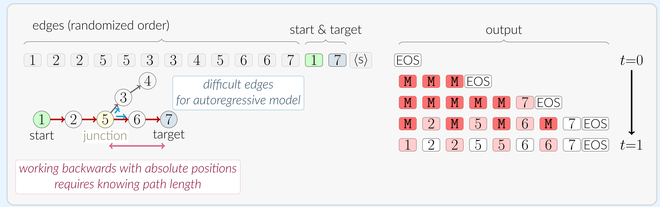
Insertion Based Sequence Generation with Learnable Order Dynamics

Dhruvesh Patel*[1] Benjamin Rozonoyer*[1] Gaurav Pandey[2] Tahira Naseem[2] Ramón Fernandez Astudillo[2] Andrew McCallum[1]

[1]University of Massachusetts Amherst [2]IBM Research

Insertion-based Masked Diffusion Models



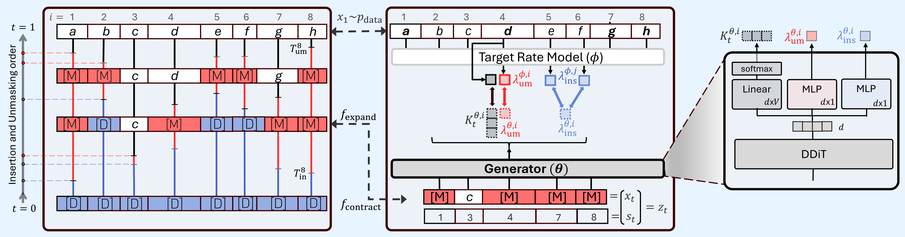
Generation process is a CTMC with two events:

- **Insertion:** Arbitrary number of mask tokens can be inserted in any gap
- **Unmasking:** Any existing mask token can be unmasked

Insertion and unmasking happen uniformly randomly and independently of the data point.

Learned Order Dynamics

The order of insertion and unmasking can be controlled by learning the respective event rates using an auxiliary target rate network.



The target process grows the final sequence x_1 from an empty sequence. For each position it samples an insertion time $T_{\text{in}}{}^{i}$ (empty → mask), then an unmasking time $T_{\text{um}}{}^{i}$ (mask → x_1^i). The unmasking times define the generation order: j comes after i when $T_{\text{um}}{}^{j} > T_{\text{um}}{}^{i}$.

The schedule network ϕ reads x_1 and outputs per-token insertion rates $\lambda_{\text{in}}^{\phi}$ and unmasking rates $\lambda_{\text{um}}^{\phi}$, which induce the event-time densities f^{ϕ}. On removing the placeholder [D] tokens from z_t one obtains the partial sequence x_t.

The generator θ is a transformer with an LM head and two rate heads, trained to match ϕ's projected target rates.

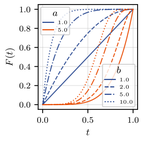
Contributions

- **ELBO.** Extend Discrete Flow Matching to projected partial sequences and derive the ELBO for **joint training** of the generator and target schedules.
- **Tractable Schedules.** Parameterize the target schedules with **Kumaraswamy CDFs**, enabling parallel event-time sampling and a closed-form likelihood.
- **Efficient Training.** Train both networks jointly with a **REINFORCE leave-one-out** estimator, without simulating full trajectories.
- **Practical Guide.** Detailed ablations to elucidate the design choices and trade-offs in the framework.

Target Schedule

- Order is controlled purely through per-position **hazard rates**, leaving the terminal distribution unchanged—so order can be learned without breaking tractability.
- Per-token insertion and unmasking event-time CDFs, Kumaraswamy form.
- **Fix** shape a (position-independent constant); **learn** per-token rate multipliers $b_{\text{in}}^{\phi}(z_1), b_{\text{um}}^{\phi}(z_1)$ with a transformer.
- Per-position likelihood stays in closed form.

$$F_*(t) = 1 - (1 - t^a)^{b_*}$$
$$\lambda_*(t) = b_* \frac{a\, t^{a-1}}{1 - t^a}$$
$$\text{for } * \in \{\text{in}, \text{um}\}.$$



Training

- Jointly optimize generator θ and schedule network ϕ via a projected rate-matching loss.
- Order sampled with REINFORCE leave-one-out (RLOO); two rollouts as control variates. No trajectory simulation needed.
- **Schedule regularization** ($\mathcal{L}_{\text{reg}}$) pulls the mixture CDF $\bar{F}^{\phi}(t) = \frac{1}{L}\sum_i F^{\phi,i}(t)$ toward the diagonal $F(t) = t$, preventing event pile-up at $t = 0, 1$.

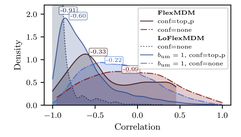
$$\mathcal{L}_{\text{um}} = \mathbb{E}_{\phi} \sum_{i \text{ masked}} \left[-\lambda_{\text{um}}^{\phi} \log K^{\theta,i} + D_{\psi}(\lambda_{\text{um}}^{\phi}, \lambda_{\text{um}}^{\theta})\right]$$
$$\mathcal{L}_{\text{in}} = \mathbb{E}_{\phi} \sum D_{\psi}\left(\textstyle\sum_j \lambda_{\text{in}}^{\phi}, \lambda_{\text{in}}^{\theta,i}\right)$$
$$\mathcal{L}_{\text{reg}} = \sum_k w_k (\bar{F}_{\text{in}}^{\phi}(t_k) - t_k)^2 + \sum_k w_k (\bar{F}_{\text{um}}^{\phi}(t_k) - t_k)^2$$

while not converged **do**
  Sample $z_1 \sim p_{\text{data}}$, $t \sim \mathcal{U}(0,1)$
  $(a^{\phi,i}, b^{\phi,i}) \leftarrow \phi(z_1)\ \forall i$
  for $k = 1, 2$ **do**
    $T_{\text{in}}{}^{i,(k)} \sim F_{\text{in}}^{\phi,i}$
    $T_{\text{um}}{}^{i,(k)} \sim \text{TruncSample}(T_{\text{in}}{}^{i,(k)}, F_{\text{um}}^{\phi,i})$
    $z_t^{i,(k)} \leftarrow$ [D]/[M]/z_1^i as t passes $T_{\text{in}}{}^{i,(k)}, T_{\text{um}}{}^{i,(k)}$
    $x_t^{(k)}, \mathbf{s}_t^{(k)} \leftarrow f_{\text{contract}}(z_t^{(k)})$
    $\mathcal{L}_k \leftarrow \mathcal{L}^{\phi,\theta}(x_t^{(k)}, \mathbf{s}_t^{(k)}, z_1)$
  end for
  $(\theta, \phi) \leftarrow (\theta, \phi) - \eta \nabla \mathcal{L}$
end while

Star-Graph Traversal

Controlled task with a known optimal order: work backwards from endpoints to the junction.

Model	medium	hard
ARM	75.0	23.0
MDM	36.5	21.0
FlexMDM	91.3	7.4
LoFlexMDM	**93.0**	**88.1**

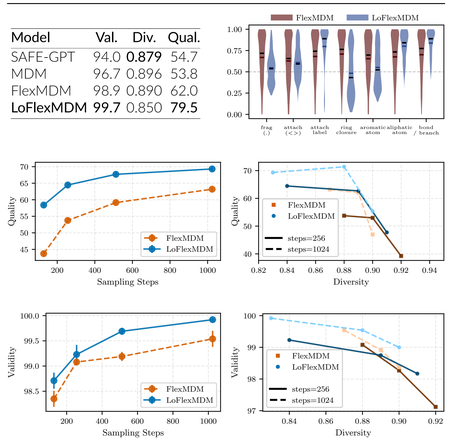


Design choices.

- **Fix the unmasking rate (b_{um}=1).** A trainable b_{um} destabilizes training; fixing it gives a large gain on hard while still leaving room to learn order.
- **Separate > shared backbone.** Sharing one backbone for generator and schedule network hurts noticeably.

De Novo Molecule Generation

Model	Val.	Div.	Qual.
SAFE-GPT	94.0	0.879	54.7
MDM	96.7	0.896	53.8
FlexMDM	98.9	0.890	62.0
LoFlexMDM	**99.7**	0.850	**79.5**



Generation Trajectories

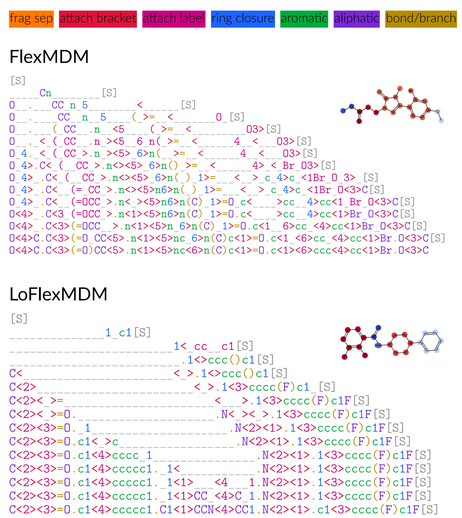


Figure 1: Example *de novo* trajectories. Each line is a subsampled step. **LoFlexMDM** resolves ring closures and fragment separators first, then fills in atoms and bond/branch tokens, whereas FlexMDM mixes categories throughout. Molecule atoms are coloured by generation time (early → late).

Fragment-Constrained Generation

Task	Quality (%) ARM	MDM	FlexMDM	LoFlexMDM
Linker design	21.7	21.9	50.8	**51.7**
Motif ext.	18.6	30.1	46.9	**53.6**
Scaffold dec.	10.0	31.8	39.0	**40.5**
Superstructure	14.3	34.8	35.8	**37.0**

Table 1. Quality across constrained tasks.

Takeaways

- Learned generation order improves quality without compromising tractability of DFM training.
- Up to **+17.5%** and **+6.7%** improvement in de novo and constrained generation quality over FlexMDM, respectively.
- The learned order is **interpretable**.

Project