	92.80
DPLM-150M + P2	**80.98**	**0.68**	11.43	49.86	3.25	92.63
DPLM-650M	80.02	0.67	11.69	51.86	3.20	91.45
DPLM-650M + P2	80.78	**0.68**	**11.39**	**53.43**	3.24	91.97

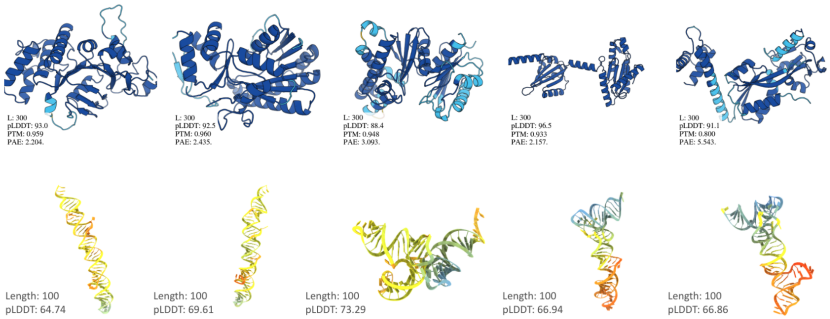

Figure 2: Visualizing the Predicted Structures of Generated Protein (top) and RNA (bottom) Sequences. The protein structures are predicted by ESMFold (Lin et al., 2023) and the RNA structures are predicted by AlphaFold3 (Abramson et al., 2024). See Appendix Figure 9 for more.

We adopted the protein sequence evaluation protocols, using an external folding model (Shen et al., 2024) to estimate structural quality via pLDDT. We additionally calculate the Minimum Free Energy (MFE), GC Content (%), and sequence entropy. We generate 100 RNA sequences of 100 base pairs (bp) each. Visualizations are described in Appendix H.8.6.

Baselines. Two RNA language models, RiNALMo-150M and RiNALMo-650M (Penić et al., 2024), served as primary language model baselines. Additionally, a reference set of 100 native 100-bp RNA sequences was included for comparative purposes. We apply the existing sampling strategies along with the two P2 variants self-planning and BERT-planning (RiNALMo-150M) to the MDM. We evaluated stochasticity parameters ranging from 0 to 2 in 0.02 increments.

Results. As summarized in Table 2, self-planning outperforms native sequences baseline models (RiNALMo), and existing sampling strategies. Employing the RiNALMo planner further improves the key metrics, including pLDDT, predicted minimum free energy (MFE), and GC content with slight compromises in MFE and GC content.

Table 2: RNA Sequence Generation Benchmark. The "Native" row represents subsampled natural RNA sequences. "MDM" refers to a pretrained 150M Masked Diffusion Model trained on RNACentral (Petrov, 2021).

Sequence Source	**pLDDT (↑)**	**MFE (kcal/mol) (↓)**	**Entropy (↑)**	**GC Content (%) (↑)**
Native	48.26	-35.83	**1.96**	49.64
RiNALMo-150M	59.01	-30.12	1.29	29.50
RiNALMo-650M	46.99	-31.90	1.33	28.06
MDM + Ancestral	68.12	-48.46	1.93	60.84
MDM + RDM	67.35	-47.54	1.89	59.42
MDM + P2 (self-planning)	**69.41**	-48.21	1.89	59.84
MDM + P2 + Planner RiNALMo-150M	**73.28**	**-51.91**	1.86	**65.47**

5 CONCLUSION

We demonstrated that unmasking order significantly impacts the generative performance of masked diffusion models (MDMs). By expanding the ELBO formulation, we introduced a *planner* that optimizes token selection during inference. We proposed *Path Planning (P2)*, a sampling framework that generalizes all existing MDM sampling strategies. P2 delivers state-of-the-art improvements across diverse tasks, including language generation and biological sequence design, enabling MDMs to outperform larger autoregressive models. Our findings highlight the importance of inference strategies in discrete diffusion models, paving the way for more efficient and effective sequence generation.

6 ACKNOWLEDGMENTS

We extend sincere gratitude to Jiaxin Shi, Xinyou Wang, Zaixiang Zheng, Chengtong Wang, and Bowen Jing, Kaiwen Zheng for their invaluable insights on DPLM, and likewise extend our gratitude to Jim Nolen for his invaluable insights.

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pp. 1–3, 2024.

Sarah Alamdari, Nitya Thakkar, Rianne van den Berg, Alex X. Lu, Nicolo Fusi, Ava P. Amini, and Kevin Kaichuang Yang. Protein generation with evolutionary diffusion: sequence is all you need. *bioRxiv*, 2024. URL https://api.semanticscholar.org/CorpusID:261893498.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *CoRR*, abs/2107.03006, 2021. URL https://arxiv.org/abs/2107.03006.

Mohammad Bavarian, Heewoo Jun, Nikolas A. Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle. *ArXiv*, abs/2207.14255, 2022. URL https://api.semanticscholar.org/CorpusID:251135268.

Joe Benton, Yuyang Shi, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. From denoising diffusions to denoising markov models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(2):286–301, 01 2024. ISSN 1369-7412. doi: 10.1093/jrsssb/qkae005. URL https://doi.org/10.1093/jrsssb/qkae005.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*, 2023.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sid Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, Franccois Yvon, and Andy Zou. Lessons from the trenches on reproducible evaluation of language models. *ArXiv*, abs/2405.14782, 2024. URL https://api.semanticscholar.org/CorpusID:269982020.

Amarjit Budhiraja and Paul Dupuis. *Analysis and Approximation of Rare Events: Representations and Weak Convergence Methods*, volume 94 of *Probability Theory and Stochastic Modelling*. Springer US, New York, NY, 2019. ISBN 978-1-4939-9577-6 978-1-4939-9579-0. doi: 10.1007/978-1-4939-9579-0. URL http://link.springer.com/10.1007/978-1-4939-9579-0.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models, 2022. URL https://arxiv.org/abs/2205.14987.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and T. Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *ArXiv*, 2024. URL https://api.semanticscholar.org/CorpusID:267523194.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11315–11325, June 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021. URL https://api.semanticscholar.org/CorpusID:239998651.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:52967399.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching, 2024. URL https://arxiv.org/abs/2407.15595.

Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(76)90041-3. URL https://www.sciencedirect.com/science/article/pii/0021999176900413.

Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, dec 1977. ISSN 0022-3654. doi: 10.1021/j100540a008. URL https://doi.org/10.1021/j100540a008. Publisher: American Chemical Society.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models, 2024. URL https://arxiv.org/abs/2410.17891.

Ishaan Gulrajani and Tatsunori Hashimoto. Likelihood-based diffusion language models. *ArXiv*, abs/2305.18619, 2023. URL https://api.semanticscholar.org/CorpusID:258967177.

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf A. Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. *Science*, 0(0):eads0018, 2025. doi: 10.1126/science.ads0018. URL https://www.science.org/doi/abs/10.1126/science.ads0018.

Brian L. Hie, Duo Xu, Varun R. Shanker, Theodora U. J. Bruun, Payton A.-B. Weidenbacher, Shaogeng Tang, and Peter S. Kim. Efficient evolution of human antibodies from general protein language models and sequence information alone. *bioRxiv*, 2022. URL https://api.semanticscholar.org/CorpusID:248151609.

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *10th International Conference on Learning Representations*, 2022.

Jean Jacod and Albert Shiryaev. *Limit theorems for stochastic processes*, volume 288. Springer Science & Business Media, 2013.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and

Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Peter Kerpedjiev, Stefan Hammer, and Ivo L Hofacker. Forna (force-directed rna): Simple and effective online rna secondary structure diagrams. *Bioinformatics*, 31(20):3377–3379, 2015.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2019. URL https://api.semanticscholar.org/CorpusID:202888986.

Xuanlin Li, Brandon Trabucco, Dong Huk Park, Michael Luo, Sheng Shen, Trevor Darrell, and Yang Gao. Discovering non-monotonic autoregressive orderings with variational inference, 2021. URL https://arxiv.org/abs/2110.15797.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574. URL https://www.science.org/doi/abs/10.1126/science.ade2574.

Sulin Liu, Juno Nam, Andrew Campbell, Hannes Stärk, Yilun Xu, T. Jaakkola, and Rafael G'omez-Bombarelli. Think while you generate: Discrete diffusion with planned denoising. *ArXiv*, abs/2410.06264, 2024. URL https://api.semanticscholar.org/CorpusID:273229043.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019. URL https://api.semanticscholar.org/CorpusID:198953378.

Ronny Lorenz, Stephan H Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for molecular biology*, 6: 1–14, 2011.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:264451832.

Ang Lv, Kaiyi Zhang, Shufang Xie, Quan Tu, Yuhan Chen, Ji-Rong Wen, and Rui Yan. Are we falling in a middle-intelligence trap? an analysis and mitigation of the reversal curse. *arXiv preprint arXiv:2311.07468*, 2023.

N. Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. *ArXiv*, abs/1604.01696, 2016. URL https://api.semanticscholar.org/CorpusID:1726501.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text, 2024. URL https://arxiv.org/abs/2410.18514.

Erik Nijkamp, Jeffrey A. Ruffolo, Eli N. Weinstein, Nikhil Vijay Naik, and Ali Madani. Progen2: Exploring the boundaries of protein language models. *Cell systems*, 2022. URL https://api.semanticscholar.org/CorpusID:250089293.

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, 2024. URL https://arxiv.org/abs/2406.03736.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and R. Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *ArXiv*, abs/1606.06031, 2016. URL https://api.semanticscholar.org/CorpusID:2381275.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Rafael Josip Penić, Tin Vlašić, Roland G Huber, Yue Wan, and Mile Šikić. Rinalmo: General-purpose rna language models can generalize well on structure prediction tasks. *arXiv preprint arXiv:2403.00043*, 2024.

Anton I. Petrov. Rnacentral 2021: secondary structure integration, improved sequence search and new member databases. *Nucleic acids research*, 49(D1):D212–D220, 2021.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *preprint*, 2019.

Yinuo Ren, Haoxuan Chen, Grant M. Rotskoff, and Lexing Ying. How discrete and continuous diffusion meet: Comprehensive analysis of discrete diffusion models via a stochastic integral framework, 2024. URL https://arxiv.org/abs/2410.03601.

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

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P. de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models, 2024. URL https://arxiv.org/abs/2412.10193.

Tao Shen, Zhihang Hu, Siqi Sun, Di Liu, Felix Wong, Jiuming Wang, Jiayang Chen, Yixuan Wang, Liang Hong, Jin Xiao, et al. Accurate rna 3d structure prediction using a language model-based deep learning approach. *Nature Methods*, pp. 1–12, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K Titsias. Simplified and generalized masked diffusion for discrete data. *arXiv preprint arXiv:2406.04329*, 2024.

Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive models the right way, 2022. URL https://arxiv.org/abs/2205.13554.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *ArXiv*, abs/2211.16750, 2022. URL https://api.semanticscholar.org/CorpusID:254096040.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melissa Hall Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023. URL https://api.semanticscholar.org/CorpusID:259950998.

Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *Proceedings of the 31th International Conference on Machine Learning*, 2014.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *ArXiv*, abs/2402.18567, 2024a. URL https://api.semanticscholar.org/CorpusID:268063857.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Dplm-2: A multimodal diffusion protein language model. *ArXiv*, abs/2410.13782, 2024b. URL https://api.semanticscholar.org/CorpusID:273403705.

G. George Yin and Qing Zhang. *Continuous-Time Markov Chains and Applications*, volume 37 of *Stochastic Modelling and Applied Probability*. Springer, New York, NY, 2013. ISBN 978-1-4614-4345-2 978-1-4614-4346-9. doi: 10.1007/978-1-4614-4346-9. URL http://link.springer.com/10.1007/978-1-4614-4346-9.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *ArXiv*, abs/2401.02385, 2024. URL https://api.semanticscholar.org/CorpusID:266755802.

Yixiu Zhao, Jiaxin Shi, Lester Mackey, and Scott Linderman. Informed correctors for discrete diffusion models, 2024. URL https://arxiv.org/abs/2407.21243.

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, 2024a. URL https://arxiv.org/abs/2409.02908.

Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Mingying Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *ArXiv*, abs/2409.02908, 2024b. URL https://api.semanticscholar.org/CorpusID:272397565.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. A reparameterized discrete diffusion model for text generation. *ArXiv*, abs/2302.05737, 2023. URL https://api.semanticscholar.org/CorpusID:256826865.

APPENDIX

A REPRODUCIBILITY STATEMENT

We provide the PyTorch implementation in Appendix Section E. For the experiments, we integrate our approach into the SMDM (Gong et al., 2024) GitHub codebase[1] to obtain the results for "MDM (1.1B) + P2" reported in Table 8. Similarly, the results for "DiffuLLaMA (7B) + P2" in Table 8 are derived using the DiffuLLaMA (Nie et al., 2024) GitHub codebase[2]. For the protein sequence generation experiments, we utilize the DPLM (Wang et al., 2024a) open-source codebase[3]. The RNA sequence generation results are obtained by adapting the DPLM codebase for MDM training, combined with the RiNALMo (Penić et al., 2024) language model architecture.

B RELATED WORKS

Masked Diffusion Models (MDMs) represent a promising alternative to autoregressive models for discrete data generation, particularly in language modeling. Recent advancements have focused on simplifying and generalizing the MDM framework to improve performance and training efficiency (Shi et al., 2024; Sahoo et al., 2024). These studies introduced a continuous-time variational objective for MDMs, expressed as a weighted integral of cross-entropy losses, facilitating the training of models with state-dependent masking schedules. At the GPT-2 scale, these MDMs outperformed prior diffusion-based language models and demonstrated superior capabilities in zero-shot language modeling tasks (Nie et al., 2024; Gong et al., 2024).

MDMs generate sequences starting from a fully masked input and progressively unmasking positions until a clean sequence is reached. Once a token is unmasked, it will stay unchanged. However, there is not guarantee that the state is correct, considering the approximation errors arise from the imperfect fit to real-world data distributions. Additionally, time discretization (Zhao et al., 2024) and numerical errors (Zheng et al., 2024a) may further the error incurred during sampling processes.

To address these challenges, several solutions have been proposed. These include methods allowing models to revise prior predictions and guiding sampling trajectories using internal or external knowledge. Examples include informed correctors (Zhao et al., 2024), greedy ancestral methods (Gong et al., 2024), and RDM sampling techniques (Zheng et al., 2023; Wang et al., 2024a), which leverage model scores to replace random masking with targeted corrections. None of these works, however, allow for the use of an external planner, and (Zheng et al., 2023; Wang et al., 2024a) are simply using a top-k sampling strategy without any concern for the theoretical underpinnings of the sampling strategies viability.

In terms of theoretically-backed methods for selecting the denoising order during a generative model's sampling process, the current literature is quite sparse. Shih et al. (2022); Li et al. (2021) discuss this task from the perspective of Any-Order Autoregressive models, with Li et al. (2021) requiring a specially-trained external planner model using a specially designed architecture and Shih et al. (2022) taking the perspective that a fixed family of possible generation orders should be chosen a priori to eliminate redundancy.

The most closely related work to ours is likely the recent DDPD (Liu et al., 2024) introduced a generative process divided into a planner, which identifies corrupted positions, and a denoiser, which refines these positions. Though they discuss the ability to employ a MDM denoiser within their framework, their analysis and sampling is through the lens of uniform discrete diffusion models. In particular, as with Li et al. (2021), the success of their strategy is contingent upon training a large specialized planner model of comparable size to the denoiser itself. Moreover, in their framework, since they are based on uniform diffusion models, the partially de-noised sequence never contains any masked states, and there is no way for the planner to be separated into masked and unmasked components to design a sampling strategy with guaranteed finite-time along the lines of our Algorithm D.7. Given the possible perceived similarity of this concurrent work with ours, we provide a thorough

[1] https://github.com/ML-GSAI/SMDM
[2] https://github.com/HKUNLP/DiffuLLaMA
[3] https://github.com/bytedance/dplm

comparison of DDPD with P2 in Appendix D.4, highlighting the greater flexibility and difference in role of P2s' planners.

C Additional Background

C.1 Discrete Diffusion/Flow Models: Problem Setup

Here we discuss the general formulation of the problem setup and motivation behind discrete diffusion Austin et al. (2021); Lou et al. (2023); Sun et al. (2022); Campbell et al. (2022) and discrete flow models Campbell et al. (2024); Gat et al. (2024). This helps contextualize this manuscript in the broader landscape of the generative modeling framework, as well as introduce some additional notation that will be useful for the mathematical derivations in Appendix D.

Suppose we have a set if N tokens, $S = \{1, ..., N\}$, and samples of sequences of length L comprised of elements of S from some distribution $p_{data} \in P(S^L)$. We seek to generate new samples from p_{data} via learning a "denoising" function D^θ which allows one to sample from $p^\theta \approx p_{data}$.

To find such a function, we choose a family of probability measures $\{P_t(\cdot;\mu)\}_{t\in[0,1],\mu\in P(S^L)}$ such that $P_0(\cdot;\mu) = \mu$ and $P_1 = \pi$, where $\pi \in P(S^L)$ is some easily-sampled from reference distribution. Then we find $\{\overleftarrow{X}_t\}_{t\in[0,1]}$ a continuous-time Markov chain with $(\overleftarrow{X}_t = x) = \overleftarrow{P}(x; p_{data}) := P_{1-t}(x; p_{data})$, and seek to use the "denoising function" D^θ to simulate a continuous time Markov chain $\{X_t^\theta\}_{t\in[0,1]}$ which is close in distribution to $\overleftarrow{X}$. In the end, we will have that taking $X_0^\theta \sim \pi$ and simulating the chain to time 1, $X_1^\theta \overset{d}{\approx} \overleftarrow{X}_1 \sim p_{data}$. To understand what this process X^θ is and why the use of this intermediary Markov chain is useful for finding a choice of D^θ, we first briefly review the theory of continuous time Markov chains in Appendix C.2.

C.2 Time-Inhomogeneous Continuous Time Markov Chains (CTMC)

A (time-inhomogenous) continuous-time Markov chain $\{X_t\}_{t\geq 0}$ on a finite set X is a stochastic process satisfying the Markov property, which can be formally summarized $(X_t = y|X_{s_1} = x_1, ..., X_{s_k} = x_k, X_s = x) = (X_t = y|X_s = x), \forall y, x_1, ..., x_k, x \in X, 0 \leq s_1 < s_2 < ... < s_k < s < t \leq 1$. One can construct such a process by specifying a "rate matrix" $Q_t \in \mathbb{R}^{|X|\times|X|}$ with $Q_t(y, x) > 0$ and $Q_t(x, x) = -\sum_{y\neq x} Q(y, x)$ for all $x \neq y \in X$ and $t \geq 0$. Along with an initial distribution $\mu \in P(X)$, Q determines the 1-dimensional time marginals $(X_t = \cdot) \in P(X)$ via the Kolmogorov equation:

$$\begin{aligned} \frac{d}{dt}(X_t = \cdot) &= Q_t(X_t = \cdot), \qquad t \geq 0 \\ (X_0 = x) &= \mu(x), \qquad x \in X. \end{aligned} \tag{6}$$

When the above holds, we will say Q "generates" X. Note that one can see necessarily that if Q generates X, $Q_t(y, x) := \lim_{s\downarrow t} \frac{d}{ds}(X_s = y|X_t = x), x \neq y \in X$. Knowing the entries of Q also provides a means of generating samples from X_t at any given time, since paths of $\{X_t\}_{t\geq 0}$ can be realized via a sequence of jump times $\{\tau_n\}_{n\in N}$, with $\tau_i = \inf\{t > \tau_{i-1} : X_t \neq X_{\tau_{i-1}}\}$ and the effective discrete-time jump process $\{X_{\tau_i}\}_{i\in N}$. Then

$$(X_{\tau_{i+1}} = y|X_{\tau_i} = x, \tau_i = t) = -\frac{Q_t(y, x)}{Q_t(x, x)}, \tag{7}$$

and

$$\log((\tau_{i+1} > t|X_{\tau_{i+1}} = x, \tau_i = s)) = \int_s^t Q_p(x, x)dp.$$

For more background on time-inhomogenous continuous-time Markov chains, see e.g. Chapter 2 of Yin & Zhang (2013) or the appendix of Ren et al. (2024).

C.3 The Role of the Denoiser and the Approximate Backwards Process

In the "discrete diffusion model" framework, one in fact starts with specifying a rate matrix Q_t generating some Markov chain $\{X_t\}_{t\in[0,1]}$ with $X_0 \sim p_{data}$ and $X_1 \sim \pi$ and defines $P_t(x; p_{data}) :=$

$(X_t = x)$. $\overleftarrow{X}_t$ is then simply defined as X_{1-t}, and a rate matrix $\overleftarrow{Q}_t$ which generates $\overleftarrow{X}$ can be found from Q_t via an application of Bayes' rule (see Prop. 3.2 in Sun et al. (2022)). In the "Discrete Flow Model" framework, one instead starts with a desired interpolation $P_t(\cdot; p_{data})$ between p_{data} and π, and constructs a rate matrix $\overleftarrow{Q}_t$ generating a $\overleftarrow{X}_t$ with one-dimensional time marginals $\overleftarrow{P}_t(\cdot; p_{data})$ a posteriori.

As explained above, in order to generate samples of $\overleftarrow{X}_t$ at a given time (and in particular of $\overleftarrow{X}_1 \sim p_{data}$), it is sufficient to have access to the entries of $\overleftarrow{Q}_t$. In both settings, however, the entries of $\overleftarrow{Q}_t$ will naturally depend on the unknown distribution p_{data}, and hence, using the form of this dependence, a the denoiser function D^θ is constructed in an attempt to approximate these unknown quantities. This results in a rate matrix $Q_t^\theta \approx \overleftarrow{Q}_t$, which generates the approximate backwards Markov chain $\{X_t^\theta\}_{t\in[0,1]}$. The distribution of the output of the resulting sampling scheme is then

$$p^\theta = P_1^\theta = (X_1^\theta = \cdot).$$

The form of the denoiser, as well as the choice of $P_t, \overleftarrow{Q}$, and Q^θ in our particular setup are introduced in Sections 2.1 and 2.1.

C.4 The Conditional Backwards Process

A pervasive assumption made in the literature is that for any fixed $x^0 \in S^L$,

$$P_t(y; \delta_{x^0}) = \prod_{i=1}^{L} p_t(y_i|x_i^0) \tag{8}$$

for a family of probability measures $\{p_t(\cdot|x_i^0)\}_{t\in[0,1]} \subset P(S)$. We denote by $\overleftarrow{X}^{x^0}$ the "conditional backwards process," on the point x^0, defined as the Markov chain with distribution $(\overleftarrow{X}_t^{x_0} = y) = \overleftarrow{P}(y; \delta_{x^0})$, and by $\overleftarrow{Q}^{x^0}$ its rate matrix. The coordinates $(\overleftarrow{X}_1^{x_0}, ..., \overleftarrow{X}_L^{x_0})$ of $\overleftarrow{X}^{x_0}$ are thus assumed independent, and each described by a continuous-time Markov chain $\{\overleftarrow{x}_t^i\}_{t\in[0,1]}$ with rate matrix $\overleftarrow{\hat{Q}}_t^{x_i^0} \in \mathbb{R}^{N\times N}$ for $i = 1, ..., L$, $t \in [0, 1]$ that yields $(\overleftarrow{x}_t^i = y_i) = \overleftarrow{p}_t(y_i|x_i^0)$ for all $t \in [0, 1]$ and $y_i \in S$. The hope in making this assumption is that each coordinate of $X_t^\theta \approx \overleftarrow{X}_t$ will be able to be simulated independently in parallel without introducing significant error Sun et al. (2022).

$P_t(y; \mu)$ is taken to be linear in μ, so we have $P_t(y; p_{data}) = \sum_{x\in S^L} P_t(y; \delta_x)p_{data}(x)$, and hence specifying $p_t(j|i), i, j \in S$ is what ultimately what determines the form of $\overleftarrow{Q}_t$ and hence the functions needed to be approximated by D^θ in order to construct Q^θ. The most common choices explored this far in the literature are the "uniform diffusion," Lou et al. (2023); Schiff et al. (2024) which sets

$$p_t(j|i) = \alpha(t)\delta_i(j) + \frac{1-\alpha(t)}{S} \tag{9}$$

for $\alpha : [0, 1] \to [0, 1]$ with $\alpha(0) = 1, \alpha(1) = 0$ and the "masked diffusion," which is out subject of focus.

Note that in the Discrete Diffusion Model framework, $p_t(j|i)$ is not always defined explicitly, and is often implicitly prescribed by asserting the "forward noising" process is the independent evolution of a CTMC on S with rate matrix $\hat{Q}_t \in \mathbb{R}^{N\times N}$ on each coordinate (see e.g. Equations (15) and (16) in Lou et al. (2023)). $p_t(j|i)$ is then found by solving equation 6 with $Q = \hat{Q}$ and $\mu = \delta_i$.

In the case of a "masked diffusion model," one extends S to $\bar{S} = S \cup \{M\}$ for M some "masked state" outside the dictionary of tokens S, and takes:

$$p_t(j|i) = \alpha(t)\delta_i(j) + (1-\alpha(t))\delta_M(j), \quad i, j \in \bar{S} \tag{10}$$

for a monotone-decreasing, continuously differentiable noise scheduler $\alpha : [0,1][0,1]$ with $\alpha(0) = 1$ and $\alpha(1) = 0$. This choice of forward/noising process has been seen to outperform the uniform diffusion process Schiff et al. (2024) as well as other choices of p_t Austin et al. (2021) consistently among applications. If corresponds to the coordinate-wise forward matrix $\hat{Q}_t(j,i) = \sigma(t)\delta_M(j)(1 - \delta_M(i)), i \neq j \in \bar{S}$ with $\sigma(t) = -\frac{d}{dt}\log(\alpha(t))$, and through equation 8 yields equation 1.

In the masked-diffusion setting, both the Discrete Flow Model and Discrete Diffusion Model framework use the rate matrices for the conditional reversed process' coordinates (Campbell et al. (2024) Appendix F.1.):

$$\overleftarrow{Q}_t^{x_i^0}(j,i) = -\frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)}\delta_M(i)\delta_{x_i^0}(j), \quad i,j \in \bar{S}.$$

The resulting conditional rate matrix generating $\overleftarrow{X}_t^{x^0}$ is then, for $x \neq y \in \bar{S}^L$:

$$\overleftarrow{Q}_t^{x^0}(y,x) = -\frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)}\sum_{i=1}^{L}\delta_M(x_i)\delta_{x_i^0}(y_i)\delta_{y^{-i}}(x^{-i}) \tag{11}$$

with $\overleftarrow{Q}_t^{x^0}(x,x) = \frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)}\sum_{i=1}^{L}\delta_M(x_i)$.

C.5 Role of the ELBO

The training objective in general is obtained via the same methodology in both the Discrete Flow and Discrete Diffusion Model framework - in fact this methodology can also be used for continuous diffusion models and denoising processes described by more general Markovian dynamics Benton et al. (2024).

We seek to minimize the KL divergence:

$$\begin{aligned} D_{KL}(p_{data}||P_1^\theta) &= \sum_{x\in S^L} p_{data}(x)\log\left(\frac{p_{data}(x)}{P_1^\theta(x)}\right) \\ &= \sum_{x\in S^L} p_{data}(x)\log p_{data}(x) - \sum_{x\in S^L} p_{data}(x)\log(P_1^\theta(x)). \end{aligned}$$

The first term, the entropy of p_{data} is constant in θ, and so we turn our attention to finding an "Evidence Based Lower Bound"

$$E(x^0) \leq \log(P_1^\theta(x^0))$$

for each fixed $x^0 \in S^L$.The loss that we seek to minimize will is then defined as:

$$L_E \coloneqq -\sum_{x\in S^L} p_{data}(x)E(x). \tag{12}$$

Letting $\mathbb{P}^x \in P(D([0,1];S^L))$ denote the Law (on the Skorokhod space of all cádlág paths from $[0,1]$ to S^L) of $\overleftarrow{X}^{x^0}$ and $\mathbb{P}^\theta \in P(D([0,1];S^L))$ the same but for X^θ, we have, by the data-processing inequality (see, e.g. Budhiraja & Dupuis (2019) Lemma 2.4 (f)):

$$\log(P_1^\theta(x^0)) = -D_{KL}(\delta_{x^0}||P_1^\theta) \geq -D_{KL}(\mathbb{P}^{x^0}||\mathbb{P}^\theta) \coloneqq E(x^0),$$

That is, in order to make sure the approximate reverse process has the desired terminal distribution, by minimizing L_E we attempt to make it so that the entire path of the approximate reverse process matches that of the exact one.

$E(x^0)$ can be found via an application of Girsanov's Theorem for Markov Jump processes (see e.g. Theorem III.5.34 in Jacod & Shiryaev (2013) for a general result or Ren et al. (2024) Theorem 3.3 for the specific Markov Chain setting), and is expressed solely in terms of $\overleftarrow{Q}^{x^0}$, D^θ, and $P_t(\cdot;\delta_x)$.

In the masked diffusion setting, where Q^θ is given by $Q^{\theta,\text{mask}}$ from equation 2 and $\overleftarrow{Q}^{x^0}$ is given by equation 11, this expression is (Sahoo et al. (2024) Equation (10)):

$$E_{mask}(x^0) = -\int_0^1 \frac{\dot{\alpha}(t)}{1-\alpha(t)} E_{X_t \sim P_t(\cdot;\delta_{x^0})} \left[\sum_{i:[X_t]_i = M} \log(D^\theta_{i,x^0_i}(X_t)) \right] dt, \tag{13}$$

with P_t as in equation 1. This is exactly E_D from Proposition D.1.

D Mathematical Details

D.1 Complete Mathematical Formulation

In order to formulate P2 we begin by modifying the jump matrix for the approximate backwards process (equation 2), introducing a new function $G^\theta : S^L \times \bar{S}^L[0,1]^L$, which we refer to as the planner. $G^\theta_j(y,x)$ approximates the likelihood that the j'th token in a partially denoised sequence $x \in \bar{S}^L$ should be (re)sampled given the conditional information about the rest of the sequence x and of the clean data y as predicted by D^θ. In Section H.1, we discuss potential choices of planners and how previous works fall into this general framework.

We next define $F^\theta : \bar{S}^L \times \bar{S}^L[0,1]^L$ by

$$\begin{aligned} F^\theta_j(y,x) \coloneqq\ & \delta_M(x_j) E_{Y \sim D^\theta(x)}[G^\theta_j(Y^{-j,y_j}, x)] \\ & + (1-\delta_M(x_j)) E_{Y \sim D^\theta(x)}[G^\theta_j(Y^{-j,x_j}, x)] \end{aligned}$$

where here we use the shorthand $Y \sim D^\theta(x)$ to mean $Y \sim \otimes_{i=1}^L D^\theta_{i,\cdot}(x)$.

Via our interpretation of the role of G^θ, $F^\theta_j(y,x)$ gives the likelihood that the j'th position of x should be (re)sampled given the information about the rest of the sequence x and the data's j'th token via averaging out the information provided about the rest of the data's tokens from D^θ.

Finally, we define

$$\hat{D}^\theta_{i,y_i}(x) = D^\theta_{i,y_i}(x)\delta_M(x_i) + \frac{D^\theta_{i,y_i}(x^{-i,M})}{1 - D^\theta_{i,x_i}(x^{-i,M})}(1-\delta_M(x_i)).$$

That is, when x_i is not masked $\hat{D}^\theta_{i,y_i}(x)$ approximates the probability that the i'th token of x should be unmasked to y_i given the conditional information about the unmasked tokens in x, and when x^i is not masked, $\hat{D}^\theta_{i,y_i}(x)$ approximates the probability that i'th token of x should be resampled to a value other than x_i, given the conditional information about the unmasked tokens in x other than x_i.

We now seek to modify $Q^{\theta,\text{mask}}$ from equation 2 in a way so that F^θ - by way of the planner G^θ - plays the role of selecting which position should be unmasked/resampled and $\hat{D}^\theta$ plays the role of choosing what it should be (re)sampled to.

For $x \neq y \in \bar{S}^L$, we thus set:

$$Q^\theta_t(y,x) \coloneqq -\frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)} \sum_{i=1}^L F^\theta_i(y,x)\hat{D}^\theta_{i,y_i}(x)\delta_{y^{-i}}(x^{-i}). \tag{14}$$

For reference, we provide a computationally viable Gillespie sampling method Gillespie (1977; 1976) which approximates samples from X^θ with jump matrix Q^θ and provides intuition for the role of the Planner is given by Algorithm D.5 in Appendix D.5.

Observing Algorithm D.5, we see that P2 allows for the planner G^θ to guide the denoising process towards a more optimal path of denoising orders using the information from both the partially noised sequence x_t and the predicted clean sequence y from the denoiser, and further introduces the ability to resample previously masked tokens using information from both the partially generated sequence and the output of the denoiser.

The interpretation of the Planner as a mechanism for guiding the denoising process toward an optimal path is furthered by the following: Define $P_1^\theta \in P(S)$ by $P_1^\theta(x) = (X_1^\theta = x)$, where X^θ is the CTMC with rate matrix given in Equation equation 14. Then we have an "Evidence Based Lower Bound" $E(x^0) \leq \log(P_1^\theta(x^0))$ for each fixed $x^0 \in S^L$ given by $E(x^0) = E_{MP}(x^0) + E_{UP}(x^0) + E_D(x^0)$, where:

$$
\begin{aligned}
E_{MP}(x^0) = &-\int_0^1 \frac{\dot{\alpha}(t)}{1-\alpha(t)} E_{X_t \sim P_t(\cdot;\delta_{x^0})}\Bigg[\sum_{i=1}^L \delta_M([X_t]_i) \\
&\times E_{Y\sim D^\theta(X_t)}[\log(G_i^\theta(Y^{-i,x_i^0}, X_t))]\Bigg] dt \\
E_{UP}(x^0) = &-\int_0^1 \frac{\dot{\alpha}(t)}{1-\alpha(t)} E_{X_t \sim P_t(\cdot;\delta_{x^0})}\Bigg[\sum_{i=1}^L (1-\delta_M([X_t]_i)) \\
&\times E_{Y\sim D^\theta(X_t)}[\log(1-G_i^\theta(Y^{-i,x_i^0}, X_t))]\Bigg] dt \\
E_D(x^0) = &-\int_0^1 \frac{\dot{\alpha}(t)}{1-\alpha(t)} E_{X_t \sim P_t(\cdot;\delta_{x^0})}\Bigg[\sum_{i=1}^L \delta_M([X_t]_i) \\
&\times \log(D_{i,x_i^0}^\theta(X_t))\Bigg] dt.
\end{aligned}
$$

Here P_t is defined per equation 1.

This ELBO offers a simple interpretation, recalling we seek to maximize the expected value of each term with respect to $x_0 \sim p_{data}$. $E_{MP}(x^0)$ optimizes the role of the Planner as it pertains to masked tokens in a partially denoised sequence. That is, as a mechanism for selecting the a viable masked position to insert a "clean" token as suggested by D^θ. If D^θ suggests to unmask the coordinate i to a value which is representative of the data distribution, then G_i^θ should be large so that the i'th position is selected. $E_{UP}(x^0)$ optimizes the role of the Planner as it pertains to unmasked tokens in a partially denoised sequence. That is, as a mechanism for selecting the an unmasked token to resample via remasking and inserting back into D^θ. If the i'th token already contains a token which is representative of the data distribution, then G_i^θ should be small, so that the i'th token remains in the sequence. $E_D(x^0)$ is the the ELBO used for the denoiser of a standard masked diffusion model (see equation 13).

It is worth observing that $E(x^0) \leq E_D(x^o)$, so our ELBO is necessarily a worse lower bound than that arrived at via a standard masked diffusion model. One can observe that setting $G_i^\theta(y, x) = \delta_M(x)$, $E_{MP}(x) = E_{UP}(x) = 0$, and a standard masked diffusion model is recovered. However, the ELBO is only a bound on the KL divergence between the true data distribution and the approximate one (see the discussion in Appendix C.5). Moreover, our ELBO provides a mathematically-backed methodology for assessing when a choice of pretrained model may serve as an effective planner for a given denoiser. In Table 6, we show that planners ranging from 8M to 3B parameters have similar ELBO and thus have similar generation performance (Figure 4). Lastly, it provides a methodology for training a Planner for a given denoiser, or training both in tandem, in a principled way. Training models for this specific purpose is an interesting avenue for future research.

D.2 A Family of Planners: The Complete P2 Sampling Strategy

Here we introduce the P2 sampling strategy, which allows for controllability over the role of the planner, exploitation of the information provided about all tokens in the sequence from G^θ and D^θ, and guaranteed convergence of the sampling procedure to a fully unmasked sequence.

We decompose the planner into two components:

$$
\begin{aligned}
G_j^\theta(y, x) = &\ \delta_M(x_j) G_j^{\theta,M}(y, x) \\
&+ (1-\delta_M(x_j))(1 - G_j^{\theta,U}(y, x)).
\end{aligned}
$$

That is, the "masked token planner" $G_j^{\theta,M}(y, x)$ predicts the liklihood that a masked token at the j'th position should be unmasked, and the "unmasked token planner" $G_j^{\theta,U}(y, x)$ predicts the likelihood that an unmasked token at the j'th position should be kept.

We then employ a modified "top k" sampling strategy, which introduces the possibility of changing multiple tokens per iteration and better exploits the information provided by the scheduler. We define $\kappa : \{1, ..., L\}\{1, ..., L\}$ to be any monotone non-decreasing function with $\kappa(L) = L$, which will serve as an "unmasking scheduler" for how many tokens should be denoised at a given time step. In particular, at the t'th iteration, $\kappa(t)$ tokens are guaranteed to be unmasked in the partially generated sequence.

We further introduce a stochasticity strength parameter η, and define the family of probability measures:

$$\tilde{G}_j^\eta(x, y) \propto \eta\delta_M(x_j)G_j^M(y, x) + (1 - \delta_M(x_j))G_j^U(y, x) \tag{15}$$

for $\eta \geq 0$. Note that while the Planner G_j^θ determines if the j'th token is a valid candidate to change (a masked token to an unmasked one or vice versa), $\tilde{G}_j^\eta$ determines whether the j'th token is valid to be unmasked (or kept unmasked if it already is). As η increases, we will keep fewer unmasked tokens, so the frequency of remasking increases. Tuning η allows us to control the stochasticity (frequency of remasking) of the sampling process as proposed in DFM (Campbell et al., 2024), which is overlooked in existing sampling strategies (Shi et al., 2024; Gong et al., 2024; Zheng et al., 2023; Wang et al., 2024a;b; Liu et al., 2024).

Letting $\text{TopPos}_k(v)$ return the indices of the largest k values in a non-negative vector v, our sampling algorithm is given in Algorithm D.7 in the Appendix. See also Figure 1 for a diagram exhibiting a toy example of generation with P2 Sampling.

D.3 Equivalence of MDMs with AOARMs

Here, for completeness, we recall the connection between Masked Diffusion Models and Any-Order Autoregressive Models Uria et al. (2014); Hoogeboom et al. (2022) as described in Zheng et al. (2024a); Ou et al. (2024). We start by providing a simplified derivation of the equivalence of the two types of models' sampling schemes.

We begin by obtaining the diagonals for the matrix equation 2. Recalling $D^\theta_{i,y_i}(x) = \delta_{x_i}(y_i)$ if $x_i \neq M$, and $\sum_{y_i=1}^N D^\theta_{i,y_i}(x) = 1$ if $x_i = M$:

$$\begin{aligned}
-\sum_{y \neq x} Q_t^\theta(y, x) &= \frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)} \sum_{i=1}^L \delta_M(x_i) \sum_{y_i \neq x_i} D^\theta_{i,y_i}(x) \\
&= \frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)} \sum_{i=1}^L \delta_M(x_i) \sum_{y_i=1}^N D^\theta_{i,y_i}(x) \\
&= \frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)} \sum_{i=1}^L \delta_M(x_i).
\end{aligned}$$

Then, if one considers the effective jump chain's transition probabilities as described in equation 7, we have, for $x \neq y$:

$$(X^{\theta,\text{mask}}_{\tau_{k+1}} = y | X^{\theta,\text{mask}}_{\tau_k} = x, \tau_k = t) = (X^{\theta,\text{mask}}_{\tau_{k+1}} = y | X^{\theta,\text{mask}}_{\tau_y} = x) = \frac{\sum_{i=1}^L \delta_M(x_i) D^\theta_{i,y_i}(x) \delta_{y^{-i}}(x^{-i})}{\sum_{i=1}^L \delta_M(x_i)}.$$

We note that this is zero when the Hamming distance $d_{HAM}(x, y) \neq 1$.

Then, for any $j \in \{1, ..., N\}$:

$$
\begin{aligned}
([X^{\theta,\text{mask}}_{\tau_{k+1}}]_j \neq [X^{\theta,\text{mask}}_{\tau_{k+1}}]_j | X^{\theta,\text{mask}}_{\tau_k} = x, \tau_k = t) &= \sum_{y \in \bar{S}^L : y_j \neq x_j} ([X^{\theta,\text{mask}}_{\tau_{k+1}}]_j = y | X^{\theta,\text{mask}}_{\tau_k} = x) \\
&= \sum_{y \in \bar{S}^L : y_j \neq x_j} \frac{\sum_{i=1}^L \delta_M(x_i) D^\theta_{i,y_i}(x) \delta_{y^{-i}}(x^{-i})}{\sum_{i=1}^L \delta_M(x_i)} \\
&= \sum_{y_j \neq x_j} \frac{\delta_M(x_j) D^\theta_{j,y_j}(x)}{\sum_{i=1}^L \delta_M(x_i)} \\
&= \frac{\delta_M(x_j) \sum_{y_j=1}^N D^\theta_{j,y_j}(x)}{\sum_{i=1}^L \delta_M(x_i)} \\
&= \frac{\delta_M(x_j)}{\sum_{i=1}^L \delta_M(x_i)}
\end{aligned}
$$

and, for x such that $x_j = M$:

$$
\begin{aligned}
&([X^{\theta,\text{mask}}_{\tau_{k+1}}]_j = y'_j | X^{\theta,\text{mask}}_{\tau_k} = x, \tau_k = t, [X^{\theta,\text{mask}}_{\tau_{k+1}}]_j \neq [X^{\theta,\text{mask}}_{\tau_{k+1}}]_j) \\
&= \frac{([X^{\theta,\text{mask}}_{\tau_{k+1}}]_j = y'_j, [X^{\theta,\text{mask}}_{\tau_{k+1}}]_j \neq [X^{\theta,\text{mask}}_{\tau_{k+1}}]_j) | X^{\theta,\text{mask}}_{\tau_k} = x, \tau_k = t)}{([X^{\theta,\text{mask}}_{\tau_{k+1}}]_j \neq [X^{\theta,\text{mask}}_{\tau_{k+1}}]_j) | X^{\theta,\text{mask}}_{\tau_k} = x, \tau_k = t)} \\
&= \frac{\sum_{i=1}^L \delta_M(x_i)}{\delta_M(x_j)} \sum_{y \in \bar{S}^L : y_j = y'_j \neq x_j} ([X^{\theta,\text{mask}}_{\tau_{k+1}}]_j = y | X^{\theta,\text{mask}}_{\tau_k} = x) \\
&= \sum_{y \in \bar{S}^L : y_j = y'_j \neq x_j} \sum_{i=1}^L \delta_M(x_i) D^\theta_{i,y_i}(x) \delta_{y^{-i}}(x^{-i}) \\
&= \delta_M(x_j) D^\theta_{j,y'_j}(x) \\
&= D^\theta_{j,y'_j}(x).
\end{aligned}
$$

Defining for $x \in \bar{S}^L$, $M(x) := \{j \in \{1, \ldots, L\} : x_j = M\}$, the corresponding Gillespie sampling scheme Gillespie (1977; 1976) for a standard masked diffusion model is thus as follows: [h] Gillespie Sampler for Masked Diffusion Models

Initialize: $x_0 \leftarrow (M, M, \ldots, M)$, denoiser D^θ
for $t = 1 : L$ **do**
  Choose Random Coordinate for Unmasking:
  Sample dimension $d' \sim \text{Unif}\left(M(x_t)\right)$
  Denoise:
  Sample $y_{d'} \sim D^\theta_{d',\cdot}(x_t)$
  $[x_{t+1}]_{d'} \leftarrow y_{d'}$
end for
return x_L

Letting $\mathbb{S}_L$ be the set of all permutations of $\{1, ..., L\}$, we then have:

$$
\begin{aligned}
(X^{\theta,\text{mask}}_1 = x) &= \frac{1}{L!} \sum_{\sigma \in \mathbb{S}_L} \prod_{i=1}^L D^\theta_{\sigma(i), x_{\sigma(i)}}(x^{-\sigma(\geq i), M}) \\
&= E_{\sigma \sim \text{Unif}(\mathbb{S}_L)} \left[(X^{\theta,\text{mask}}_1 = x | \sigma) \right]
\end{aligned}
$$

where $x^{\sigma(<i),M} \in \bar{S}^L$ is x but with $x_{\sigma(j)} = M, \forall j \geq i$. Here $\sigma(i)$ represents the coordinate which is unmasked at time τ_i. From this it is clear that with each unmasking, D^θ is gaining additional conditional information about the sequence it is denoising, and could potentially benefit from backtracking and remasking previously unmasked tokens.

Moreover, in Ou et al. (2024), it is proved that the loss that D^θ is trained on (see equation 12 and equation 13) is equivalent to:

$$\begin{aligned} L_{\text{mask}}(\theta) &= -E_{x\sim p_{data}}\left[E_{\sigma\sim\text{Unif}(\mathbb{S}_L)}\left[\log\left((X_1^{\theta,\text{mask}}=x|\sigma)\right)\right]\right] \\ &= E_{\sigma\sim\text{Unif}(\mathbb{S}_L)}\left[D_{KL}(p_{data}||(X_1^{\theta,\text{mask}}=\cdot|\sigma))\right]+H(p_{data}), \end{aligned}$$

where H is the Shannon Entropy of p_{data}. This is minimized with value $H(p_{data})$ if and only if $(X_1^{\theta,\text{mask}}=\cdot|\sigma)=p_{data}, \forall\sigma\in\mathbb{S}_L$; that is, if every choice of unmasking order exactly recovers the data distribution.

It becomes clear that if the training objective used for a Masked Diffusion Model was made uniformly 0, every choice of unmasking order would exactly recover the data distribution (the KL divergence is 0 if and only if the distributions are equal - see e.g. Budhiraja & Dupuis (2019) Lemma 2.1). In practice, however, D^θ is far from perfect (and even if it were, it is trained using samples form p_{data}, so would just recover those samples). As such, not all such orders will be created equal - that is there will be denoising orders $\sigma,\hat{\sigma}\in S_L$ such that

$$D_{KL}(p_{data}||(X_1^{\theta,\text{mask}}=\cdot|\sigma)) >> D_{KL}(p_{data}||(X_1^{\theta,\text{mask}}=\cdot|\hat{\sigma})).$$

This was observed empirically in Ou et al. (2024) Appendix G.4, Shih et al. (2022), and Li et al. (2021) Section 6.

D.4 Comparison with DDPD

As it the most similar work to ours in the existing literature, here we provide a thorough comparison with DDPD Liu et al. (2024). Given that our objective is to plan a denoising order assuming access to a Masked Diffusion Model for our denoiser (as with DDPD-MaskD) and not to train a uniform diffusion-based denoiser from scratch (as with DDPD-DFM-Uni), we focus on their framework in the former setting.

Even with DDPD-MaskD, the framework uses a "uniform discrete diffusion" equation 9 as the starting-point for their token-wise forward noising process, as opposed to the "masked diffusion" forward noising process equation 10 used in our work. They modify the state space S^L to $\tilde{S}^L$, where $\tilde{S}=S\times\{N,D\}$. Dor $(y,z)\in\tilde{S}^L$, (y_i,z_i) denotes the pair describing the state $y_i\in S$ in of i'th token and $z_i\in\{N,D\}$ denotes whether that token is noise (N) or data (D). They then modify the forward noising process to:

$$p_t((j,\zeta)|i)=\alpha(t)\delta_{(i,D)}(j,\zeta)+\frac{1-\alpha(t)}{S}\delta_N(\zeta),\quad i,j\in S,\quad \zeta\in\{N,D\},$$

see Equation (17) therein.

Thus, their reference distribution $\pi\in P(\tilde{S}^L)$ is given by $\pi=\text{Unif}(S^L)\otimes\delta_{N^L}$, where $N^L\in\{N,D\}^L$ consists of all N's, and the corresponding backwards processes' S^L marginal is initialized at the Unif(S^L) as opposed to δ_{M^L} as in our setting.

They approximate a resulting true backward process on S^L's rate matrix $\overleftarrow{Q}_t$ (given by Proposition 3.1 therein) with $Q_t^{\theta,\text{DDPD}}$ given by:

$$Q_t^{\theta,\text{DDPD}}(y,x)=-\frac{\dot{\alpha}(1-t)}{1-\alpha(1-t)}\sum_{i=1}^{L}G_{i,N}^{\theta,\text{DDPD}}(x)E_{Z\sim G^\theta(x)}[D_{i,y_i}^\theta(x^{Z,-i,M})]\delta_{y^{-i}}(x^{-i})$$

where $D^\theta:\bar{S}^LP(S)^L$ is a denosier for a masked diffusion model trained via the ELBO equation 13 as in equation 3. Here for $x\in S^L$, $z\in\{N,D\}^L$, $x^{z,-i,M}\in\bar{S}^L$ is obtained from x via:

$$x_j^z=\begin{cases}M, & z_j=N\\ x_j, & z_j=D, j\neq i\\ M, & j=i\end{cases}.$$

$G^{\theta,\text{DDPD}} : S^L P(\{N, D\})^L$ is another neural network with $G^{\theta,\text{DDPD}}_{i,N}(x)$ approximating the probability that the i'th coordinate of $x \in S^L$ is noise, and is trained via equation 12 with $E(x^0) = E^{\text{DDPD}}(x^0)$ given by:

$$E^{\text{DDPD}}(x^0) = E^{\text{DDPD}}_P(x^0) + E^{\text{DDPD}}_D(x^0)$$

$$E^{\text{DDPD}}_P(x^0) = -\int_0^1 \frac{\dot{\alpha}(t)}{1-\alpha(t)} E_{(\tilde{X}_t, Z_t)\sim P^{\text{DDPD}}_t(\cdot|\delta_{(x^0,D^L)})}\left[\sum_{i=1}^L \log\left(G^{\theta,\text{DDPD}}_{i,[Z_t]_i}(\tilde{X}_t)\right)\right]dt$$

$$E^{\text{DDPD}}_D(x^0) = -\int_0^1 \frac{\dot{\alpha}(t)}{1-\alpha(t)} E_{(\tilde{X}_t, Z_t)\sim P^{\text{DDPD}}_t(\cdot|\delta_{(x^0,D^L)})}\left[\sum_{i=1}^L \delta_{[Z_t]_i=N} E_{\hat{Z}\sim G^{\theta,\text{DDPD}}(\tilde{X}_t)}\left[\log\left(D^\theta_{i,x^0_i}(\tilde{X}_t^{\hat{Z},-i,M})\right)\right]\right]dt$$

$$P_t((y,z)|\delta_{(x^0,D^L)}) \coloneqq \alpha(t)\delta_{(x^0,D^L)}(y,z) + \frac{(1-\alpha(t))}{S^L}\delta_{N^L}(z), \qquad y \in S^L, z \in \{N, D\}^L$$

Note that in the above ELBO, E^{DDPD}_D is slightly modified from what which they present in Theorem 4.1. As written, they would take the expected value with respect to $G^{\theta,\text{DDPD}}$ inside the second $\log$, which requires 2^{L-1} function evaluations of D^θ. When the denoiser D^θ is given by that of a masked diffusion, one should instead use the above, which can be readily arrived at the same proof with an extra application of Jensen's inequality.

Comparing this with our Proposition equation D.1, the comparison between DDPD and P2 becomes evident: $E^{\text{DDPD}}_P(x^0)$ is playing the role of $E_{UP}(x^0) + E_{MP}(x^0)$ (that is, it yields the training objective for the Planner) and $E^{\text{DDPD}}_D(x^0)$ is playing the role of $E_D(x^0)$ (that is, it yields the training objective for the denoiser). However, we note the following key distinguishing factors:

1. In P2, E_D is the same as the ELBO originally used to train the denoiser D^θ: that is, D^θ has already be trained to maximize $E_{x_0\sim p_{\text{data}}}[E_D(x^0)]$. Meanwhile, E^{DDPD}_D depends on the output of $G^{\theta,\text{DDPD}}$, increasing the importance of the role of planner in the quality of the output of the generation. For this reason, DDPD must train an external Planner whose model size is comparable to that of the denoiser - they are essentially asking the planner to play a role akin to the denoiser in a uniform diffusion model. Meanwhile, due to the "flipped" importance of the roles of the planner and denoiser in P2, we show that we can use lightweight BERT models or even the denoiser itself as an effective Planner. See Table 5, where we confirm DDPD's inability to make use of such lightweight models.
2. In P2, we separate the Planner's training objective into two components. This is natural because our planner may use information both from the partially masked data X_t and the output of the denoiser Y. Meanwhile, in DDPD, the Planner only has access to $\tilde{X}_t$-unmasked data perturbed by random flips of its tokens. Because DDPD's generation process is grounded in a uniform diffusion process, there is no ability to separate the Planner into unmasked and masked components as we do in Section equation 3.2. In particular, their framework does not allow for a general enough planner to introduce our stochasticity strength parameter η and design an algorithm analogous to the P2 Sampler D.7.

The practical differences between DDPD and P2 are further elucidated by comparing their Gillespie sampling strategy (Algorithm 1 therein) with ours (see Alg. D.5). For convenience, we reproduce it here.

Letting $\hat{G}^{\theta,\text{DDPD}} : S^L P(\{1, ..., L\})$ be given by $\hat{G}^{\theta,\text{DDPD}}_j(x) = \frac{G^{\theta,\text{DDPD}}_{j,N}(x)}{\sum_{j=1}^L G^{\theta,\text{DDPD}}_{j,N}(x)}$, DDPD's Gillespie sampling algorithm is given by Alg. D.4.

[!h] DDPD Sampler

init $i \leftarrow 0, x_0 \sim \text{Unif}(S^L)$, planner $G^{\theta,\text{DDPD}}$, denoiser D^θ, maximum steps T
for $t = 1 : T$ **do**
  Plan Sample dimension $d' \sim \hat{G}^{\theta,\text{DDPD}}_{\cdot}(x_t)$
  Denoise Sample $z \sim G^{\theta,\text{DDPD}}$
  Sample $y_{d'} \sim D^\theta_{d',\cdot}(x_t^{z,-i,M})$
  Update: $[x_{t+1}]_{d'} \leftarrow y_{d'}$
end for

8: **return** x_T

As is clear from Alg. D.4, in DDPD, the input to the Planner only depends on some unmasked, randomly flipped sequence of tokens, and does not depend on the output of the denoiser, and the input to the denoiser is entirely dependent on the output of the planner. Meanwhile, in P2, the Planner may use the both the information about the partially unmasked sequence (whose unmasked tokens all result from samples from the denoiser) and the output of the denoiser, and the input to the denoiser only depends on the output of the planner insofar as it may choose to remask a single token.

D.5 Deriving the P2 Gillespie Scheme Alg. D.5

Let $\{\tau_k\}_{k\in N}$ be the jump times for the CTMC X^θ with rate matrix Q^θ as described in Equation equation 14 (see Appendix C.2). To derive a Gillespie sampling scheme, we need to find the transition probabilities for the effective jump chain as described in equation 7. We first need to obtain the diagonal entries for the jump matrix Q^θ. We have for $x \in \bar{S}^L$:

$$
\begin{aligned}
-\sum_{y\neq x} Q_t^\theta(y,x) &= \frac{\dot\alpha(1-t)}{1-\alpha(1-t)} \sum_{i=1}^{L} \sum_{y_i=1, y_i\neq x_i}^{N} F_{i,N}^\theta(y,x)\hat{D}_{i,y_i}^\theta(x) \\
&= \frac{\dot\alpha(1-t)}{1-\alpha(1-t)} \sum_{i=1}^{L} \Bigg[\delta_M(x_i) \sum_{y_i=1, y_i\neq x_i}^{N} E_{Z\sim D^\theta(x)}[G_i^\theta(Z^{-i,y_i})] D_{i,y_i}^\theta(x) \\
&+ \frac{(1-\delta_M(x_i))}{1-D_{i,x_i}^\theta(x^{-i,M})} \sum_{y_i=1, y_i\neq x_i}^{N} E_{Z\sim D^\theta(x)}[G_i^\theta(Z^{-i,x_i},x)] D_{i,y_i}^\theta(x^{-i,M}) \Bigg] \\
&= \frac{\dot\alpha(1-t)}{1-\alpha(1-t)} \sum_{i=1}^{L} \delta_M(x_i) E_{Z\sim D^\theta(x)}[G_i^\theta(Z,x)] + (1-\delta_M(x_i)) E_{Z\sim D^\theta(x)}[G_i^\theta(Z^{-i,x},x)] \\
&= Q_t^\theta(x,x).
\end{aligned}
$$

Then for $x \neq y \in \bar{S}^L$, $k \in N$, and $t \in [0,1]$:

$$
(X_{\tau_{k+1}}^\theta = y | X_{\tau_k}^\theta = x, \tau_k = t) = \frac{\sum_{i=1}^{L} F_i^\theta(y,x)\hat{D}_{i,y_i}^\theta(x)\delta_{y^{-i}}(x^{-i})}{\sum_{i=1}^{L} \delta_M(x_i) E_{Z\sim D^\theta(x)}[G_i^\theta(Z,x_i)] + (1-\delta_M(x_i)) E_{Z\sim D^\theta(x)}[G_i^\theta(Z^{-i,x},x)]}.
$$

We note that this is zero when the Hamming distance $d_{HAM}(x,y) \neq 1$ and independent of t and k.

Then, for $j \in \{1, ..., N\}$ and x, y, k, t as before:

$$
\begin{aligned}
&([X_{\tau_{k+1}}^\theta]_j \neq [X_{\tau_{k+1}}^\theta]_j | X_{\tau_k}^\theta = x, \tau_k = t) \\
&= \sum_{y\in\bar{S}^L : y_j \neq x_j} ([X_{\tau_{k+1}}^{\theta,\text{mask}}]_j = y | X_{\tau_k}^{\theta,\text{mask}} = x) \\
&= \sum_{y\in\bar{S}^L : y_j \neq x_j} \frac{\sum_{i=1}^{L} F_i^\theta(y,x)\hat{D}_{i,y_i}^\theta(x)\delta_{y^{-i}}(x^{-i})}{\sum_{i=1}^{L} \delta_M(x_i) E_{Z\sim D^\theta(x)}[G_i^\theta(Z,x)] + (1-\delta_M(x_i)) E_{Z\sim D^\theta(x)}[G_i^\theta(Z^{-i,x},x)]} \\
&= \sum_{y_j=1, y_j\neq x_j}^{N} \frac{F_j^\theta(y,x)\hat{D}_{j,y_j}^\theta(x)}{\sum_{i=1}^{L} \delta_M(x_i) E_{Z\sim D^\theta(x)}[G_i^\theta(Z,x)] + (1-\delta_M(x_i)) E_{Z\sim D^\theta(x)}[G_i^\theta(Z^{-i,x},x)]} \\
&= \frac{\delta_M(x_j) E_{Z\sim D^\theta(x)}[G_j^\theta(Z,x)] + (1-\delta_M(x_j)) E_{Z\sim D^\theta(x)}[G_j^\theta(Z^{-j,x_j},x)]}{\sum_{i=1}^{L} \delta_M(x_i) E_{Z\sim D^\theta(x)}[G_i^\theta(Z,x)] + (1-\delta_M(x_i)) E_{Z\sim D^\theta(x)}[G_i^\theta(Z^{-i,x_i},x)]} \\
&=: P(j,x)
\end{aligned}
$$

and for $y'_j \in S$ with $y'_j \neq x_j$:

$$
\begin{aligned}
&([X^\theta_{\tau_{k+1}}]_j = y'_j | X^\theta_{\tau_k} = x, \tau_k = t, [X^\theta_{\tau_{k+1}}]_j \neq [X^\theta_{\tau_{k+1}}]_j) \\
&= \frac{([X^\theta_{\tau_{k+1}}]_j = y'_j, [X^\theta_{\tau_{k+1}}]_j \neq [X^\theta_{\tau_{k+1}}]_j | X^\theta_{\tau_k} = x, \tau_k = t)}{([X^\theta_{\tau_{k+1}}]_j \neq [X^\theta_{\tau_{k+1}}]_j) | X^\theta_{\tau_k} = x, \tau_k = t)} \\
&= \sum_{y \in \bar{S}^L : y_j = y'_j \neq x_j} \frac{([X^\theta_{\tau_{k+1}}]_j = y'_j | X^\theta_{\tau_k} = x)}{([X^\theta_{\tau_{k+1}}]_j \neq [X^\theta_{\tau_{k+1}}]_j) | X^\theta_{\tau_k} = x, \tau_k = t)} \\
&= \frac{F^\theta_j(x^{-j,y'_j}, x) \hat{D}^\theta_{j,y'_j}(x)}{\delta_M(x_j) E_{Z \sim D^\theta(x)}[G^\theta_j(Z, x)] + (1 - \delta_M(x_j)) E_{Z \sim D^\theta(x)}[G^\theta_j(Z^{-j,x_j}, x)]} \\
&= \frac{\delta_M(x_j) E_{Z \sim D^\theta(x)}[G^\theta_j(Z^{-j,y'_j}, x)] D^\theta_{j,y'_j}(x) + (1 - \delta_M(x_j)) E_{Z \sim D^\theta(x)}[G^\theta_j(Z^{-j,x_j}, x)] \frac{D^\theta_{j,y'_j}(x^{-i,M})}{1 - D^\theta_{i,x_i}(x^{-i,M})}}{\delta_M(x_j) E_{Z \sim D^\theta(x)}[G^\theta_j(Z, x)] + (1 - \delta_M(x_j)) E_{Z \sim D^\theta(x)}[G^\theta_j(Z^{-j,x_j}, x)]} \\
&= \delta_M(x_j) \frac{E_{Z \sim D^\theta(x)}[G^\theta_j(Z^{-j,y'_j}, x)]}{E_{Z \sim D^\theta(x)}[G^\theta_j(Z, x)]} D^\theta_{j,y'_j}(x) + (1 - \delta_M(x_j)) \frac{D^\theta_{j,y'_j}(x^{-i,M})}{1 - D^\theta_{i,x_i}(x^{-i,M})} \\
&=: \tilde{P}(j, x, y'_j).
\end{aligned}
$$

Thus, an exact Gillespie sampling scheme would be given by Gillespie (1977; 1976):

When the chain is in state $x \in \bar{S}^L$, sample a dimension $d' \sim \hat{P}(\cdot, x)$ to change, then sample a value $y \sim \tilde{P}(d', x, \cdot)$ to change it to.

In practice it is impractical to approximate these expected values with respect to $Z \sim D^\theta(x)$, as this would require many function evaluations of the denoiser. However, assuming that the token space is large, conditioning on the value of one coordinate should have little impact on the expected output of the Planner over the entire sequence (see e.g. the discussion under Proposition 3.5. and Appendix E.4 in Liu et al. (2024)). Given that Alg. D.5 is provided for the purpose of exposition and in practice we make use of Alg. D.7 in sampling, we use this intuition to formally approximate:

$$\tilde{P}(j, x, y'_j) \approx \delta_M(x_j) D^\theta_{j,y'_j}(x) + (1 - \delta_M(x_j)) \frac{D^\theta_{j,y'_j}(x^{-i,M})}{1 - D^\theta_{i,x_i}(x^{-i,M})}$$

and

$$P(j, x) \approx \frac{E_{Z \sim D^\theta(x)}[G^\theta_j(Z, x)]}{\sum_{i=1}^L E_{Z \sim D^\theta(x)}[G^\theta_i(Z, x)]} \approx E_{Z \sim D^\theta(x)}[\hat{G}_j(Z, x)],$$

where : $\hat{G}^\theta : S^L \times \bar{S}^L P(\{1, ..., L\})$ is given by:

$$\hat{G}_j(y, x) := \frac{G^\theta_j(y, x)}{\sum_{j=1}^L G^\theta_j(y, x)}.$$

We then arrive at: [!h] Our Gillespie Sampler

Initialize: $t \leftarrow 0, x_0 \leftarrow (M, \ldots, M)$, planner G^θ, denoiser D^θ, maximum steps T
for $t = 1 : T$ **do**
  Plan Sample $y \sim D^\theta(x_t)$
  Sample dimension $d' \sim \hat{G}^\theta_\cdot(y, x_t)$
  Denoise
  if $[x_t]_{d'} \neq M$ **then**
    $[x_t]_d \leftarrow M$
    Resample $y_{d'} \sim D^\theta_{d',\cdot}(x_t)$
    $[x_{t+1}]_{d'} \leftarrow y_{d'}$
  else
    $[x_{t+1}]_{d'} \leftarrow y_{d'}$
  end if
end for
return x_T

D.6 Proof of the ELBO Proposition D.1

As per the discussion in Section C.5, it suffices to find a lower bound on $-D_{KL}(\mathbb{P}^{x^0}||\mathbb{P}^\theta)$, where $\mathbb{P}^{x^0}$ is the Law of the continuous time Markov chain $\overleftarrow{X}^{x^0}$ with rate matrix $\overleftarrow{Q}^{x^0}$ given by equation 11, $\mathbb{P}^\theta$ is the Law of the continuous time Markov chain X^θ with rate matrix Q^θ given by equation 14, and $\overleftarrow{X}_0^{x^0} = X_0^\theta = M^L$. Via an application of Girsanov's Theorem for CTMCs (see e.g. Theorem III.5.34 in Jacod & Shiryaev (2013) for a general result or Ren et al. (2024) Theorem 3.3 for the specific CTMC setting):

$$
\begin{aligned}
&-D_{KL}(\mathbb{P}^{x^0}||\mathbb{P}^\theta) \\
&= -\int_0^1 E_{X_t\sim P_{1-t}(\cdot;\delta_{x^0})}\left[\sum_{y\neq X_t} Q_t^\theta(y,X_t) - \overleftarrow{Q}^{x^0}(y,X_t) + \overleftarrow{Q}^{x^0}(y,X_t)\log\left(\frac{\overleftarrow{Q}^{x^0}(y,X_t)}{Q_t^\theta(y,X_t)}\right)\right]dt \\
&= -\int_0^1 E_{X_t\sim P_{1-t}(\cdot;\delta_{x^0})}\left[-Q_t^\theta(X_t,X_t) + \overleftarrow{Q}^{x^0}(X_t,X_t) + \sum_{y\neq X_t}\overleftarrow{Q}^{x^0}(y,X_t)\log\left(\frac{\overleftarrow{Q}^{x^0}(y,X_t)}{Q_t^\theta(y,X_t)}\right)\right]dt \\
&= -\int_0^1 \frac{\dot\alpha(t)}{1-\alpha(t)} E_{X_t\sim P_t(\cdot;\delta_{x^0})}\left[\sum_{i=1}^L \delta_M([X_t]_i)(1 - E_{Y\sim D^\theta(X_t)}[G_i^\theta(Y,X_t)])\right. \\
&\left. - (1-\delta_M([X_t]_i))E_{Y\sim D^\theta(X_t)}[G_i^\theta(Y^{-i,[X_t]_i},X_t)] + \delta_M([X_t]_i)\log(F_i^\theta(x^0,X_t)\hat{D}^\theta_{i,x_i^0}(X_t))\right]dt,
\end{aligned}
$$

where in the third equality we have inserted the definitions of $\overleftarrow{Q}^{x^0}$ and Q^θ and reversed the role of the time parameter $t \mapsto 1-t$, and P_t is as in equation 1.

We consider this as 4 parts:

$$
\begin{aligned}
E_1(x^0) &\coloneqq -\int_0^1 \frac{\dot\alpha(t)}{1-\alpha(t)} E_{X_t\sim P_t(\cdot;\delta_{x^0})}\left[\sum_{i=1}^L \delta_M([X_t]_i)(1 - E_{Y\sim D^\theta(X_t)}[G_i^\theta(Y,X_t)]\right]dt \\
E_2(x^0) &\coloneqq -\int_0^1 \frac{\dot\alpha(t)}{1-\alpha(t)} E_{X_t\sim P_t(\cdot;\delta_{x^0})}\left[\sum_{i=1}^L (1-\delta_M([X_t]_i))\left(-E_{Y\sim D^\theta(X_t)}[G_i^\theta(Y^{-i,[X_t]_i},X_t)]\right)\right]dt \\
E_3(x^0) &\coloneqq -\int_0^1 \frac{\dot\alpha(t)}{1-\alpha(t)} E_{X_t\sim P_t(\cdot;\delta_{x^0})}\left[\sum_{i=1}^L \delta_M([X_t]_i)\log(F_i^\theta(x^0,X_t))\right]dt \\
E_4(x^0) &\coloneqq -\int_0^1 \frac{\dot\alpha(t)}{1-\alpha(t)} E_{X_t\sim P_t(\cdot;\delta_{x^0})}\left[\sum_{i=1}^L \delta_M([X_t]_i)\log(\hat{D}^\theta_{i,x_i^0}(X_t))\right]dt
\end{aligned}
$$

Recalling that $\dot\alpha(t) \leq 0$ for all $t \in [0,1]$ and $G_i^\theta(y,x) \in [0,1]$ for all $i \in \{1,\dots,L\}$, $y \in S^L$ and $x \in \bar{S}^L$, we see $E_1(x^0)$ is positive for all $x^0 \in S^L$, and artificially attempting to ensure that the rates of the original CTMC and our modified one do not differ too much out of masked positions (see the discussion of the "Rate Forcing Term" in Appendix C.2 of Campbell et al. (2024)). Hence we simply bound it below by zero:

$$E_1(x^0) \geq 0,$$

because we are only interested in P_1^θ being close to p_{data}, not the entire trajectory of the chains X^θ and $\overleftarrow{X}$ being close.

For the $E_2(x^0)$ we note that, by definition of P_t, when $[X_t]_i \neq M$, it is equal to its initial value x_i^0. Along with the bound $-z \geq \log(1-z), \forall z \in [0,1)$, this yields:

$$E_2(x^0) \geq -\int_0^1 \frac{\dot\alpha(t)}{1-\alpha(t)} E_{X_t\sim P_t(\cdot;\delta_{x^0})}\left[\sum_{i=1}^L (1-\delta_M([X_t]_i))\log(E_{Y\sim D^\theta(X_t)}[1 - G_i^\theta(Y^{-i,x_i^0},X_t)])\right]dt.$$

Applying Jensen's inequality to move the expected value with respect to $D^\theta(X_t)$ outside of the log yields:

$$E_2(x^0) \geq E_{UP}(x^0), \forall x^0 \in S^L.$$

For $E_3(x^0)$ we note that, by definition, when $[X_t]_i = M$, $F_i^\theta(x^0, X_t) = E_{Y \sim D^\theta(X_t)}[G_j^\theta(Y^{-i,x_i^0}, X_t)]$. An application of Jensen's inequality to $D^\theta(X_t)$ outside of the log yields:

$$E_3(x^0) \geq E_{MP}(x^0), \forall x^0 \in S^L.$$

Finally, for $E_4(x^0)$, we note that, by definition, when $[X_t]_i = M$, $\hat{D}^\theta_{i,x_i^0}(X_t) = D^\theta_{i,x_i^0}(X_t)$, so

$$E_4(x^0) = E_{MP}(x^0), \forall x^0 \in S^L.$$

This results in the desired bound.

D.7 THE P2 SAMPLER PSEUDOCODE

[h] P2 Sampler (Pytorch Implementation in Appendix Sec. E).

Initialize: $t \leftarrow 0$, $x_0 \leftarrow (M, \ldots, M)$, planner G^θ, denoiser D^θ, scheduler K
for $t = 1 : L$ **do**
  Plan:
  Sample $y \sim D^\theta(x_t)$
  UpdatePos $\leftarrow$ TopPos$_{\kappa(t)}\big(\tilde{G}^\theta_\cdot(y, x_t)\big)$
  Denoise:
  for $j \in$ UpdatePos **do**
    if $[x_t]_j = M$ **then**
      $[x_t]_j \leftarrow y_j$
    end if
  end for
  for $j \notin$ UpdatePos **do**
    if $[x_t]_j \neq M$ **then**
      $[x_t]_j \leftarrow M$
    end if
  end for
end for
return x_L

E IMPLEMENTATION DETAILS

In Listing 1, we provide a self-contained PyTorch implementation of our *Path-Planning Sampling* procedure. The code consists of three core components, each addressing a distinct step in the sampling process:

1) `topk_lowest_masking`: Given a matrix of scalar scores, this function returns a boolean mask that flags the "lowest-scoring" positions per row. The user can specify how many positions should be re-masked by providing a `cutoff_len` tensor. Internally, the function sorts the score matrix and determines the threshold score for each row before comparing every score to this cutoff.

2) `stochastic_sample_from_categorical`: This function draws samples from a categorical distribution using Gumbel noise. It first applies Gumbel noise to the input logits (if a non-zero temperature is specified), then computes the log-softmax to obtain token probabilities. The sampled tokens and their corresponding log probabilities are returned.

3) `path_planning_sampling`: Positions initially set to the `mask_token_id` are iteratively predicted and updated. At each iteration, we:

1. Compute model logits and identify positions that remain masked.

2. Sample from the model outputs via `stochastic_sample_from_categorical`.
3. Integrate a `planner` (if provided) to re-score predictions for currently unmasked positions, giving users the flexibility to incorporate any additional guidance or constraints.
4. Construct a `score` and re-mask positions with the lowest scores. Fixed positions are ignored by assigning them infinite scores so that they cannot be re-masked.
5. Scale the scores of unmasked positions by the factor η, which adjusts how aggressively new tokens are updated.

The function continues for `num_steps`, revealing high-confidence predictions and re-masking uncertain positions. Finally, any remaining masks are replaced with the last sampled tokens. The key parameters are:

- `xt`: The initial token matrix of shape $[B, L]$, containing masked tokens.
- `model`: A callable mapping tokens to logits.
- `tokenizer`: Provides the special `mask_token_id`.
- `num_steps`: Number of refinement iterations.
- `tau`: Temperature for controlling sampling noise.
- `kappa_fn`: A schedule function in $[0, 1]$ that dictates how many positions remain masked vs. unmasked over time.
- `eta`: A multiplier for scores in unmasked positions.
- `planner`: An optional model for additional re-scoring.
- `score_type`: Either `'confidence'` (uses log probabilities) or `'random'` (random re-masking).

Listing 1: Path-Planning Sampling procedure in PyTorch

```
import torch

def topk_lowest_masking(scores, cutoff_len):
   sorted_scores, _ = scores.sort(dim=-1)
   threshold = sorted_scores.gather(dim=-1, index=cutoff_len)
   return scores < threshold

def stochastic_sample_from_categorical(logits, temperature=1.0,
    noise_scale=1.0):
   logits = logits.double()
   if temperature != 0.0:
      gumbel = -torch.log(-torch.log(torch.rand_like(logits) + 1e-8) + 1e
          -8)
      logits = logits / temperature + noise_scale * gumbel
   scores, tokens = logits.log_softmax(dim=-1).max(dim=-1)
   return tokens, scores

@torch.inference_mode()
@torch.cuda.amp.autocast()
def path_planning_sampling(
   xt,
   model,
   tokenizer,
   num_steps,
   tau=1.0,
   kappa_fn=lambda t: t,
   eta=1.0,
   planner=None,
   score_type='confidence'
):
   fix_mask = (xt != tokenizer.mask_token_id)
   dt = 1.0 / num_steps

   for step in range(1, num_steps + 1):
```

```
        t = step * dt
        kappa_t = kappa_fn(t)
        logits = model(xt).double()

        last_mask = (xt == tokenizer.mask_token_id)
        unmask_candidates = ~last_mask & ~fix_mask

        x0, logp = stochastic_sample_from_categorical(logits, temperature=
            tau)

        if planner is not None:
           planner_logits = planner(x0).double()
           planner_logp = planner_logits.log_softmax(dim=-1).gather(-1, x0.
               unsqueeze(-1)).squeeze(-1)
           logits[unmask_candidates] = planner_logits[unmask_candidates]
           logp[unmask_candidates] = planner_logp[unmask_candidates]

        if score_type == 'confidence':
           score = logp
        elif score_type == 'random':
           score = torch.rand_like(logp).log()
        else:
           raise ValueError("Invalid score_type.")

        score = score.masked_fill(fix_mask, float('inf'))
        score[unmask_candidates] *= eta

        num_to_mask = ((~fix_mask).sum(dim=1, keepdim=True).float() * (1 -
            kappa_t)).long()
        mask = topk_lowest_masking(score, num_to_mask)
        xt[mask] = tokenizer.mask_token_id

        mask_to_x0 = last_mask & ~mask
        xt[mask_to_x0] = x0[mask_to_x0]

    remaining_mask = (xt == tokenizer.mask_token_id)
    xt[remaining_mask] = x0[remaining_mask]

    return xt
```

F Experimental Details

F.1 Example of Language generation Task

We provide Table 3 consisting of examples for the five language generation tasks.

F.2 Protein Sequence Generation

Setup We compare our method with state-of-the-art protein sequence generation models, including three discrete diffusion models—DPLM (Wang et al., 2024a), EvoDiff (Alamdari et al., 2024), and ESM3 (Hayes et al., 2025)—and an autoregressive model, ProGen2 (Nijkamp et al., 2022), across three model sizes: small, medium, and large. Additionally, we benchmark masked language models, ESM2 (Lin et al., 2023), at three scales: 150M, 650M, and 3B parameters.

For our path-planning algorithm (P2), we vary the stochasticity strength from 1.0 to 2.0 in increments of 0.1 and report optimal results. Baselines are evaluated with default sampling strategies. Since ESM2 lacks a masked diffusion loss, it uses ancestral sampling. Each model generates 100 sequences for sequence lengths in $[200, 300, \ldots, 800]$. DPLM employs a sequence length matching the number of sampling steps and a temperature of 0.9, with rejection-resampling disabled for fairness. ESM3 is sampled with a temperature of 1, a cosine schedule, top-$p = 1$, and 500 steps. Special tokens are removed to ensure valid amino acid sequences.

Table 3: Examples from language understanding benchmarks.

Metric	Question	Answer
LAMBADA	"Again, he left that up to you. However, he was adamant in his desire that it remain a private ceremony. He asked me to make sure, for instance, that no information be given to the newspaper regarding his death, not even an obituary. I got the sense that he didn't want anyone, aside from the three of us, to know that he'd even __."	died
GSM8K	Weng earns $12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?	10
TriQA	The Dodecanese Campaign of WWII that was an attempt by the Allied forces to capture islands in the Aegean Sea was the inspiration for which acclaimed 1961 commando film?	The Guns of Navarone
ROCStories	Morgan and her family lived in Florida. They heard a hurricane was coming. **(Story infills here...)** They arrived and learned from the news that it was a terrible storm. They felt lucky they had evacuated when they did.	They decided to evacuate to a relative's house.

Code	`from typing import List`
	
	`def has_close_elements(numbers: List[float], threshold: float) -> bool:`
	`  """`
	`  Check if in given list of numbers, are any two numbers closer`
	`  to each other than given threshold.`
	
	`  >>> has_close_elements([1.0, 2.0, 3.0], 0.5)`
	`  False`
	`  >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)`
	`  True`
	`  """`
	`  # Infill Code`
	`  if distance < threshold:`
	`  return True`
	`  return False`
		`for idx, elem in enumerate(numbers):`
		`  for idx2, elem2 in enumerate(numbers):`
		
		`  if idx != idx2:`
		`  distance = abs(elem - elem2)`

Evaluation. Protein sequence generation quality is evaluated via protein folding models, using ESMFold (Lin et al., 2023) as a proxy for structural stability. We extract three folding metrics:

- **pLDDT** (predicted Local Distance Difference Test): Measures local structural accuracy.
- **pTM** (predicted Template Modeling): Assesses global structural plausibility.
- **pAE** (predicted Alignment Error): Evaluates overall compactness.

A sequence can achieve high pLDDT while exhibiting poor global compactness (high pAE). To ensure robust evaluation, we define *foldability* as the proportion of sequences satisfying pLDDT > 80, pTM > 0.7, and pAE < 10. This metric effectively identifies low-quality sequences, such as repetitive patterns (e.g., "ABABABAB"), which tend to have high pAE.

Beyond folding scores, we compute:

- **Token entropy**, excluding tokens not present in generated sequences.
- **Sequence diversity**, defined as $1-$ pairwise sequence identity within a batch. Since all sequences in a batch share equal length, no sequence alignment is needed.

These metrics detect mode collapse, where models generate highly repetitive sequences.

G Protein Sequence Generation

G.0.1 Training Details of the 150M MDM.

We train a 150M mask diffusion model on protein sequences for the ablation of self-planning. The 150M MDM is trained using the open-sourced DPLM code[4]. We use the same transformer architecture as DPLM-150M as well as ESM2-150M. We train our MDM from scratch for 500k steps with a total of 320K tokens in each iteration, which is achieved by multi-GPU and multi-node

[4]https://github.com/bytedance/dplm

training with gradient accumulation. The training data is Uniref50, consisting of around 40M protein sequences with 50% sequence-identity cutoff, namely, the sequences in uniref50 are at least higher than 50% dissimilar. Uniref50 is widely used for training protein language models.

G.1 Computing the ELBO

The Evidence Lower Bound (ELBO) serves as the training objective of mask diffusion models and can be used to assess how well the model fits the data. The ELBO experiments are conducted on protein sequence generation tasks. We compute the negative ELBO for five planners, namely ESM-8M, ESM-35M, ESM-150M, ESM-650M, and ESM-3B, alongside the self-planning ELBO, using a weighted cross-entropy loss function to quantify reconstruction accuracy.

Dataset Preparation. We utilize sequences from the UniRef50 dataset, filtering to include only test sequences with lengths shorter than 300 residues to align with the experiments in Figure 4 and mitigate memory constraints. The dataset is loaded into a PyTorch DataLoader using a sequence length of 1022 tokens and a maximum token budget of 60,000. For consistent evaluation, we run the ELBO calculation over 20 independent simulations and report the average across these runs.

Masking Strategy. For each sequence, we randomly generate a mask ratio uniformly sampled from the range $[1/500, 1 - 1/500]$. Positions are masked based on this ratio, but masking is constrained to avoid altering non-maskable tokens (e.g., special symbols). The masked tokens are replaced with a designated mask token provided by the denoiser model.

Loss Calculation. To compute the ELBO, the denoiser and planner models predict the original tokens for both masked and unmasked positions. The cross-entropy loss is calculated separately for these categories. Both masked and unmasked loss values are weighted inversely by the mask ratio to ensure probabilistic consistency in the evaluation. Each model is evaluated across 20 independent simulations, and the average ELBO is reported to capture the robustness of the planners under stochastic settings.

H Additional Results

H.1 Plug-and-Play Path Planning Sampler

H.1.1 Self-Planning with Denoiser-Predicted Probabilities

We propose a self-planning mechanism by leveraging denoiser-predicted probabilities to guide unmasking and remasking decisions. Within the P2 framework, the unmask planner and mask planner are unified by setting $G_j^U(y, x) = G_j^M(y, x) = D^\theta_{j,y_j}(x)$, that is, the denoiser itself serves as the planner. For mask positions, the denoiser is trained to predict tokens given the surrounding context, and the predicted probabilities serve as confidence estimates for the correctness of token predictions. This methodology aligns with established practices in the literature (Gong et al., 2024; Chang et al., 2022; Zheng et al., 2023; Wang et al., 2024a;b). However, a concern arises for unmasked positions, as these tokens act as context during training and are not directly supervised. This raises the question: *Are the predicted probabilities for unmask positions meaningful?* Our empirical evaluation demonstrates that, despite the absence of supervision for unmask positions, the ELBO (weighted cross-entropy loss, see Prop. D.1) for unmasked tokens surpasses that of BERT, which explicitly trains on both masked and unmasked tokens (see Table 6). Furthermore, ablating the denoiser-predicted probabilities for unmasked positions by replacing them with uniformly sampled values results in significant performance degradation (see Table 7). This evidence confirms that the probabilities for unmask tokens are indeed informative, even without direct training. We hypothesize two key factors behind this phenomenon. 1) During masked token prediction, the model inherently learns robust representations of unmasked tokens for predicting the masked positions. 2) The model's output layer projects embeddings of both masked and unmasked tokens into a shared logits space. Consequently, unmasked tokens can yield meaningful logits.

H.1.2 BERT-PLANNING

In BERT-planning, we introduce a class of special planner BERT (Devlin et al., 2019), a bidirectional language model trained to predict the correct tokens given the corrupted sequences (15% of tokens masked and 1.5% of tokens uniformly flipped to other tokens). Despite such a simple training objective, BERT learns to estimates the naturalness of a token with the predicted probabilities which demonstrates wide application in zero-shot mutation prediction (Hie et al., 2022). Compared to training a dedicated planner that is equal-size to denoiser as in DDPD (Liu et al., 2024), BERT is more versatile, flexible in sizes and often available in common tasks such as text (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2019), protein Lin et al. (2023); Hayes et al. (2025); Wang et al. (2024a;b) and RNA (Penić et al., 2024).

Let $B^\theta : S^L P(S)^L$ be a pretrained BERT model, so that $B^\theta_{j,y_j}(y)$ is assigning the probability that the jth token in the sequence y is clean. In BERT planning we set unmask planner to be the BERT $G^U_j(y, x) = B^\theta_{j,y_j}(y)$ and mask planner to be the denoiser $G^M_j(y, x) = D^\theta_{j,y_j}(x)$.

H.2 P2 GENERALIZES EXISTING SAMPLING METHODS

In Table 4, we show the existing sampling methods fit into our P2 framework with specific parameters. Ancestral sampling disables the remasking by setting the Unmasked Planner ($G^U_j(y, x)$) to always output 1, i.e., the likelihood that an unmask token should be kept is always 1, and the mask planner $G^M_j(y, x)$ functions as a uniform sampler as it randomly selects mask positions. Greedy ancestral sampling improves open this by using the denoiser $D^\theta_{j,y_j}(x)$ as the mask planner $G^M_j(y, x)$. DFM Sampling randomly selects positions, and enables remasking by introducing a tunable stochasticity strength η. RDM functions identically to our self-planning by using the denoiser for both mask and unmask planning but it omits the stochasticity control with the default stochasticity strength $\eta = 1$. DDPD introduces external planners and purely relies on the planner for both mask and unmask position planning with default stochasticity strength $\eta = 1$. See Appendix D.4 for further comparison of P2 with DDPD.

Table 4: Generalization of Existing Sampling Methods within our P2 Framework. **Mask Planner ($G^M_j(y, x)$)** gives the likelihood that a mask token should be unmasked. **Unmask Planner ($G^U_j(y, x)$)** gives the likelihood that an unmask token should be kept. $D^\theta_{j,y_j}(x)$ gives the prediction probability of the denoiser at position j for token y_j. $B^\theta(\cdot)$ is a BERT. $G^\theta(\cdot)$ is an external planner.

Method	Remasking	Planning	Stochasticity Control	Mask Planner ($G^M_j(y,x)$)	Unmask Planner ($G^U_j(y,x)$)
Ancestral (Shi et al., 2024; Sahoo et al., 2024)				$\mathcal{U}(0,1)$	1
Greedy Ancestral (Gong et al., 2024)		✓		$D^\theta_{j,y_j}(x)$	1
DFM Sampling (Campbell et al., 2024)			✓	$\mathcal{U}(0,1)$	$\mathcal{U}(0,1)$
RDM Sampling (Zheng et al., 2023; Wang et al., 2024a;b)	✓	✓		$D^\theta_{j,y_j}(x)$	$D^\theta_{j,y_j}(x)$
DDPD (Liu et al., 2024)	✓	✓		$G^\theta_j(y)$	$G^\theta_j(y)$
Path Planning (Self-Planning, ours)	✓	✓	✓	$D^\theta_{j,y_j}(x)$	$D^\theta_{j,y_j}(x)$
Path Planning (BERT Planner, ours)	✓	✓	✓	$D^\theta_{j,y_j}(x)$	$B^\theta_{j,y_j}(y)$

H.3 THE DESIGN SPACE OF PATH PLANNING

Our Path Planning (P2) framework generalizes existing sampling strategies, including vanilla ancestral sampling, greedy ancestral sampling, RDM sampling, and DFM sampling, by incorporating specific parameterizations. In Figure 3, we instantiate these sampling algorithms and evaluate their performance on protein sequence generation, focusing on foldability (additional metric results are provided in Appendix Figure 7).

Vanilla and greedy ancestral sampling employ a stochasticity strength of 0, effectively disabling remasking, which results in poor performance. DFM sampling introduces tunable stochasticity, leading to improved performance over ancestral sampling; however, it lacks trajectory planning, which limits its effectiveness. RDM sampling, by contrast, enables remasking with a default stochasticity strength of 1 and utilizes the denoiser's confidence for self-planning, yielding better sampling quality.

P2 combines the advantages of these existing algorithms, offering both controllable stochasticity strength and planning guidance. By tuning stochasticity strength, P2 can enhance RDM sampling and optionally leverage an external BERT planner to further steer the sampling trajectory toward generating high-quality sequences.

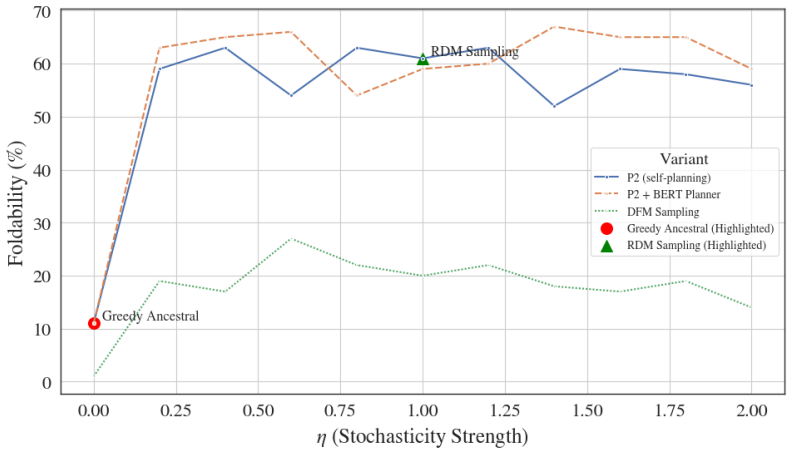

Figure 3: The Design Space of P2 (See Appendix Figure 7 for more). P2 Generalizes existing sampling algorithms with specific stochasticity strength and planner choice.

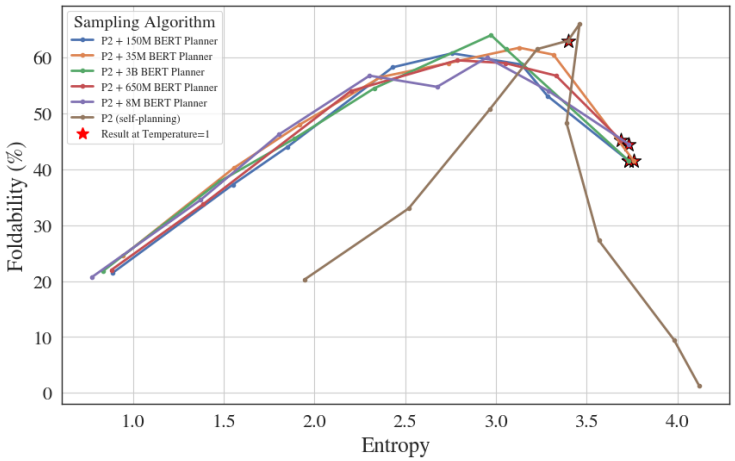

Figure 4: Ablation of the Planner Size: an 8M BERT planner functions similarly to a 3B BERT. Self-planning performs better in a default temperature of 1. We sweep the temperature from 0.1 to 2.0 and plot the scaling between the resultant sequence entropy and the foldability. See Appendix Figure 8 for more.

H.4 Ablation of Path Planning

Table 5: Ablation of Sampling Strategies. Path planning (P2) outperforms existing sampling strategies, including DDPD. The arrows indicate whether higher (↑) or lower (↓) values are better.

Sampling Algorithm	pLDDT (↑)	pTM (↑)	pAE (↓)	Foldability (%) (↑)	Entropy (↑)	Diversity (%) (↑)
Vanilla Ancestral	44.08	0.34	20.61	2.00	**4.03**	**93.63**
RDM Sampling	74.67	0.71	10.33	43.00	3.85	93.12
P2 + 8M BERT Planner	**78.24**	**0.74**	**9.11**	**44.50**	3.80	92.77
DDPD + 8M BERT Planner	46.51	0.24	23.20	0.25	0.31	51.69
Ancestral	52.67	0.46	17.64	7.75	3.98	93.42

In this section, we utilize the protein sequence generation task as an ablation benchmark to analyze the implications of our Path Planning (P2) design choices. We experiment with the ESM2 (Lin et al., 2023) family of protein language models, including versions with 8M, 35M, 150M, 650M, and 3B parameters, for variants incorporating a BERT planner. For the denoiser, we train a 150M MDM

Table 6: Comparison of negative ELBOs for Path Planning Planners and self-planning, averaged on 20 runs. Lower values (↓) indicate better ELBO. The ELBO is computed at default temperature 1, corresponding to the star-annotation results in Figure 4.

Method	Unmasked pos.-ELBO (↓)	Masked pos.-ELBO (↓)
P2 + Planner ESM2-8M	22.5	13.4
P2 + Planner ESM2-35M	22.0	13.4
P2 + Planner ESM2-150M	21.8	13.4
P2 + Planner ESM2-650M	21.7	13.4
P2 + Planner ESM2-3B	21.6	13.4
P2 (self-planning)	15.7	13.4

Table 7: Ablation study of self-planning. We compare self-planning using denoiser-predicted probabilities with a uniformly sampled probability baseline. finetuned MDM refers to MDM finetuned from BERT (DPLM-150M (Wang et al., 2024a)), while tfs-MDM refers to MDM trained from scratch.

Configuration	pLDDT (↑)	pTM (↑)	pAE (↓)	Foldability (↑)	Entropy (↑)	Diversity (↑)
finetuned MDM	82.62	0.72	9.15	63.00	3.40	93.05
finetuned MDM + Uniform	72.61	0.66	11.82	39.00	4.01	93.62
tfs-MDM	74.67	0.71	10.33	43.00	3.85	93.12
tfs-MDM + Uniform	59.88	0.52	15.57	20.00	4.00	93.57

from scratch, using the same architecture as ESM2-150M and DPLM-150M, for 500k steps with approximately 320k tokens per step. Training details are provided in Appendix G.0.1.

Results. Table 5 demonstrates that our P2 approach consistently outperforms existing sampling strategies across all folding metrics, while maintaining strong token entropy and sequence diversity. Notably, results are further enhanced when an external BERT planner is utilized. To provide a comparative perspective, we perform an apple-to-orange evaluation against a planner-based sampling algorithm, DDPD, equipped with the same BERT planner. DDPD is prone to generating low-entropy, repetitive sequences with poor foldability, as it relies exclusively on the planner to dictate both unmasking and remasking. In contrast, P2 separates these responsibilities: remasking is delegated to the BERT planner, while unmasking is guided by the denoiser itself. This decomposition mitigates the planner's bias and leverages the denoiser's planning capabilities effectively.

In Figure 4, we ablate the size of the planner and evaluate foldability under varying temperatures (entropy). Additional metric results are shown in Appendix Figure 8. Our findings reveal that an 8M BERT planner is sufficient to guide a 150M MDM, achieving competitive performance relative to its 3B counterpart across a broad range of entropy values. Furthermore, the BERT planner demonstrates superior scalability compared to the self-planning variant, preserving foldability under extreme high and low temperature conditions.

Self-Planning Analysis. In our self-planning approach, we leverage the predicted probabilities from unmasked positions to guide unmasking decisions. This raises a key question: Are the predicted probabilities from unmasked tokens meaningful? We conducted an ablation study where we replaced predicted probabilities for unmasked tokens with uniformly random values and performed the experiments on two MDM variants: one trained from scratch and another fine-tuned from a BERT-based model (DPLM-150M (Wang et al., 2024a)). The DPLM-150M was fine-tuned from ESM2, which was pretrained to predict both masked and randomly mutated tokens, making it more likely to inherit meaningful logits for unmasked positions. As shown in Table 7, randomizing unmasked token probabilities leads to a substantial decline in performance across both variants. This finding confirms that unmasked token logits are informative, despite the lack of direct supervision. It is also evidenced by the ELBO from Proposition D.1 in 6 where self-planning displays an even better ELBO compared with BERT planners, further validating its effectiveness.

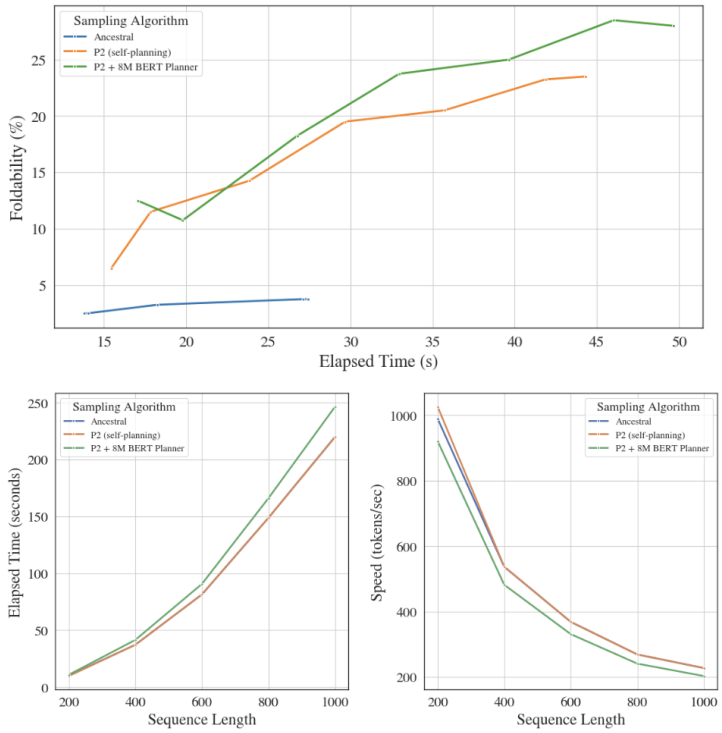

Figure 5: Top: Performance vs. Sampling Time (steps). Bottom: Running Time (left) and Speed (right) vs. Sequence Length.

H.5 Sampling Efficiency

Increasing the number of sampling steps generally enhances generative quality, albeit with increased computational time. To evaluate the scaling efficiency, we benchmark three sampling algorithms—ancestral sampling, P2 (self-planning), and P2 augmented with an 8M BERT planner—on the task of protein sequence generation. We measure the foldability across increasing sampling steps in terms of elapsed time (benchmarked on NVIDIA A100 GPUs). In Figure 5 top, P2 achieves superior foldability compared to ancestral sampling, while the inclusion of the external BERT planner demonstrates exceptional scalability, particularly at higher sampling steps. In Figure 5 bottom, we further analyze inference efficiency by examining elapsed time and speed (tokens per second) as a function of sequence length. P2 with self-planning maintains the same inference cost as ancestral sampling, as it does not rely on an external model. Conversely, P2 with the BERT planner doubles the number of sampling steps due to one additional BERT evaluation. However, since the planner is a lightweight 8M model compared to the 150M MDM, the overhead is negligible. This is evident in the figure, where the performance gap between P2 (self-planning) and P2 with the 8M BERT planner becomes indistinguishable at higher sampling scales.

H.6 Language Generation

It has been widely pointed out that the existing evaluation such as toy datasets and NLL in text generation can be easily gamed to achieve low perplexity (Zheng et al., 2024b). In our evaluation, we follow the language benchmarking from SMDM (Gong et al., 2024) and DiffuLLama (Nie et al., 2024), and investigate the capabilities of MDMs in real-world evaluation language generation tasks that have been largely overlooked in prior works (Austin et al., 2021; Lou et al., 2023; Sahoo et al., 2024; Shi et al., 2024). We additionally provide the experiments of breaking the reverse curse in the Appendix H.7.1.

Benchmarks. We consider TriviaQA (Joshi et al., 2017) to test the reading comprehension of models and the last word completion task Lambada (Paperno et al., 2016)to test how models capture long-range dependencies in text. These two tasks are measured by exact match accuracy, i.e., given a

Table 8: Language generation benchmarks, including reading comprehension (TriQA), last word completion task (LAMBADA), math reasoning (GSM8K), story infilling (ROCStories), and code generation. Baseline results are adopted from (Nie et al., 2024; Gong et al., 2024). For the infilling task, we use ROUGE-1/2/L score; for other tasks, we use the accuracy (%) metric. We employ P2 for the MDMs (1.1B) (Nie et al., 2024) and DiffuLLama (7B) (Nie et al., 2024) and show consistent improvement.

Model	**TriQA (↑)**	**LAMBADA (↑)**	**GSM8K (↑)**	**ROCStories (↑)**	**Code (↑)**
GPT2-S (127M)	4.0	25.9	44.8	(7.8/0.8/7.4)	1.6
DiffuGPT-S (127M)	2.0	45.0	50.2	13.7/1.4/12.6	0.3
SEDD-S (170M)	1.5	12.4	45.3	11.9/0.7/10.9	0.7
GPT2-M (355M)	6.7	37.7	50.7	(8.6/0.9/8.2)	2.6
DiffuGPT-M (355M)	3.8	60.5	52.6	18.7/2.7/17.0	2.9
SEDD-M (424M)	1.8	23.1	53.5	13.1/1.4/12.2	0.5
Plaid1B (1.3B)	1.2	8.6	32.6	12.1/1.1/11.2	0.1
TinyLlama (1.1B)	-	43.22	-	-	-
GPT-2 (1.5B)	-	44.61	-	-	-
Llama-2 (7B)	45.4	68.8	58.6	(11.6/2.1/10.5)	1.7
MDM (1.1B)	-	52.73	58.5	-	-
MDM (1.1B) + P2	**-**	**52.88**	**60.9**	**-**	**-**
DiffuLLama (7B)	18.5	53.72	-	20.31/2.83/18.16	13.2
DiffuLLama (7B) + P2	**18.8**	**54.80**	**-**	**25.44/7.10/23.41**	**17.6**

prompt, we use MDMs to generate responses and calculate matching accuracy against the ground truth. Additionally, we employ complex tasks such as GSM8K (Cobbe et al., 2021), grade school math problem, to assess the *math reasoning* and story-infilling task using ROCStories (Mostafazadeh et al., 2016) and evaluate using ROUGE score (Lin, 2004). To test the code infilling, we also adopted Humaneval (Bavarian et al., 2022) single line infilling task, which is evaluated by pass@1 rate. We employ Language Model Evaluation Harness framework (Biderman et al., 2024) for performance assessment.

Baselines. We adopt the baselines and their results from previous works (Nie et al., 2024; Gong et al., 2024) , including continuous diffusion model Plaid1B (1.3B) (Gulrajani & Hashimoto, 2023), discrete diffusion model SEDD-S (170M), SEDD-M (424M) (Lou et al., 2023), MDM(1B) (Gong et al., 2024), DiffuLLama(7B)(Nie et al., 2024), DiffuGPT-S (127M), DiffuGPT-M (355M)(Nie et al., 2024), and autoregressive models GPT2-S (127M) , GPT2-M (355M),GPT-2 (1.5B) (Radford et al., 2019), TinyLlama(1.1B) (Zhang et al., 2024) and Llama-2 (7B) (Touvron et al., 2023).

Setup. We equip existing mask diffusion models MDM (1.1B) and DiffuLLama (7B) with our path planning and compare them with the default ancestral sampling results. For P2, we sweep the stochasticity strength from 0 to 2.0 with a step size of 0.2 and report the best results.

Results. As shown in Table 8, equipping with P2, we consistently improve the generation performance in the five benchmarks. In tasks that require more extensive global bidirectional reasoning, math reasoning GSM8K story infilling ROCStories, and code generation, P2 consistently exhibits improved performance by a large margin compared to the ancestral sampling. Compared to AR models that rely solely on left-toright modeling capabilities, P2 presents impressive generation accuracy; in code generation, where P2 achieves 17.6% pass@1 rate (vs. 1.7% of respective autoregressive model Llama-2 (7B)). In math reasoning, P2 enables a 1.1B-parameter MDM to outperform 7B-parameter Llama2 (60.9% vs. 58.5%). We attribute the success of P2 in complex language generation task to the remasking that corrects potential mistakes made in previous steps and promotes MDMs to generate robust answers.

H.7 Additional Language Generation Tasks

H.7.1 BREAKING THE REVERSE CURSE

Benchmark. Berglund et al. (2023) introduced the concept of the reverse curse, which refers to the difficulty of ARMs in generalizing bidirectional relationships. Specifically, this occurs when a model is trained on information in the form "A is B" but fails to infer the reverse relationship "B is A." For

Table 9: Results on breaking the reverse curse: Performance comparison of models on Description-ToName and NameToDescription tasks. Metrics include accuracy (Acc.) and BLEU scores (BLEU) for both same and reverse directions.

	DescriptionToName		**NameToDescription**			
	Same direction Acc. ↑	**Reverse direction Acc. ↑**	**Same direction Acc. ↑**	**BLEU ↑ BLEU ↑**	**Reverse direction Acc. ↑**	**BLEU ↑ BLEU ↑**
GPT3 (175B)	97	0	50	-	0	-
Llama-2 (13B)	99	0	-	74	-	19
T5 (3B)	**100**	0	47	**87**	0	20
MDM (1.1B)	97	92	**49**	76	**37**	67
MDM (1.1B) + **Path Planning (P2)**	96	**93**	48	78	36	**68**

example, a model trained on the fact "Valentina Tereshkova was the first woman to travel to space" may not correctly answer the reverse question "Who was the first woman to travel to space?" This limitation raises concerns about whether large language models genuinely possess logical reasoning capabilities.

Baselines. We compare with the leading AR models including GPT3 (175B), Llama-2 (13B), and the T5 consisting of both bidirectional encoder and unidirectional decoder, finetuned on the reverse curse dataset. For the MDM baseline, We use the existing MDM (1.1B) from Gong et al. (2024) with its default greedy ancestral sampling strategy.

Setup. It is observed in SMDM(Gong et al., 2024) that MDMs easily break the reverse curse, displaying near-perfect reverse accuracy where ARs achieve 0 accuracy. We follow SMDM(Gong et al., 2024) and evaluate MDMs on the same reverse curse dataset used by Berglund et al. (2023), which consists of fictitious statements in the format "⟨name⟩ is ⟨description⟩" and the reversals. We use the pretrained MDMs and baseline results from SMDM (Gong et al., 2024) which on these statements and assess their performance using questions not seen during training. Following the same protocol as (Berglund et al., 2023), we generate responses and report the exact match accuracy and use the BLEU metric (Papineni et al., 2002) to evaluate the quality of name-to-description generation (Lv et al., 2023).

results. As shown in Table 9, both the T5 model and ARMs achieve zero accuracy and low BLEU scores with reverse queries. Equipping with P2, we successfully improve the accuracy of MDMs in Reverse direction of task Description To Name and the BLEU metric of Name To Description in both directions.

H.8 Protein Sequence Generation

Performance Across Length Categories. We analyze the performance of protein generation models across various sequence lengths, ranging from 200 to 800 base pairs. Certain models, such as ProGen, do not generate proteins of fixed lengths; therefore, we group results into length categories to facilitate meaningful comparisons. As shown in Figure 6, the performance of these models varies with length, highlighting their capabilities and limitations across diverse length categories.

H.8.1 Design Space of P2.

We explore the design space of our proposed P2 framework using key metrics, including pLDDT, pAE, pTM, entropy, and diversity. As illustrated in Figure 7, P2 demonstrates a strong ability to balance structural accuracy and diversity, underscoring its versatility and robustness in protein generation tasks.

H.8.2 Ablation Study on the Planner.

We investigate the impact of planner size on model performance through an ablation study. Figure 8 shows how varying the planner size affects key metrics such as pLDDT and diversity. These results emphasize the importance of planner size in optimizing the quality and consistency of generated sequences.

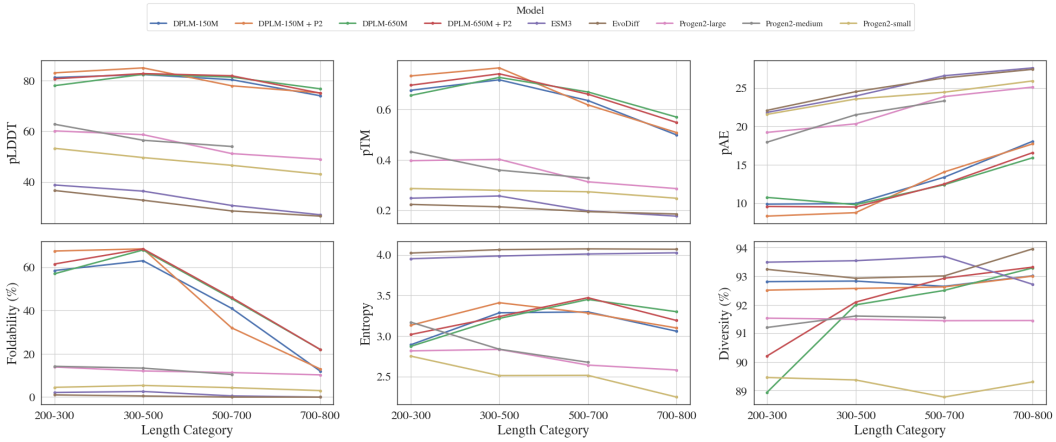

Figure 6: Protein Sequence Generation Benchmark: Performance across length categories (200–800).

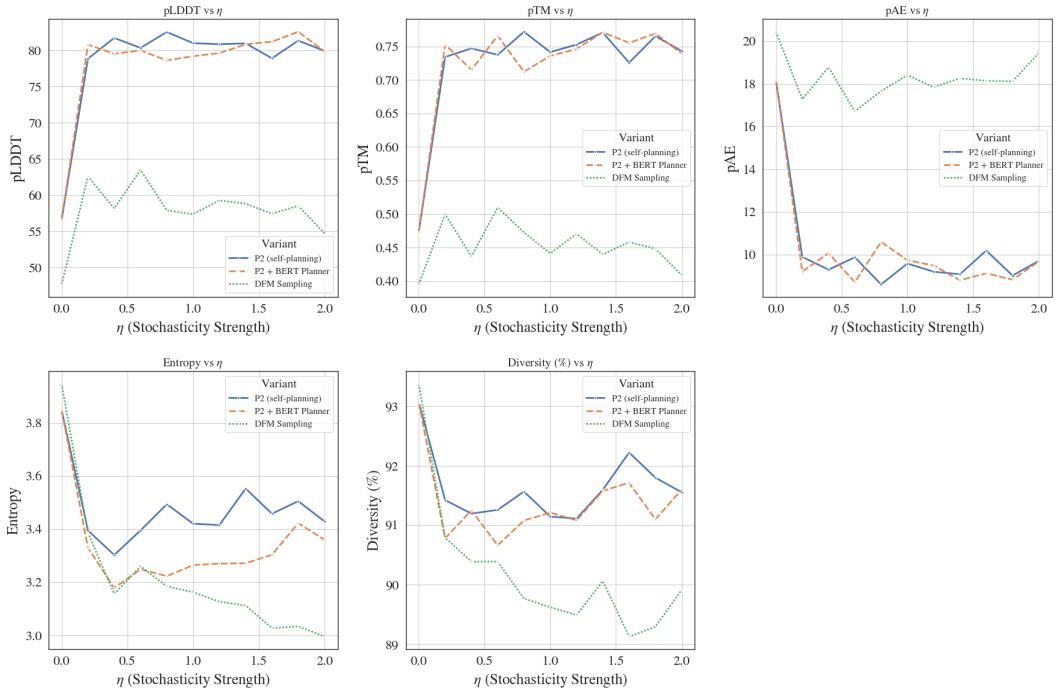

Figure 7: Design space of P2, characterized by pLDDT, pAE, pTM, entropy, and diversity metrics.

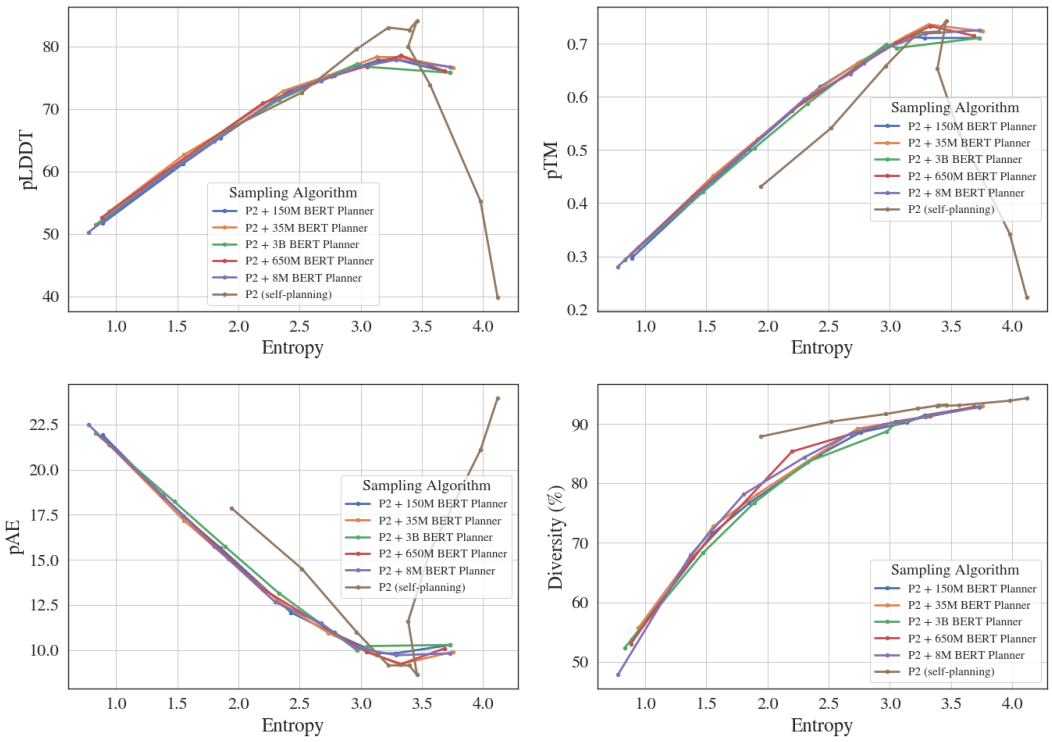

Figure 8: Ablation study of planner size and its impact on protein generation performance.

H.8.3 Inference-Time Scaling: Performance vs. Sampling Time.

To evaluate the trade-off between inference time and performance, we investigate how sampling time scales with model performance. These results will be detailed in future work, but they highlight the scalability of our approach for efficient protein generation.

H.8.4 Generated Protein Sequences and Their Predicted Structures.

We fold the protein sequences generated by our model using ESMFold and visualize their predicted structures in Figures 9–12. For each length category—200, 300, 400, 500, 600, 700, and 800—we display 15 representative proteins. These visualizations highlight the structural diversity and consistency of the generated sequences, providing evidence of the model's ability to predict biologically plausible structures across diverse lengths.

H.8.5 RNA RDM Training Implementation.

The RNA RDM follows the same discrete diffusion described in (Zheng et al., 2023). The RDM was trained using a machine mounted with 4 A100 GPUs, each with 40GB memory. The training implementation is otherwise identical to the second-stage fine-tuning described in (Wang et al., 2024a), where we continued from a RiNALMo (Penić et al., 2024) checkpoint instead of ESM-2 (Lin et al., 2023).

H.8.6 Visualizing the Predicted Structures of Generated RNA Sequences.

We extend our analysis to RNA sequence generation by folding RNA sequences of 200 base pairs using AlphaFold3 (Abramson et al., 2024). The predicted folding structures, visualized in Figures 13 and 14, highlight the diversity and consistency of the RNA structures generated by the model. Particularly, predicted structures exhibit greater diversity as sequence length increases, as is observed in nature, while their pLDDT's mirroring those computed for natural sequences. We also include the predicted secondary structures of generated RNAs in Figure 15. These results demonstrate

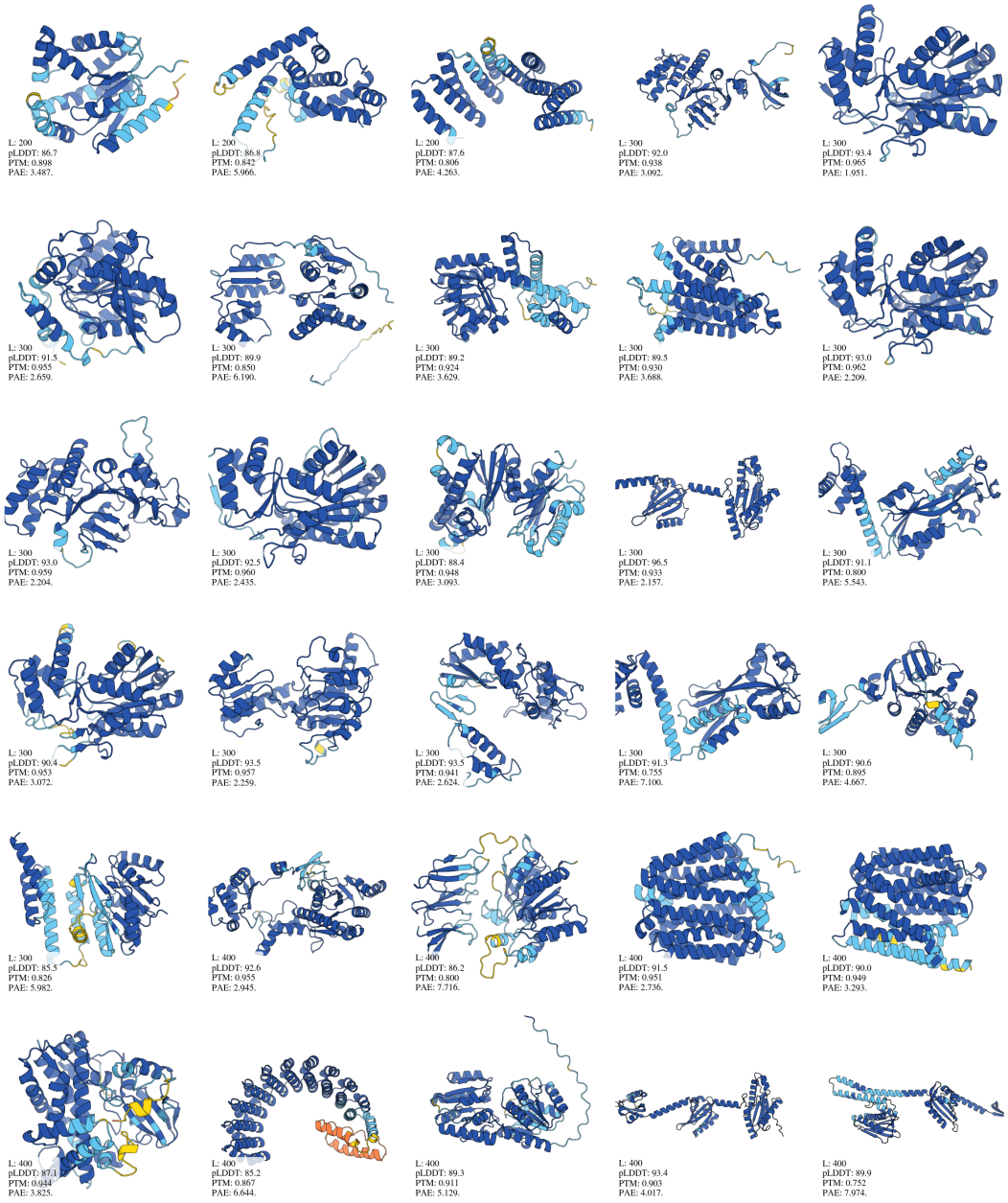

Figure 9: Predicted structures of generated protein sequences (Group 1). Each panel represents structures generated for specific length categories.

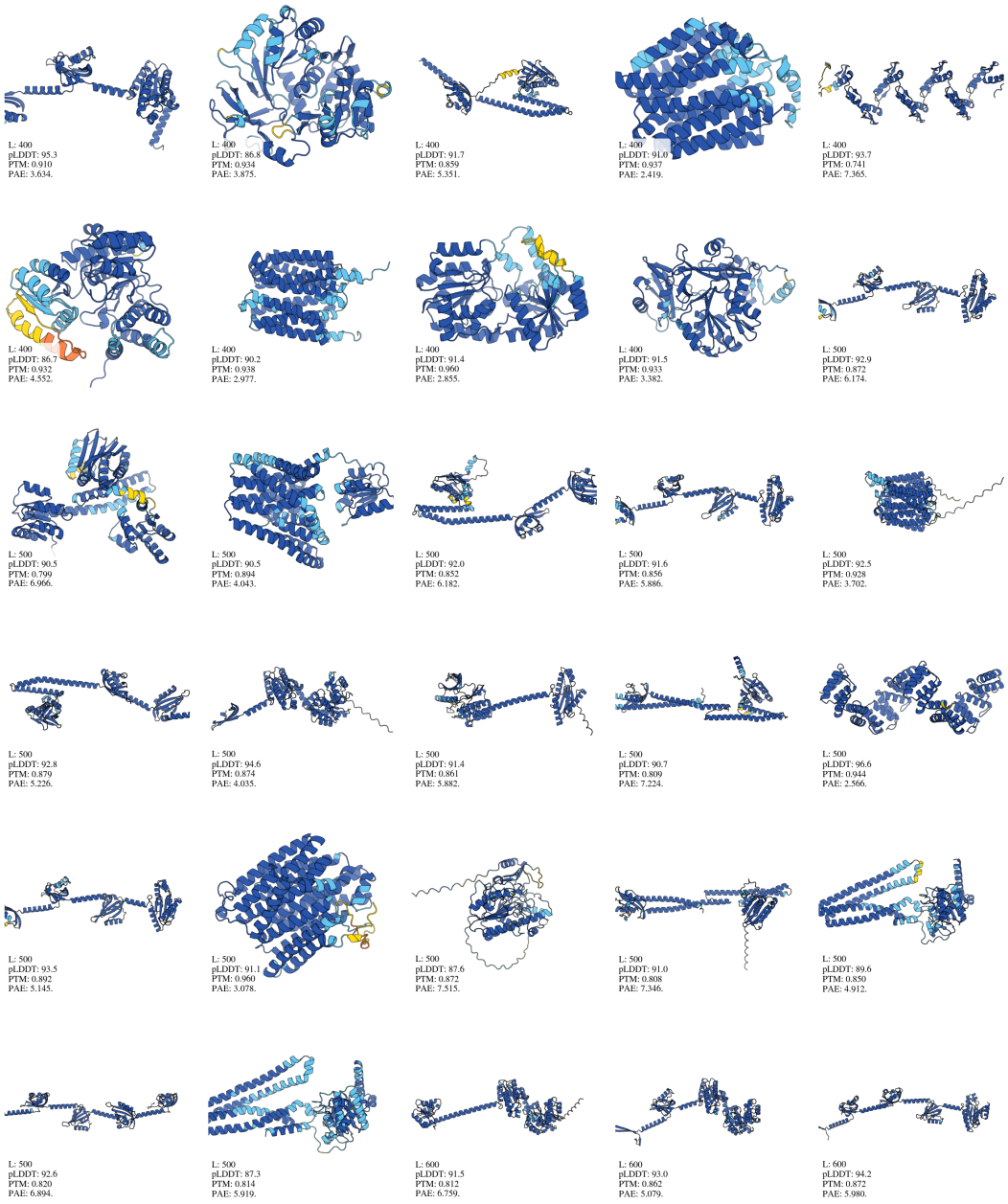

Figure 10: Predicted structures of generated protein sequences (Group 2). Each panel corresponds to different length categories.

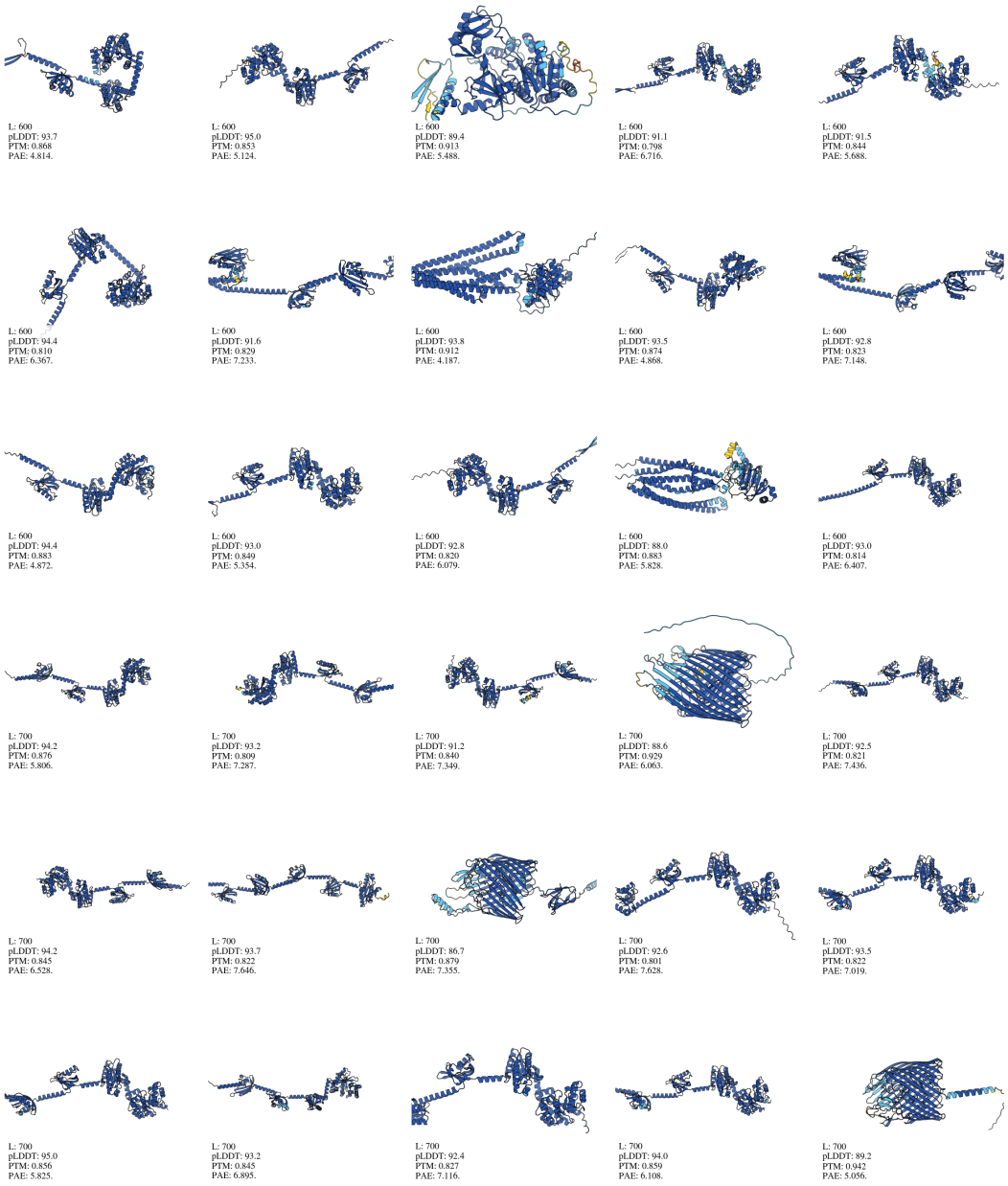

Figure 11: Predicted structures of generated protein sequences (Group 3). These structures illustrate the diversity and robustness of the generation process.

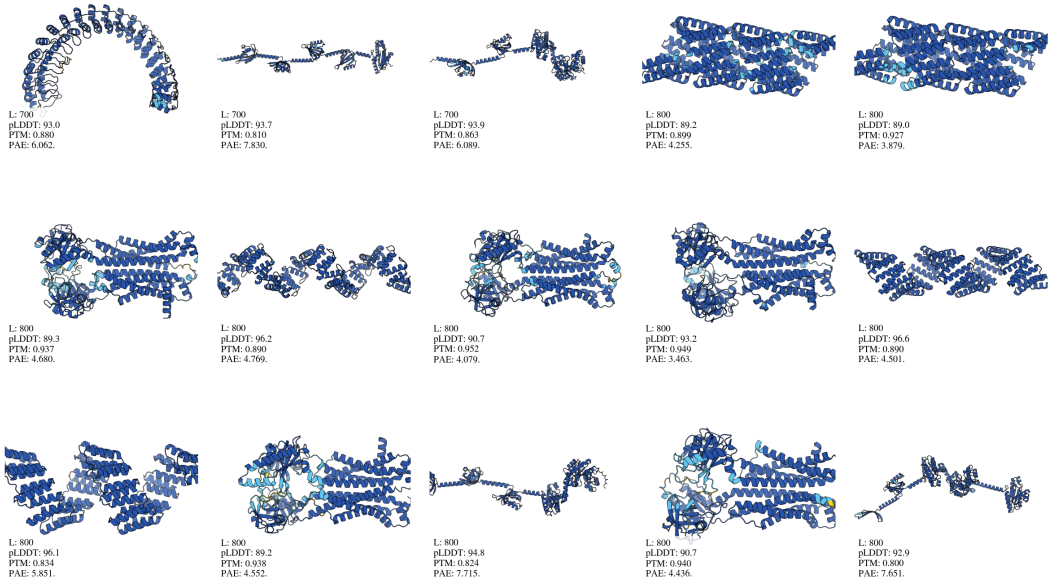

Figure 12: Predicted structures of generated protein sequences (Group 4). This group emphasizes structures for the longest generated sequences.

the model's ability to generate biologically plausible RNA sequences suitable for downstream applications.

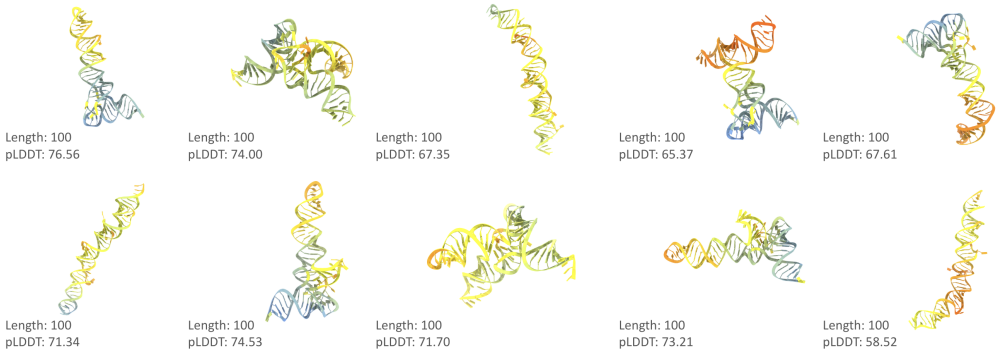

Figure 13: Predicted structures of additional generated RNA sequences (100 bps).

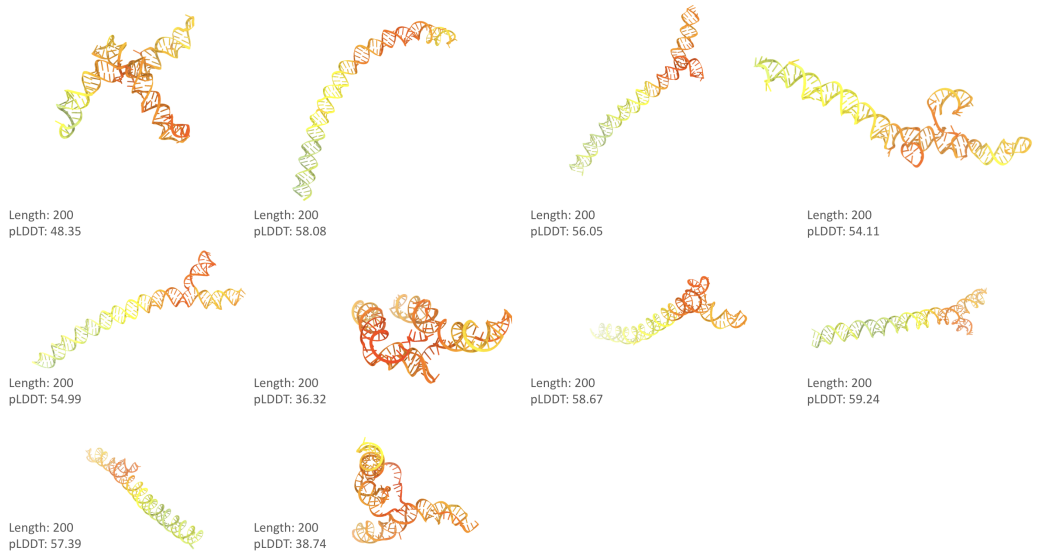

Figure 14: Predicted structures of generated RNA sequences (200 bps). This figure showcases the structural diversity of RNA sequences generated by the model as sequence length increases, which is observed in nature.

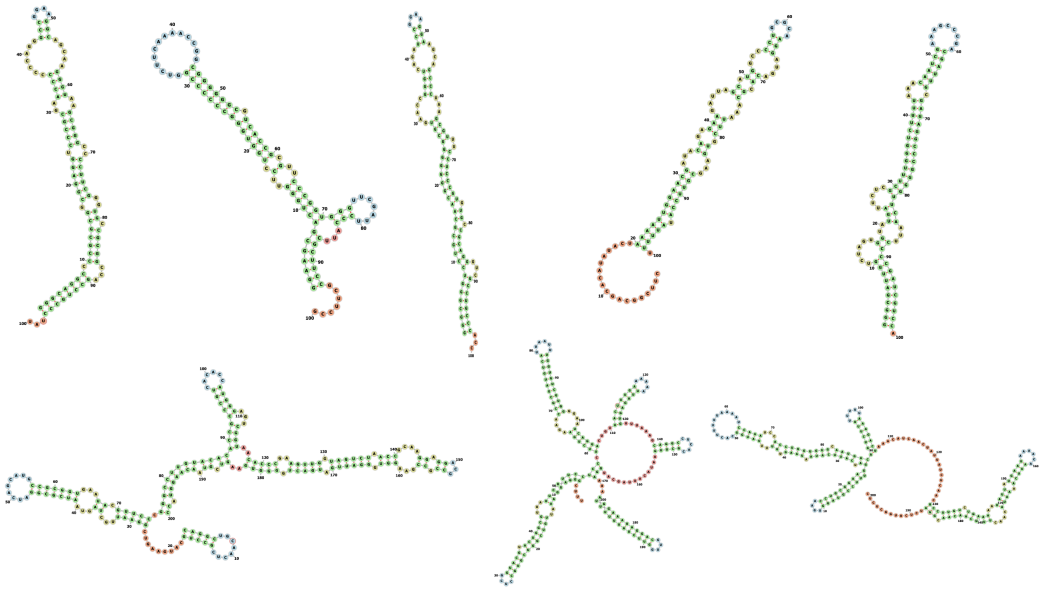

Figure 15: Predicted secondary structures of generated RNA sequences of length 100 (top) and 200 bp (bottom). Predictions were made using ViennaRNA (Lorenz et al., 2011) and visualized with forna (Kerpedjiev et al., 2015).