H-GLOW: ACCELERATING GRAPH GENERATION VIA HAZARD-GUIDED DISCRETE FLOW MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete flow matching (DFM) offers a promising approach to graph generation; however, generating high-quality graphs with few steps remains challenging due to the strong dependencies between nodes and edges. Although DFM decouples training from sampling, enabling flexible sampling strategies without retraining, existing graph DFM samplers often rely on task-specific heuristics, substantially expanding the hyperparameter space. Moreover, graph DFM models still require sampling budgets comparable to those of diffusion-based approaches, leaving their potential for accelerating graph generation only partially realized. In this work, we introduce H-Glow, a hazard-guided framework for accelerating graph generation built on the decomposition of Continuous-Time Markov chain (CTMC) generators into hazards and conditional jump distributions. H-Glow leverages these quantities to plan graph edits and dynamically adjust the sampling time to graph activity. H-Glow can be adapted to pretrained graph DFM models, substantially improving the generation quality with fewer steps. We further introduce a novel graph DFM model based on the conditional generator matching framework, which directly predicts hazards and conditional jumps, and show that its training objective decomposes exactly into hazard matching and hazard-weighted jump prediction. Experiments on molecular and synthetic graph benchmarks demonstrate improved generation quality with substantially fewer sampling steps.

1 INTRODUCTION

Generating molecular structures using generative models has the potential to significantly accelerate materials design (Sanchez-Lengeling & Aspuru-Guzik, 2018) and drug discovery (Zhang et al., 2025). Discrete-state graph diffusion models (Vignac et al., 2022) offer a promising approach for progressively refining categorical node and edge attributes through iterative denoising updates. However, their denoising process is defined on a fixed temporal grid, and sampling is therefore tied to the discrete time points used during training.

Continuous-time graph diffusion models overcome this restriction by learning to denoise at arbitrary time points, allowing flexible time discretization during inference (Campbell et al., 2022; Xu et al., 2024; Siraudin et al., 2024). More recently, discrete flow matching (DFM) has achieved state-of-the-art performance on molecular and synthetic graph benchmarks (Qin et al., 2025). Despite the sampling acceleration achieved by flow-based models in various domains, graph flow models remain computationally costly and often require as many sampling steps as diffusion-based methods, leaving their potential for efficient generation only partially realized.

Efficient sampling is particularly challenging for graphs because node and edge attributes are strongly interdependent. Existing samplers update multiple nodes and edges in parallel without accounting for potential conflicts between updates, which can lead to violations of structural or chemical constraints (Qin et al., 2025; Luo et al., 2026). The approximation errors become more pronounced as the sampling budget decreases. To address this, prior work exploits the training-sampling decoupling of DFM and uses specialized sampling-time interventions, including target guidance, additional stochasticity, and tailored sampling-time discretizations (Campbell et al., 2024; Qin et al., 2025; Luo et al., 2026) to improve generation quality. However, these interventions expand the hyperparameter space and often require extensive tuning for each task and sampling budget.

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

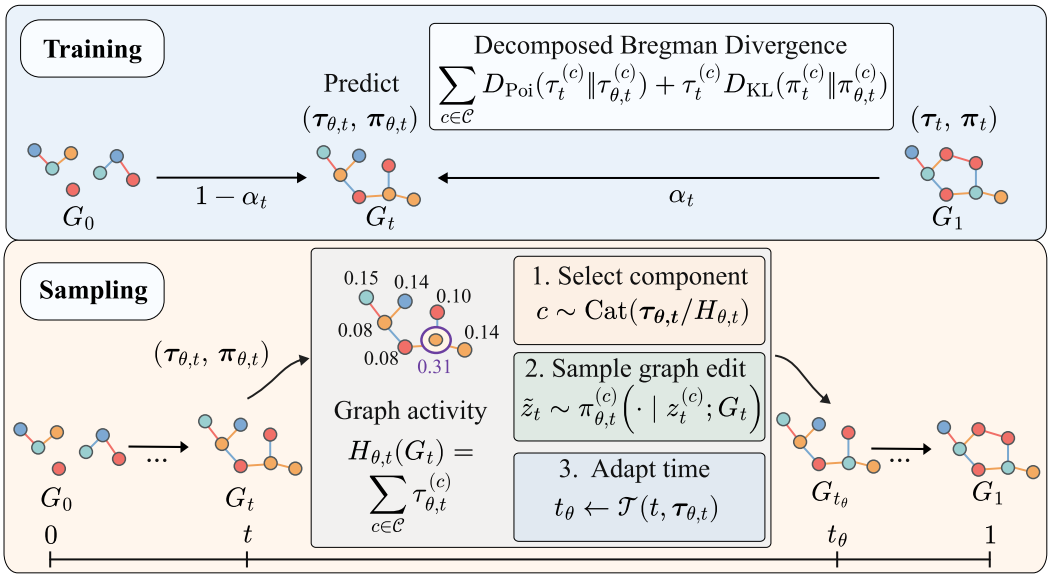


Figure 1: Overview of H-Glow.

In this work, we introduce *H-Glow*, a novel framework for graph generation built on the decomposition of CTMC generators into hazards and conditional jump distributions, as presented in Fig. 1. H-Glow decomposes the generative process into three key components: **(i)** hazard-guided planning that prioritizes graph edits and coordinates node and edge updates; **(ii)** *conditional jump distributions* that guide the refinement of selected graph components; and **(iii)** a hazard-guided clock that dynamically adjusts the sampling-time distortion to the graph's predicted activity. H-Glow substantially accelerates graph generation and can be used as a drop-in sampler for pretrained models with accessible CTMC generators. We further introduce a structured DFM model for graph generation, trained using the generator-matching framework (Holderrieth et al., 2025). H-Glow adopts a factorized CTMC parameterization to learn node and edge hazards, as well as conditional jump distributions (Li et al., 2026), thereby directly supervising the quantities that guide our sampling strategy.

We summarize our contributions as follows:

- **Hazard-guided graph sampling.** We introduce H-Glow, a novel sampler that effectively leverages predicted hazards and jump distributions to prioritize graph edits and dynamically adapt sampling-time distortions to graph activity.
- **Accelerating pretrained models.** H-Glow accelerates pretrained models with accessible CTMC generators without retraining, auxiliary networks, or complex sampling interventions.
- **Structured graph generator learning framework.** We introduce a graph DFM model that directly learns node and edge hazards, as well as conditional jump distributions, rather than relying on a denoiser network. Under the CTMC factorization, we derive an exact decomposition of the generator-matching objective into hazard matching and hazard-weighted jump prediction.
- **Empirical evaluation.** H-Glow achieves high validity and strong FCD performance on molecular benchmarks using substantially fewer sampling steps. We further demonstrate substantial acceleration in sampling for pretrained graph DFM models.

2 Background

2.1 Continuous-Time Markov Chains (CTMC)

A CTMC is a right-continuous stochastic process $\{z_t\}_{t\in[0,1]}$ taking values in a finite discrete state space $\mathcal{Z} = \{1, \ldots, Z\}$ and satisfies the Markov property $p(z_t \mid z_s, \ldots, z_0) = p(z_t \mid z_s)$ for all $0 \le s < t \le 1$. A CTMC is characterized by its *holding times* and *jumps*. Specifically, a state z_t

remains unchanged over a period of *holding time*. When the holding time ends, the state jumps to the next state $z_{t+\mathrm{d}t}$ according to a jump probability distribution. The holding-and-jumping process is then repeated. This two-step process is described using a generator matrix $\mathbf{R}_t \in \mathbb{R}^{Z \times Z}$, analogous to the velocity field in continuous state spaces, which defines transition rates between states:

$$p_{t+\mathrm{d}t|t}(\tilde{z} \mid z_t) = \delta(z_t, \tilde{z}) + R_t(z_t, \tilde{z})\mathrm{d}t, \tag{1}$$

where δ is the Kronecker delta, $R_t(z_t, \tilde{z})$ denotes an entry in the rate matrix, and $R_t(z_t, \tilde{z})\mathrm{d}t$ is the probability of state transition from z_t to $\tilde{z}$ in the next infinitesimal time step $\mathrm{d}t$. The generator matrix satisfies $R_t(z_t, \tilde{z}) \geq 0$, for $z_t \neq \tilde{z}$, and $R_t(z_t, z_t) = -\sum_{k \neq z_t} R_t(z_t, k)$ to ensure that the transition probabilities sum to 1. Any generator rate matrix can be factorized as:

$$R_t(z_t, \tilde{z}) = \tau_t(z_t)\, \pi_t(\tilde{z} \mid z_t), \qquad \tilde{z} \neq z_t. \tag{2}$$

Here, $\tau_t(z_t) := \sum_{\tilde{z} \neq z_t} R_t(z_t, \tilde{z}) \geq 0$ is the hazard of the state z_t, and $\pi_t(\tilde{z} \mid z_t)$ is the destination distribution conditional on a jump occurring, where:

$$\pi_t(\tilde{z} \mid z_t) \geq 0, \qquad \pi_t(z_t \mid z_t) = 0, \quad \text{and} \quad \sum_{\tilde{z} \neq z_t} \pi_t(\tilde{z} \mid z_t) = 1. \tag{3}$$

A large hazard indicates a high instantaneous jump rate from z_t, whereas $\pi_t(\cdot \mid z_t)$ defines where it moves. In practice, the trajectory can be simulated with finite time intervals Δt using Euler steps:

$$z_{t+\Delta t} \sim \mathrm{Cat}\Big(\delta(z_t, \cdot) + R_t(z_t, \cdot)\Delta t\Big). \tag{4}$$

2.2 Discrete flow matching

A discrete flow model constructs a probability path p_t, i.e., the generative flow, connecting a simple prior $p_0 = p_{\text{noise}}$ to the target data distribution $p_1 = p_{\text{data}}$. During inference, samples are generated by simulating a CTMC whose marginal distribution follows the probability path. To achieve this, we need to construct a rate matrix $\mathbf{R}_t$ that generates p_t, which is used to simulate the trajectory from $t = 0$ to $t = 1$ following Eq. (1). The marginal distribution and the generator matrix are related by a conservation law, the *Kolmogorov forward equation*:

$$\partial_t p_t(z_t) = \sum_{j \neq z_t} R_t(j, z_t)\, p_t(j) - \sum_{j \neq z_t} R_t(z_t, j)\, p_t(z_t), \tag{5}$$

which expresses the time derivative of the marginal probability at z_t as the net balance between incoming and outgoing probability mass. However, since it is intractable to directly derive a generator for p_t, flow matching relies on a simpler datapoint conditional flow $p_{t|1}(\cdot \mid z_1)$ satisfying:

$$p_t(z_t) = \mathbb{E}_{z_1 \sim p_{\text{data}}}\left[p_{t|1}(z_t \mid z_1)\right], \tag{6}$$

where the conditional flow starts from noise at $t = 0$, and converges to the datapoint z_1 at $t = 1$. Discrete flow matching (DFM) (Gat et al., 2024) linearly interpolates towards z_1 as follows:

$$p_{t|1}(z_t \mid z_1) = \alpha_t \delta(z_t, z_1) + (1 - \alpha_t) p_0(z_t), \tag{7}$$

where α_t is a scheduler increasing monotonically from $\alpha_0 = 0$ to $\alpha_1 = 1$.

Generator construction. One choice of a valid conditional generator that generates the conditional probability path $p_{t|1}(\cdot \mid z_1)$ is

$$R_t(z_t, \tilde{z} \mid z_1) = \frac{\dot{\alpha}_t}{1 - \alpha_t}\left[\delta(\tilde{z}, z_1) - \delta(z_t, \tilde{z})\right]. \tag{8}$$

Gat et al. (2024) show that a rate matrix generating the marginal path p_t is obtained by averaging the conditional rates over the posterior $p_{1|t}(\cdot \mid z_t)$: $R_t(z_t, \tilde{z}) = \mathbb{E}_{z_1 \sim p_{1|t}(\cdot|z_t)}\left[R_t(z_t, \tilde{z} \mid z_1)\right]$. Gat et al. (2024) derive the marginal rate matrix in closed form as follows:

$$R_t(z_t, \tilde{z}) = \frac{\dot{\alpha}_t}{1 - \alpha_t}\left[p_{1|t}(\tilde{z} \mid z_t) - \delta(z_t, \tilde{z})\right]. \tag{9}$$

Learning. DFM trains a neural network $p^\theta_{1|t}(\cdot \mid z_t)$ to approximate the true denoising distribution using the cross entropy loss, i.e., learning to predict the clean datapoint z_1 given a noisy latent z_t:

$$\mathcal{L}_{\text{ce}} = \mathbb{E}_{t \sim \mathcal{U}(0,1),\, z_1 \sim p_1(\cdot),\, z_t \sim p_{t|1}(\cdot|z_1)}\left[-\log p^\theta_{1|t}(z_1 \mid z_t)\right] \tag{10}$$

Once trained, the generator is constructed as $R_{\theta,t}(z_t, \tilde{z}) = \frac{\dot{\alpha}_t}{1 - \alpha_t}\left[p^\theta_{1|t}(\tilde{z} \mid z_t) - \delta(z_t, \tilde{z})\right]$. Together with the initial distribution p_0, this generator defines a CTMC, which can be simulated using Eq. (4).

3 CHALLENGES OF EXISTING GRAPH DFM MODELS.

Discrete flow matching has been recently extended for graphs (Qin et al., 2025), where samples are generated by simulating a multivariate CTMC factorizing over node and edge attributes:

$$p_{t+\mathrm{d}t|t}\big(\tilde{z}^{1:D} \mid z_t^{1:D}\big) = \prod_{c=1}^{D}\Big[\delta\big(\tilde{z}^{(c)}, z_t^{(c)}\big) + R_{\theta,t}^{(c)}\big(z_t^{(c)}, \tilde{z}^{(c)}; G_t\big)\mathrm{d}t\Big], \tag{11}$$

$t \in [0,1]$, $z_t^{1:D}, \tilde{z}_t^{1:D} \in \mathcal{Z}^D$, D is the number of components, and $z^{(c)}$ denotes a node or edge attribute. In the following, we discuss the challenges of existing graph DFM models.

Sampling complexity. Graph DFM models often rely on sophisticated sampling procedures, including predictor-corrector updates (Campbell et al., 2022), target guidance and additional stochasticity (Campbell et al., 2024), as well as carefully designed sampling-time distortions (Qin et al., 2025). In particular, Luo et al. (2026); Qin et al. (2025) design non-uniform time discretizations that concentrate computation in specific regions of the generation trajectory where graph validity is most sensitive. While effective, these techniques introduce additional hyperparameters and design choices, requiring substantial inference-time tuning for each task and sampling budget.

Sampling efficiency. DFM offers the potential for efficient generation with few sampling steps. However, realizing this advantage is particularly challenging for graphs, where node and edge attributes are strongly interdependent. In practice, DFM methods simulate the process in Eq. (11) using finite-step approximations such as τ-leaping, which hold transition rates fixed over a finite interval and apply the resulting component updates simultaneously. These fixed rates neglect interactions between updates within the interval, thereby introducing approximation errors when multiple graph components change. These errors are more pronounced in the low-step regime, where larger intervals tend to produce more simultaneous updates. Accurately coordinating node and edge transitions is therefore crucial for maintaining sample quality at low NFE.

4 HAZARD-GUIDED SAMPLING USING H-GLOW

We introduce *H-Glow*, a novel hazard-guided sampler for graph generation that builds on the decomposition of CTMC generators to hazards and conditional jump distributions.

Notations. We represent an undirected graph with N nodes by its node and edge attributes as $G = \big(x^{1:n:N}, e^{1:i<j:N}\big)$, where $x^{1:n:N} = \{x^{(n)}\}_{n=1}^{N}$ and $e^{1:i<j:N} = \{e^{(ij)}\}_{1\le i<j\le N}$. Here, $x^{(i)} \in \mathcal{X} = \{1,\dots,X\}$ denotes a categorical attribute of node i, and $e^{(ij)} \in \mathcal{E} = \{1,\dots,E\}$ denotes a categorical edge attribute. Following Vignac et al. (2022), we associate an edge between every pair of nodes, and reserve one category in $\mathcal{E}$ to represent the absence of an edge. $z^{(c)}$ denotes the attribute of a component c, corresponding to a node attribute $x^{(n)}$ or an edge attribute $e^{(ij)}$.

4.1 DECOMPOSITION OF GRAPH GENERATORS

In the following, we assume access to a model f_θ from which we can construct valid componentwise generators with transition rates $R_{\theta,t}^{(c)}(z_t^{(c)}, \tilde{z}; G_t)$. For each component c, we factorize the off-diagonal elements following Eq. (1) as:

$$R_{\theta,t}^{(c)}\Big(z_t^{(c)}, \tilde{z}; G_t\Big) = \tau_{\theta,t}^{(c)}(z_t^{(c)}; G_t)\,\pi_{\theta,t}^{(c)}\Big(\tilde{z} \mid z_t^{(c)}; G_t\Big), \quad \tilde{z} \neq z_t^{(c)}, \tag{12}$$

where $\tau_{\theta,t}^{(c)}$ is the exit hazard and $\pi_{\theta,t}^{(c)}$ is the jump distribution. To ensure that each row of the generator sums to zero, we set the diagonal entries to:

$$R_{\theta,t}^{(c)}\Big(z_t^{(c)}, z_t^{(c)}; G_t\Big) = -\sum_{\tilde{z}\neq z_t^{(c)}} R_{\theta,t}^{(c)}\Big(z_t^{(c)}, \tilde{z}; G_t\Big) = -\tau_{\theta,t}^{(c)}\Big(z_t^{(c)}; G_t\Big) \tag{13}$$

The overall graph generator $R_{\theta,t}\Big(G_t, \widetilde{G}\Big)$ factorizes over nodes and edges, see Proposition 1.

Proposition 1 (Graph generator). *The graph generator factorizes, i.e., it can be constructed as the sum of the generators of the individual component paths (Campbell et al., 2022; Xu et al., 2024):*

$$R_{\theta,t}(G_t, \widetilde{G}) = \sum_n \delta\left(G_t^{\backslash(n)}, \widetilde{G}^{\backslash(n)}\right) R_{\theta,t}^{X,(n)}\left(x_t^{(n)}, \tilde{x}; G_t\right) + \sum_{i<j} \delta\left(G_t^{\backslash(ij)}, \widetilde{G}^{\backslash(ij)}\right) R_{\theta,t}^{E,(ij)}\left(e_t^{(ij)}, \tilde{e}; G_t\right),$$

where $\delta\left(G_t^{\backslash(n)}, \widetilde{G}^{\backslash(n)}\right)$ *and* $\delta\left(G_t^{\backslash(ij)}, \widetilde{G}^{\backslash(ij)}\right)$ *check if* G_t *and* $\widetilde{G}$ *are identical except for node* (n) *or edge* (ij)*, respectively.* $\mathbf{R}_{\theta,t}^{X,(n)} \in \mathbb{R}^{\mathcal{X}\times\mathcal{X}}$ *and* $\mathbf{R}_{\theta,t}^{E,(ij)} \in \mathbb{R}^{\mathcal{E}\times\mathcal{E}}$ *are the node and edge generators.*

Under this construction, the graph CTMC changes one node or edge attribute at a time, with transition rates that depend on the entire graph G_t.

Proposition 2 (Graph hazard). *For the generator in Proposition 1, the total graph hazard is given by the sum of the componentwise node and edge hazards* $\tau_{\theta,t}^{X,(n)}$ *and* $\tau_{\theta,t}^{E,(ij)}$ *of* $\mathbf{R}_{\theta,t}^{X,(n)}$ *and* $\mathbf{R}_{\theta,t}^{E,(ij)}$*:*

$$H_{\theta,t}\left(G_t\right) = \sum_n \tau_{\theta,t}^{X,(n)}\left(x_t^{(n)}; G_t\right) + \sum_{i<j} \tau_{\theta,t}^{E,(ij)}\left(e_t^{(ij)}; G_t\right) \tag{14}$$

The graph hazard in Eq. (14) provides a non-negative scalar measure of graph activity, with larger values indicating a higher instantaneous rate of graph edits.

4.2 Graph generation planning under H-Glow.

Building on the hazard-jump decomposition, we design the H-Glow sampler to fully leverage the predicted hazards and jumps, as well as the available sampling budget T, to guide graph edits.

Component selection. *Hazard-based selection.* A natural choice is to sample components based on their contribution to the graph hazard:

$$c_t \sim \mathrm{Cat}\left(\left\{\tau_{\theta,t}^{(c)}/H_{\theta,t}\right\}_{c\in\mathcal{C}}\right), H_{\theta,t} > 0 \tag{15}$$

Confidence-weighted selection. Alternatively, we select components with confident jumps:

$$w_{\theta,t}^{(c)} = \tau_{\theta,t}^{(c)} \max_{\tilde{z}\neq z_t^{(c)}} \pi_{\theta,t}^{(c)}\left(\tilde{z} \mid z_t^{(c)}; G_t\right), \tag{16}$$

and sample in proportion to $w_{\theta,t}^{(c)}$.

Conditional jump. Once c_t is selected for an update, we sample its jump following:

$$\widetilde{z}_t \sim \pi_{\theta,t}^{(c_t)}\left(\cdot \mid z_t^{(c_t)}; G_t\right). \tag{17}$$

Algorithm 1 H-Glow: Hazard-guided sampling

Input: model f_θ, budget T, threshold T_{stop}
Initialize $G \sim p_0$, $t \leftarrow 0$, $k \leftarrow 0$
Let $\mathcal{C} = \mathcal{C}_X \cup \mathcal{C}_E$
while $k < T$ **do**
  Construct $\mathbf{R}_{\theta,t}^{(c)}$ and derive $\{(\tau_{\theta,t}^{(c)}, \pi_{\theta,t}^{(c)})\}_{c\in\mathcal{C}}$
  $H_{\theta,t} \leftarrow \sum_{c\in\mathcal{C}} \tau_{\theta,t}^{(c)}$
  if $H_{\theta,t} < T_{\text{stop}}$ **then return** G
  $c \sim \mathrm{Cat}\left(\left\{\tau_{\theta,t}^{(c)}/H_{\theta,t}\right\}_{c\in\mathcal{C}}\right)$
  $\widetilde{z} \sim \pi_{\theta,t}^{(c)}$
  $G \leftarrow \mathrm{Update}(G, c, \widetilde{z})$
  $t \leftarrow 1 - \frac{1-\alpha_t}{2\dot{\alpha}_t}\left(\overline{\tau}_{\theta,t}^{X} + \overline{\tau}_{\theta,t}^{E}\right)$
  $k \leftarrow k + 1$
end while
return G

As this jump excludes the current state z_t, each conditional jump changes exactly one component. Hazard-based selection and conditional jumps follow Gillespie's event-selection (Gillespie, 1976).

Hazard-based time allocation. Sampling time distortion is particularly important when generating graphs under a limited number of steps T. H-Glow benefits from the predicted hazards to *dynamically* adapt the time according to the graph activity. Higher graph activity motivates a finer discretization with more refinement steps to revise earlier predictions, whereas low activity motivates a coarser discretization with fewer refinement steps.

To derive a sampling-time policy, we first look into the hazards implied by the DFM generator. Under the DFM construction in Eq. (9), the conditional hazard simplifies to: $\tau_t^{(c)}(G_t \mid G_1) = \frac{\dot{\alpha}_t}{1-\alpha_t}\mathbf{1}\left\{z_t^{(c)} \neq z_1^{(c)}\right\}$. The marginal hazard averages conditional hazards over clean graphs:

$$\tau_t^{(c)}(G_t) = \mathbb{E}_{G_1|G_t}\left[\tau_t^{(c)}(G_t \mid G_1)\right] = \frac{\dot{\alpha}_t}{1-\alpha_t}\Pr\left(z_1^{(c)} \neq z_t^{(c)} \mid G_t\right). \tag{18}$$

Therefore, hazards estimate the remaining componentwise disagreement between the noisy latent G_t and the clean graph. We leverage this to construct the following hazard-guided sampling time:

$$\widehat{t} = 1 - \frac{1-\alpha_t}{2\dot{\alpha}_t}\left(\frac{1}{N}\sum_{i=1}^{N}\tau_{\theta,t}^{X,(i)} + \frac{1}{\binom{N}{2}}\sum_{1\le i<j\le N}\tau_{\theta,t}^{E,(ij)}\right) \tag{19}$$

This measures the average remaining graph activity. We note that we normalize node and edge hazards to prevent edge activity from dominating the clock update and to avoid larger time jumps in larger graphs. We provide the full derivation in Section C.1.

Importantly, the time allocation in Eq. (19) defines a graph-dependent sequence of evaluations that can move forward or backward in time. When time moves from t to $s < t$, we renoise the graph by independently resampling nodes and edges from their corresponding priors with probability $1 - \frac{\alpha_s}{\alpha_t}$, allowing the sampler to revisit earlier predictions (Xu et al., 2023; Lugmayr et al., 2022).

Early stopping. Once the graph hazard falls below a threshold T_{stop}, H-Glow terminates the trajectory. This prevents further graph edits when the model predicts low graph activity, reducing the risk of disrupting an otherwise valid graph. This is particularly crucial for graph generation, where even a single incorrect node or edge update can violate structural or chemical constraints.

4.3 Accelerating pretrained models.

H-Glow can be applied directly to pretrained models with accessible CTMC generators. Existing graph DFM models train a denoising network $p_{1|t}^{\theta}(\cdot \mid G_t)$ to predict distributions over clean node and edge attributes, and construct generators following Eq. (9). We decompose these generators post hoc into hazards and conditional jump distributions to guide graph edits. In Fig. 2, we show that H-Glow can significantly accelerate graph generation without additional training. We summarize our sampling strategy in Algorithm 1, omitting the generator construction and renoising step for clarity.

5 H-Glow: Discrete Flow Matching Model for Graphs

Motivated by the acceleration gains achieved by H-Glow using pretrained models, we propose a structured generator-learning framework for graphs. In the following, we propose *H-Glow*, a simple discrete flow-matching model for graph generation that directly parameterizes the generator using hazards and conditional jump distributions.

Noising. We design the noising trajectory by independently adding noise to each node and edge:

$$p_{t|1}\left(G_t \mid G_1\right) = \prod_n p_{t|1}\left(x_t^{(n)} \mid x_1^{(n)}\right)\prod_{i<j} p_{t|1}\left(e_t^{(ij)} \mid e_1^{(ij)}\right), \tag{20}$$

where $p_{t|1}\left(x_t^{(n)} \mid x_1^{(n)}\right)$ and $p_{t|1}\left(e_t^{(ij)} \mid e_1^{(ij)}\right)$ are defined according to Eq. (7). We employ empirical marginal priors over node and edge types $p_0^{\mathcal{X}}$ and $p_0^{\mathcal{E}}$ (Vignac et al., 2022).

Conditional targets. For each node or edge component c, we obtain training targets by factorizing the conditional generator $R_t(z_t^{(c)}, \tilde{z} \mid G_1)$ into its exit hazard $\tau_t^{(c)}(z_t^{(c)} \mid G_1)$ and conditional jump distribution $\pi_t^{(c)}(\cdot \mid z_t^{(c)}, G_1)$.

Optimization. We parameterize node and edge hazards and conditional jump distributions using a neural network: $f_\theta(G_t, t) = \left(\boldsymbol{\tau}_{\theta,t}^X, \boldsymbol{\pi}_{\theta,t}^X, \boldsymbol{\tau}_{\theta,t}^E, \boldsymbol{\pi}_{\theta,t}^E\right)$. We follow the conditional generator matching framework in Holderrieth et al. (2025), and use the Bregman Divergence loss to optimize the continuous-time ELBO derived for finite-state jump processes:

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{G_1\sim p_1,\\ t\sim\mathcal{T},\\ G_t\sim p_{t|1}(\cdot|G_1)}}\left[\sum_{\widetilde{G}\neq G_t}\left(R_{\theta,t}\left(G_t,\widetilde{G}\right) - R_t\left(G_t,\widetilde{G}\mid G_1\right)\log R_{\theta,t}\left(G_t,\widetilde{G}\right)\right)\right] + C \tag{21}$$

Algorithm 2 Training H-Glow

```
Input: Graph dataset 𝒟
while f_θ has not converged do
    Sample G_1 ~ 𝒟
    Sample t ~ 𝒯
    Sample G_0 ~ p_0
    Sample G_t ~ p_{t|1}(· | G_1)
    Compute (τ_t^X, π_t^X, τ_t^E, π_t^E)
    (τ_{θ,t}^X, π_{θ,t}^X, τ_{θ,t}^E, π_{θ,t}^E) ← f_θ(G_t, t)
    loss ← ℒ(θ)
    optimizer.step(loss)
end while
```

Table 2: Large molecule generation performance on MOSES and GuacaMol.

			MOSES							GuacaMol				
Model	Class	Steps	Val. ↑	Unique. ↑	Novelty ↑	Filters ↑	FCD ↓	SNN ↑	Scaf. ↑	Val. ↑	V.U. ↑	V.U.N. ↑	KLD. ↑	FCD ↑
Training set	–	–	100.0	100.0	0.0	100.0	0.01	0.64	99.1	100.0	100.0	0.0	99.9	92.8
DiGress (Vignac et al., 2022)	Diff.	500	85.7	100.0	95.0	97.1	1.19	0.52	14.8	85.2	85.2	85.1	92.9	68.0
DisCo (Xu et al., 2024)	Diff.	500	88.3	100.0	97.7	95.6	1.44	0.50	15.1	86.6	86.6	86.5	92.6	59.7
Cometh (Siraudin et al., 2024)	Diff.	500	86.1	100.0	96.0	97.3	1.27	0.50	16.4	94.4	94.4	93.5	94.1	67.4
Cometh-PC (Siraudin et al., 2024)	Diff.	500	90.5	99.9	92.6	99.1	1.27	0.54	16.0	98.9	98.9	97.6	96.7	72.7
SimGFM (Luo et al., 2026)	Flow	50	86.1	100.0	95.1	98.6	1.44	0.51	16.9	98.5	98.5	98.3	93.6	77.1
SimGFM (Luo et al., 2026)	Flow	200	89.4	100.0	94.6	98.9	1.08	0.55	17.3	99.2	99.2	98.5	94.6	76.4
SimGFM (Luo et al., 2026)	Flow	500	91.1	100.0	93.9	99.2	1.19	0.56	14.4	99.2	99.2	98.6	95.5	73.5
DeFoG (Qin et al., 2025)	Flow	50	82.9	99.9	95.1	97.8	1.79	0.53	19.8	91.7	91.7	91.2	92.3	57.9
DeFoG (Qin et al., 2025)	Flow	500	91.5	99.9	90.0	99.2	1.47	0.55	11.5	99.0	99.0	97.9	97.7	73.8
H-Glow (Ours)	Flow	50	89.5	100.0	94.4	99.1	0.98	0.55	18.8	95.2	95.2	93.6	89.9	69.3
H-Glow (Ours)	Flow	100	89.9	99.9	93.0	99.2	0.96	0.56	16.0	99.2	99.2	98.1	96.9	77.7
H-Glow (Ours)	Flow	200	91.9	99.9	93.1	99.4	1.08	0.58	15.5	99.4	99.4	97.9	96.8	77.3

We show in Theorem 1 that, under H-Glow's factorization, this objective separates exactly into a hazard matching and a hazard-weighted jump prediction optimization.

Theorem 1 (Decomposed graph generator matching objective)**.** *Assume that the conditional target and learned graph generators follow the single-component construction in Proposition 1 and admit hazard-jump decompositions. Then, the conditional generator matching objective decomposes as:*

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{G_1 \sim p_1,\, t \sim \mathcal{T}, \\ G_t \sim p_{t|1}(\cdot|G_1)}} \left[\sum_{n=1}^{N} \mathcal{L}_{\theta,t}^{X,(n)} + \sum_{1 \le i < j \le N} \mathcal{L}_{\theta,t}^{E,(ij)} \right], \tag{22}$$

where, for each graph component c,

$$\mathcal{L}_{\theta,t}^{(c)} = D_{\mathrm{Poi}}\left(\tau_t^{(c)} \,\middle\|\, \tau_{\theta,t}^{(c)}\right) + \tau_t^{(c)} D_{\mathrm{KL}}\left(\pi_t^{(c)} \,\middle\|\, \pi_{\theta,t}^{(c)}\right). \tag{23}$$

Here, $D_{\mathrm{Poi}}(a\|b) = a \log \frac{a}{b} - a + b$.

We provide a proof in Section A. Theorem 1 shows that learning the graph generator reduces to two separate prediction tasks for every node and edge. The hazard head learns the instantaneous rate at which the component changes, whereas the jump head learns the destination distribution conditional on a jump occurring. Importantly, the jump loss is weighted by the target hazard. Therefore, components with a high hazard receive greater weight, whereas the jump loss vanishes for components with zero target hazard. We summarize the training algorithm in Algorithm 2.

6 Experiments

We evaluate H-Glow on molecular and synthetic graph benchmarks. To isolate the effect of the network architecture, we build the H-Glow model using the graph transformer in Vignac et al. (2022). We describe the datasets, baselines, and model architectures in Section B.

6.1 Molecular graph generation

Small molecule generation. We evaluate H-Glow on QM9 (Wu et al., 2018) and report the results in Table 1. With only 50 sampling steps, H-Glow achieves the lowest FCD among the evaluated models, surpassing both 50- and 500-step baselines while maintaining high validity.

Table 1: Performance on QM9.

Model	Steps	Valid ↑	Relaxed Validity ↑	Unique ↑	FCD ↓
Training set	–	99.3	99.5	99.2	0.03
DiGress (Vignac et al., 2022)	500	99.0	–	96.2	–
CatFlow (Eijkelboom et al., 2024)	–	99.8	–	100.0	0.44
DisCo (Xu et al., 2024)	500	99.3	–	–	–
Cometh (Siraudin et al., 2024)	500	99.6	–	96.8	0.25
SID (Boget, 2025)	500	99.7	–	–	0.50
CID (Boget, 2025)	500	99.9	–	–	1.76
SimGFM (Luo et al., 2026)	50	99.7	99.8	96.3	0.13
SimGFM (Luo et al., 2026)	500	99.8	99.9	94.9	0.31
DeFoG (Qin et al., 2025)	50	98.9	99.2	96.2	0.26
DeFoG (Qin et al., 2025)	500	99.3	99.4	96.3	0.12
H-Glow (Ours)	20	99.2	99.5	95.0	0.37
H-Glow (Ours)	50	99.1	99.5	96.2	0.10

Large molecules. We evaluate on GuacaMol (Brown et al., 2019) and Moses (Polykovskiy et al., 2020) and report results in Table 2. We compare with recent discrete diffusion and flow matching models. H-Glow achieves the best FCD on both datasets with only 100 steps, outperforming the 500-step baselines. H-Glow achieves a high validity with 200 steps.

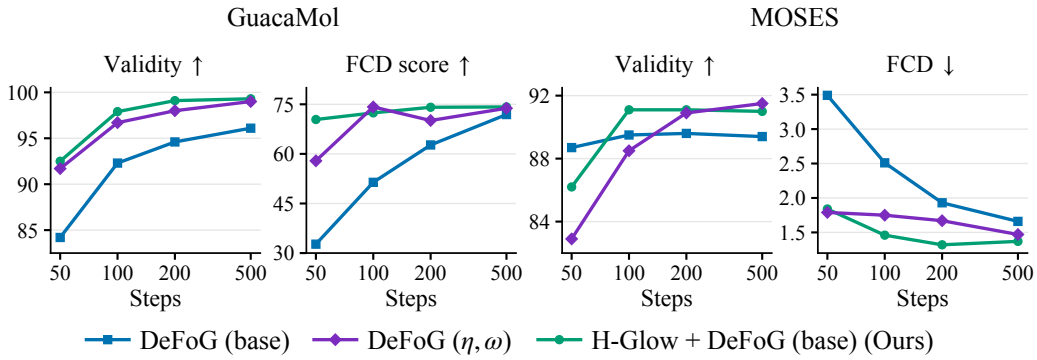


Figure 2: We sample from the base pretrained DeFoG (Qin et al., 2025) model using Euler sampler (blue), Euler sampling with added target guidance and stochasticity (purple), and our H-Glow sampler (green). H-Glow substantially accelerates sampling while maintaining high sample quality.

6.2 Accelerating pretrained DFM models

H-Glow exploits the generator decomposition to accelerate pretrained graph DFMs as a drop-in sampler, without retraining or added sampling complexity. We evaluate it on pretrained DeFoG (Qin et al., 2025), and compare against its base sampler and DeFoG(η, ω), which adds stochasticity and target guidance during inference (Campbell et al., 2024), see Fig. 2. On GuacaMol and MOSES, H-Glow - DeFoG matches or exceeds the 500-step base Euler sampler in validity and FCD using only 100 steps, and achieves quality comparable to 500-step DeFoG(η, ω) using 200 steps.

6.3 Synthetic graphs

The sampler in Algorithm 1 achieves strong empirical performance on both small and large molecular graphs using few NFEs, see Section 6.1. This is possible because the sampling initialization from the empirical node and edge priors p_0^X and p_0^E enforces sparsity, where most candidate edges are assigned to the no-bond category, and only a few graph edits can be sufficient to generate a valid molecule. For graphs requiring more extensive structural changes, where the number of required graph edits exceeds the available budget, we extend H-Glow to perform up to B edits in parallel at each step; see Algorithm 3. We evaluate on Planar (Martinkus et al., 2022) and Tree (Bergmeister et al., 2024), reporting means and standard deviations over 5 runs of 40 graphs (Table 3). With 20 steps, H-Glow achieves 95.0% and 93.5% V.U.N. on Planar and Tree, respectively, and 100% at 200 steps with competitive Ratios.

Table 3: Performance on synthetic graphs.

		Planar		Tree	
Model	Steps	V.U.N. ↑	Ratio ↓	V.U.N. ↑	Ratio ↓
Training set	–	100.0	1.0	100.0	1.0
EDGE (Chen et al., 2023)	1000	0.0	431.4	0.0	850.7
DiGress (Vignac et al., 2022)	1000	77.5	5.1	90.0	1.6
HSpectre (Bergmeister et al., 2024)	–	95.0	2.1	100.0	4.0
GruM (Jo et al., 2023)	–	90.0	1.8	–	–
DisCo (Xu et al., 2024)	500	83.6	–	–	–
Cometh (Siraudin et al., 2024)	500	92.5	–	–	–
Cometh-PC (Siraudin et al., 2024)	–	99.5	–	–	–
CatFlow (Eijkelboom et al., 2024)	–	80.0	–	–	–
SimGFM (Luo et al., 2026)	20	94.0 ± 4.4	2.3 ± 0.6	88.0 ± 4.8	2.5 ± 0.9
SimGFM (Luo et al., 2026)	50	99.5 ± 1.0	1.8 ± 0.5	97.0 ± 1.0	2.0 ± 0.7
SimGFM (Luo et al., 2026)	200	100.0 ± 0.0	9.3 ± 2.6	99.5 ± 1.0	1.5 ± 0.2
SimGFM (Luo et al., 2026)	1000	100.0 ± 0.0	21.0 ± 1.5	100.0 ± 0.0	2.1 ± 0.8
DeFoG (Qin et al., 2025)	50	95.0 ± 3.2	3.2 ± 1.1	73.5 ± 9.0	2.5 ± 1.0
DeFoG (Qin et al., 2025)	1000	99.5 ± 1.0	1.6 ± 0.4	96.5 ± 2.6	1.6 ± 0.4
H-Glow (Ours)	20	95.0 ± 3.5	1.9 ± 0.5	93.5 ± 4.6	2.2 ± 0.8
H-Glow (Ours)	50	99.0 ± 1.2	1.9 ± 0.4	98.0 ± 2.9	2.7 ± 0.9
H-Glow (Ours)	100	99.0 ± 1.2	1.8 ± 0.3	98.5 ± 2.0	1.8 ± 0.4
H-Glow (Ours)	200	100.0 ± 0.0	2.4 ± 0.2	100.0 ± 0.0	2.0 ± 0.8

6.4 Ablations

Sample time distortion. Uniform time discretization is suboptimal for graph generation, as specific regions of the generation trajectory can benefit from finer control. Prior work employs specialized time distortions to redistribute steps along the generation trajectory. For example, on Planar, Qin et al. (2025) use polydec-2, while Luo et al. (2026) use $g_k(t) = 1 - (1-t)^k$ with $k = 10$. Our hazard-guided clock instead directly adapts sampling times to predicted graph activity. To isolate its contribution, we compare uniform, polynomial, and hazard-guided schedules under the H-Glow sampler on Planar, see Table 4. Our hazard-guided clock provides the best trade-off.

Table 4: Clock ablations.

	$T = 50$		$T = 100$	
Clock	VUN ↑	Ratio ↓	VUN ↑	Ratio ↓
Hazard-guided	99.0	1.86	99.0	1.81
Uniform	99.5	23.43	100.0	49.11
Polydec-2	100.0	20.20	100.0	42.62
Polydec-10	99.5	5.57	99.5	12.66
Polydec-20	95.5	3.88	99.5	6.08

Design choices. We ablate our design choices in Table 5. First, we find that confidence-weighted hazard selection can improve quality over hazard-only selection. Second, removing early stopping reduces sample quality, suggesting that further edits at low predicted activity can introduce errors. We report the number of function evaluations (NFE) in Table 10. Also, directly applying the time policy in Eq. (19) leads to large time jumps early in the generation process. To moderate this, we interpolate between the current time t_k and the hazard-guided time $\widehat{t}_k$:$t_{k+1} = (1-\gamma_k)t_k + \gamma_k \widehat{t}_k$, where $\gamma_k \in (0, 1]$ limits the time jumps, allowing further graph refinement, see Section D.

Table 5: Ablations on MOSES.

	$T = 50$		$T = 100$	
Variant	Val. ↑	FCD ↓	Val. ↑	FCD ↓
H-Glow	89.5	0.98	89.9	0.96
Hazard-only	88.1	1.10	89.2	1.12
w/o early stopping	88.1	1.41	88.7	1.82
w/o hazard annealing	88.5	1.43	87.9	1.39

Table 6: Sampler comparison.

	$T = 50$		$T = 100$	
Sampler	Val. ↑	FCD ↓	Val. ↑	FCD ↓
H-Glow	89.5	0.98	89.9	0.96
Euler	80.3	1.38	85.8	1.38
RVF	86.4	1.03	87.7	1.08

Sampler comparison. Using the same H-Glow model trained on MOSES, we compare our sampler with DeFoG's parallel Euler sampler (Campbell et al., 2022; Qin et al., 2025) and the RVF variant proposed in Luo et al. (2026), see Table 6. Compared to other samplers, our H-Glow sampler achieves higher validity and lower FCD at $T = 50$ than Euler and RVF at $T = 100$, demonstrating more effective use of the sampling budget. In Section D, we provide an ablation comparing denoiser and hazard–jump parameterizations under different sampling strategies.

7 Related work

Discrete diffusion and flow matching for graphs. Discrete-state graph diffusion models rely on categorical corruption (Austin et al., 2021; Vignac et al., 2022), with recent extensions introducing re-noising and learned critics to identify corrupted nodes and edges (Boget, 2025). Continuous-time formulations based on CTMCs allow flexible sampling-time discretization (Campbell et al., 2022; Xu et al., 2024; Siraudin et al., 2024), but their reverse dynamics remain tied to the forward noising process. In contrast, discrete flow matching (DFM) (Campbell et al., 2024; Gat et al., 2024) decouples training from the choice of sampling dynamics. Recent graph adaptations (Qin et al., 2025; Luo et al., 2026) further improve generation by optimizing rate matrices and time distortions. Diffusion- and flow-based graph models typically learn a denoiser network. Recently, Li et al. (2026) propose a diffusion model that directly parameterizes hazards and conditional jumps. We adapt this parameterization within the generator-matching framework (Holderrieth et al., 2025) and propose a structured generator-learning framework for graphs.

Improved diffusion sampling. Continuous-time approaches adopt finite-time approximators to simulate the CTMC trajectory, including independent Euler updates and τ-leaping (Campbell et al., 2022; Gat et al., 2024), where additional predictor-corrector steps can be used for further refinement. Liu et al. (2025) decompose language generation into planning and denoising, introducing a latent variable that indicates whether each token is corrupted. They further propose an adaptive sampling procedure guided by the planner's predictions. In contrast, H-Glow decomposes the generative process by directly exploiting the decomposition of CTMC generators into hazards and conditional jumps to guide sampling without an auxiliary planner network. In masked diffusion, Wang et al. (2026) enables iterative refinement of pretrained masked language models by remasking previously decoded tokens, allowing additional computation to improve sample quality.

8 Conclusion

We present H-Glow, a hazard-guided framework for accelerating graph generation. By factorizing generators into hazards and conditional jump distributions, H-Glow leverages the event structure of CTMC generators to coordinate graph edits. Under the same decomposition, we propose a novel DFM model for graph generation. We limit our experiments to accelerate pretrained DFM models; however, H-Glow can be readily applied to discrete diffusion models with accessible generators, which we leave for future work. Also, H-Glow benefits from the empirical prior over node and edge attributes, where initialized edges are dominated by the no-bond category, unlocking accelerated molecular generation within a few NFEs. To accommodate graphs that require extensive updates, we provide an extension of H-Glow that performs parallel updates per step. Future work could explore selection strategies that exploit graph structure to coordinate parallel edits more effectively.

AI USE STATEMENT

In this work, we used generative AI tools for language editing, including grammar, spelling, and word choice, as well as for assisting in polishing mathematical proofs and formulating ideas. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean or reformat datasets, support qualitative or thematic data analysis, or interpret results. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

In this work, we propose a generative model for discrete data and apply it to the generation of molecules and synthetic graphs. As generative models can be applied in various fields, we see potential positive and negative societal impacts. Generative modeling can accelerate scientific discovery; however, there are potential societal consequences, specifically those related to the generation of unsafe drugs.

REPRODUCIBILITY STATEMENT

The experimental setup is described in Section B and Section C.

REFERENCES

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *International Conference on Learning Representations*, volume 2024, pp. 27698–27736, 2024.

Yoann Boget. Simple and critical iterative denoising: A recasting of discrete diffusion in graph generation. *arXiv preprint arXiv:2503.21592*, 2025.

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3): 1096–1108, 2019.

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

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*, 2024.

Dexiong Chen, Markus Krimmel, and Karsten Borgwardt. Flatten graphs as sequences: Transformers are scalable graph generators. *Advances in Neural Information Processing Systems*, 38: 69071–69109, 2026.

Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. *arXiv preprint arXiv:2305.04111*, 2023.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International conference on machine learning*, pp. 2302–2312. PMLR, 2020.

Floor Eijkelboom, Grigory Bartosh, Christian A Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. *Advances in Neural Information Processing Systems*, 37:11735–11764, 2024.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

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.

Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of the web conference 2020*, pp. 1253–1263, 2020.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky TQ Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. In *International Conference on Learning Representations*, volume 2025, pp. 52153–52219, 2025.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pp. 10362–10383. PMLR, 2022.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. *arXiv preprint arXiv:2302.03596*, 2023.

Jingyuan Li, Xiaoyi Jiang, Fukang Wen, Wei Liu, Renqian Luo, Yi Zhu, Zuoqiang Shi, and Pipi Hu. Neural continuous-time markov chain: Discrete diffusion via decoupled jump timing and direction. *arXiv preprint arXiv:2604.15694*, 2026.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

Sulin Liu, Juno Nam, Andrew Campbell, Hannes Stärk, Yilun Xu, Tommi Jaakkola, and Rafael Gómez-Bombarelli. Think while you generate: Discrete diffusion with planned denoising. In *International Conference on Learning Representations*, volume 2025, pp. 81979–82026, 2025.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11451–11461. IEEE, 2022.

Chunyu Luo, Yuankai Luo, Xiao-Ming Wu, and Lei Shi. Simgfm: Simplifying discrete flow matching for graph generation. In *Forty-third International Conference on Machine Learning*, 2026.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, pp. 15159–15179. PMLR, 2022.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:565644, 2020.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: discrete flow matching for graph generation. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 50269–50326, 2025.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):140022, 2014.

Vinayak Rao and Yee Whte Teg. Fast mcmc sampling for markov jump processes and extensions. *Journal of Machine Learning Research*, 14(11), 2013.

Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

Antoine Siraudin, Fragkiskos D Malliaros, and Christopher Morris. Cometh: A continuous-time discrete-state graph diffusion model. *arXiv preprint arXiv:2406.06449*, 2024.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Guanghan Wang, Yair Schiff, Subham Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. *Advances in Neural Information Processing Systems*, 38:147282–147339, 2026.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi Jaakkola. Restart sampling for improving generative processes. *Advances in Neural Information Processing Systems*, 36:76806–76838, 2023.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. *Advances in Neural Information Processing Systems*, 37:79704–79740, 2024.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Kang Zhang, Xin Yang, Yifei Wang, Yunfang Yu, Niu Huang, Gen Li, Xiaokun Li, Joseph C Wu, and Shengyong Yang. Artificial intelligence in drug development. *Nature medicine*, 31(1):45–59, 2025.

A PROOFS

Theorem 1 (Decomposed graph generator matching objective)**.** *Assume that the conditional target and learned graph generators follow the single-component construction in Proposition 1 and admit hazard-jump decompositions. Then, the conditional generator matching objective decomposes as:*

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{G_1 \sim p_1,\, t \sim \mathcal{T}, \\ G_t \sim p_{t|1}(\cdot|G_1)}} \left[\sum_{n=1}^{N} \mathcal{L}_{\theta,t}^{X,(n)} + \sum_{1 \le i < j \le N} \mathcal{L}_{\theta,t}^{E,(ij)} \right], \tag{24}$$

where, for each graph component c,

$$\mathcal{L}_{\theta,t}^{(c)} = D_{\mathrm{Poi}}\left(\tau_t^{(c)} \,\middle\|\, \tau_{\theta,t}^{(c)} \right) + \tau_t^{(c)} D_{\mathrm{KL}}\left(\pi_t^{(c)} \,\middle\|\, \pi_{\theta,t}^{(c)} \right). \tag{25}$$

Here, $D_{\mathrm{Poi}}(a\|b) = a \log \frac{a}{b} - a + b$.

Proof. We write the conditional generator matching objective in its generalized KL form:

$$\mathcal{L}(\theta) = \mathbb{E}\Bigg[\sum_{\widetilde{G} \neq G_t} \Bigg(R_t(G_t, \widetilde{G} \mid G_1) \log \frac{R_t(G_t, \widetilde{G} \mid G_1)}{R_{\theta,t}(G_t, \widetilde{G})} - R_t(G_t, \widetilde{G} \mid G_1) + R_{\theta,t}(G_t, \widetilde{G}) \Bigg) \Bigg], \tag{26}$$

where the expectation is over $G_1 \sim p_1$, $t \sim \mathcal{T}$, and $G_t \sim p_{t|1}(\cdot \mid G_1)$.

Under the construction in Proposition 1, every transition with a nonzero off-diagonal rate changes exactly one component. Thus, the sum over graph transitions separates into sums over node and edge transitions.

Fix a graph component c. For readability, we define

$$r_t^{(c)}(\tilde{z}) = R_t^{(c)}\left(z_t^{(c)}, \tilde{z} \mid G_1 \right), \tag{27}$$

$$r_{\theta,t}^{(c)}(\tilde{z}) = R_{\theta,t}^{(c)}\left(z_t^{(c)}, \tilde{z}; G_t \right), \tag{28}$$

and suppress the conditioning arguments of the hazards and jump distributions. Therefore,

$$r_t^{(c)}(\tilde{z}) = \tau_t^{(c)} \pi_t^{(c)}(\tilde{z}), \qquad r_{\theta,t}^{(c)}(\tilde{z}) = \tau_{\theta,t}^{(c)} \pi_{\theta,t}^{(c)}(\tilde{z}), \quad \text{for } \tilde{z} \neq z_t^{(c)}. \tag{29}$$

The contribution of component c to the objective is

$$\mathcal{L}_{\theta,t}^{(c)} = \sum_{\tilde{z} \neq z_t^{(c)}} \left[r_t^{(c)}(\tilde{z}) \log \frac{r_t^{(c)}(\tilde{z})}{r_{\theta,t}^{(c)}(\tilde{z})} - r_t^{(c)}(\tilde{z}) + r_{\theta,t}^{(c)}(\tilde{z}) \right]. \tag{30}$$

First suppose that $\tau_t^{(c)} > 0$ and that the learned rates are positive wherever the target rates are positive. Substituting the hazard-jump decompositions and using

$$\sum_{\tilde{z} \neq z_t^{(c)}} \pi_t^{(c)}(\tilde{z}) = \sum_{\tilde{z} \neq z_t^{(c)}} \pi_{\theta,t}^{(c)}(\tilde{z}) = 1, \tag{31}$$

we obtain

$$\begin{aligned} \mathcal{L}_{\theta,t}^{(c)} &= \tau_t^{(c)} \log \frac{\tau_t^{(c)}}{\tau_{\theta,t}^{(c)}} - \tau_t^{(c)} + \tau_{\theta,t}^{(c)} + \tau_t^{(c)} \sum_{\tilde{z} \neq z_t^{(c)}} \pi_t^{(c)}(\tilde{z}) \log \frac{\pi_t^{(c)}(\tilde{z})}{\pi_{\theta,t}^{(c)}(\tilde{z})} \\ &= D_{\mathrm{Poi}}\left(\tau_t^{(c)} \,\middle\|\, \tau_{\theta,t}^{(c)} \right) + \tau_t^{(c)} D_{\mathrm{KL}}\left(\pi_t^{(c)} \,\middle\|\, \pi_{\theta,t}^{(c)} \right). \end{aligned} \tag{32}$$

Terms with zero target mass are interpreted using the standard convention $0 \log(0/q) = 0$. If a learned rate is zero where the target rate is positive, both sides are $+\infty$, so the identity also holds in this case. When $\tau_t^{(c)} = 0$, all target rates vanish and

$$\mathcal{L}_{\theta,t}^{(c)} = \sum_{\tilde{z} \neq z_t^{(c)}} r_{\theta,t}^{(c)}(\tilde{z}) = \tau_{\theta,t}^{(c)} = D_{\mathrm{Poi}}\left(0 \,\middle\|\, \tau_{\theta,t}^{(c)}\right). \tag{33}$$

The target jump distribution may then be chosen arbitrarily over alternative attributes, and its hazard-weighted KL term is defined as zero. Components with no alternative attributes have zero hazard and contribute zero.

Finally, summing over node and edge components gives

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{G_1 \sim p_1,\, t \sim \mathcal{T}, \\ G_t \sim p_{t|1}(\cdot|G_1)}} \left[\sum_{n=1}^{N} \mathcal{L}_{\theta,t}^{X,(n)} + \sum_{1 \leq i < j \leq N} \mathcal{L}_{\theta,t}^{E,(ij)} \right], \tag{34}$$

which proves the result. □

Proposition 2 (Graph hazard)**.** *For the generator in Proposition 1, the total graph hazard is given by the sum of the componentwise node and edge hazards:*

$$H_{\theta,t}\left(G_t\right) = \sum_{n} \tau_{\theta,t}^{X,(n)}\left(x_t^{(n)}; G_t\right) + \sum_{i<j} \tau_{\theta,t}^{E,(ij)}\left(e_t^{(ij)}; G_t\right). \tag{35}$$

Proof. By definition, the total graph hazard is the sum of the transition rates to all graphs different from the current graph:

$$H_{\theta,t}(G_t) = \sum_{G' \neq G_t} R_{\theta,t}(G_t, G'). \tag{36}$$

Since the graph generator permits only single-component transitions, every graph $G' \neq G_t$ with a nonzero transition rate differs from G_t in exactly one node or one edge attribute. Hence, the sum can be partitioned according to the component that changes:

$$\begin{aligned} H_{\theta,t}(G_t) = &\sum_{n} \sum_{\tilde{x} \neq x_t^{(n)}} R_{\theta,t}^{X,(n)}\left(x_t^{(n)}, \tilde{x}; G_t\right) \\ &+ \sum_{i<j} \sum_{\tilde{e} \neq e_t^{(ij)}} R_{\theta,t}^{E,(ij)}\left(e_t^{(ij)}, \tilde{e}; G_t\right). \end{aligned} \tag{37}$$

The componentwise hazards are defined as the corresponding sums of outgoing rates:

$$\tau_{\theta,t}^{X,(n)}\left(x_t^{(n)}; G_t\right) = \sum_{\tilde{x} \neq x_t^{(n)}} R_{\theta,t}^{X,(n)}\left(x_t^{(n)}, \tilde{x}; G_t\right), \tag{38}$$

$$\tau_{\theta,t}^{E,(ij)}\left(e_t^{(ij)}; G_t\right) = \sum_{\tilde{e} \neq e_t^{(ij)}} R_{\theta,t}^{E,(ij)}\left(e_t^{(ij)}, \tilde{e}; G_t\right). \tag{39}$$

Substituting these definitions yields equation 35. □

B Experimental Details

B.1 Datasets and Metrics

We evaluate H-Glow on the molecular graph benchmarks QM9, MOSES, and GuacaMol, following the preprocessing and dataset splits used in prior discrete graph generative models (Vignac et al., 2022; Siraudin et al., 2024; Qin et al., 2025; Xu et al., 2024). QM9 license is CC BY 4.0, MOSES and GuacaMol datasets are licensed under the MIT License. Molecules are represented as attributed undirected graphs, where nodes correspond to atom types and edges encode bond types, with the absence of a bond treated as an additional edge category. For each generated graph, the number of nodes is sampled from the empirical graph-size distribution of the training set. We summarize all data statistics in Table 7.

Table 7: Dataset statistics. E and X denote the numbers of present-edge and node categories, respectively. The no-edge category is excluded from E. B counts directed edge entries, following the convention of prior work, while N denotes the number of nodes.

Dataset	# Graphs	Train/Val/Test	E	X	Avg. B	Max. B	Avg. N	Max. N
QM9	130,831	97,734/20,042/13,055	4	4	18.9	28	8.8	9
MOSES	1,936,962	1,584,663/176,225/176,074	4	8	46.5	62	21.7	27
GuacaMol	1,398,213	1,118,633/69,926/209,654	4	12	60.4	176	27.8	88
Planar	200	128/32/40	1	1	355.7	362	64.0	64
Tree	200	128/32/40	1	1	126.0	126	64.0	64

QM9. We evaluate H-Glow on QM9 (Ramakrishnan et al., 2014). Following prior work (Vignac et al., 2022), we kekulize the molecules using the RDKit library and remove the hydrogen atoms. We use the same data split as Vignac et al. (2022) and train on 100K molecules. For the evaluation, we sample 10K molecules and report four metrics. The **Validity** is measured by sanitizing the sampled molecules and converting them to SMILES strings using the RDKit library. We also report the **Relaxed Validity**, which accounts for charged molecules. **Uniqueness** is measured across valid molecules, and it captures the proportion of non-isomorphic graphs within the generated samples. We also report the **Fréchet ChemNet Distance (FCD)** metric, which embeds the generated samples and the test data using the ChemNet neural network and compares the resulting distributions using the Wasserstein-2 distance.

MOSES. We evaluate H-Glow on the MOSES (Polykovskiy et al., 2020) benchmark. We sample 25K samples and report the Validity, as well as the Uniqueness. We further report the **Novelty**, which measures how many of these samples are non-isomorphic to any graph in the training set. We further include the **Filters** metric, which indicates the percentage of generated samples that pass the filters applied when constructing the dataset. In addition, we report **Scaffold similarity (Scaf)**, which compares the frequencies of the Bemis-Murcko scaffolds in the generated set and a reference set, as well as the **Similarity to the nearest neighbor (SNN)**, which computes the average Tanimoto similarity between the molecular fingerprints of a generated set and the fingerprints of the molecules of a reference set

Guacamol. We evaluate H-Glow on the Guacamol (Brown et al., 2019) benchmark. We note that, as Guacamol includes molecules with charges, the generated graphs are converted to charged molecules based on the relaxed validity criterion used by Jo et al. (2022) before being translated to the corresponding SMILES strings.

We sample 10K samples following prior baselines and report the Validity, as well as the percentage of **Valid and Unique (V.U.)** samples, and **Valid, Unique, and Novel (V.U.N.)** samples. The GuacaMol benchmark computes two scores, the **FCD score** and the **KL divergences (KL)** between the distributions of a set of physicochemical descriptors in the training set and the generated set.

Synthetic graphs. We evaluate the generation performance of H-Glow using the synthetic benchmarks Planar (Martinkus et al., 2022) and Tree (Bergmeister et al., 2024). Planar graphs are connected planar graphs that can be drawn on a plane without any edges crossing, while tree graphs are connected acyclic graphs.

We follow prior work (Vignac et al., 2022; Qin et al., 2025) and report V.U.N, which is the percentage of valid, unique, and novel graphs, as well as the Ratio metric. The ratio is computed by first calculating the MMD distances between the training and test sets for the same graph statistics. The final Ratio metric is then computed by dividing the average MMD distance between the generated sets and test sets by the average MMD distance between the training and test sets.

B.2 Baselines.

We compare against representative discrete graph diffusion and flow-matching methods, including DiGress (Vignac et al., 2022), DISCO (Xu et al., 2024), COMETH (Siraudin et al., 2024), and DeFoG (Qin et al., 2025). We additionally use other iterative denoising-based methods whenever comparable results are available. We note that we re-evaluate the released checkpoints from Qin

Table 8: Overview of the training hyperparameters of H-Glow.

	QM9	GuacaMol	MOSES	Planar	Tree
Epochs	1000	250	300	100000	100000
Learning rate	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
AMSGrad	Yes	Yes	Yes	Yes	Yes
RRWP steps	12	20	20	32	128
Number of layers	9	12	12	10	10
Number of heads	8	8	8	8	8
Prior distribution	Marginal	Marginal	Marginal	Marginal	Marginal
Train Distortion	Identity	Polydec	Polydec	Identity	Polydec-2
Sample Distortion	Adaptive (Ours)	Adaptive (Ours)	Adaptive (Ours)	Adaptive (Ours)	Adaptive (Ours)

et al. (2025), where the performance is generally consistent with the paper's results, with minor variations attributed to training and sampling stochasticity.

B.3 Architecture and training settings

Graph-transformer backbone. Following DiGress (Vignac et al., 2022), we use a graph transformer that jointly processes node, edge, and graph-level features. Given a noisy graph G_t and time t, the network produces representations for every node and unordered node pair. We retain the multi-head attention backbone and introduce a decomposed output parameterization with separate predictions for hazard and conditional jumps. We augment the input with Relative Random Walk Probabilities (RRWP) (Vignac et al., 2022; Qin et al., 2025).

Parametrization. H-Glow directly parameterizes the componentwise generators through separate hazard and jump prediction heads:

$$f_\theta(G_t, t) = \left(\boldsymbol{\tau}^X_{\theta,t}, \boldsymbol{\pi}^X_{\theta,t}, \boldsymbol{\tau}^E_{\theta,t}, \boldsymbol{\pi}^E_{\theta,t}\right),$$

where $\boldsymbol{\tau}^X_{\theta,t}$ and $\boldsymbol{\tau}^E_{\theta,t}$ are non-negative hazards, and $\boldsymbol{\pi}^X_{\theta,t}$ and $\boldsymbol{\pi}^E_{\theta,t}$ are the predicted jump distributions satisfying: $\sum_{\tilde{z} \neq z_t^{(c)}} \pi^{(c)}_{\theta,t}(\tilde{z} \mid z_t^{(c)}, G_t) = 1$, and $\pi^{(c)}_{\theta,t}(z_t^{(c)} \mid z_t^{(c)}, G_t) = 0$. H-Glow directly parameterizes the componentwise generators through separate hazard and conditional jump predictions. For a component c with transformer representation $h_t^{(c)}$, we first compute$u_t^{(c)} = \mathrm{ReLU}\left(W_h h_t^{(c)} + b_h\right)$. The network produces logits: $s^{(c)} = w_a^\top u_t^{(c)} + b_a, \ell^{(c)} = W_r u_t^{(c)} + b_r$. We parameterize the hazard as

$$\tau^{(c)}_{\theta,t} = a_t\, \sigma\left(s^{(c)}\right), \qquad a_t = \frac{\dot{\alpha}_t}{1 - \alpha_t},$$

where σ denotes the sigmoid function, ensuring $\tau^{(c)}_{\theta,t} \in [0, a_t]$.

We note that for the jump head, we mask the current category $i = z_t^{(c)}$ and apply softmax over the remaining categories:

$$\pi^{(c)}_\theta(j) = \begin{cases} \dfrac{\exp(\ell_j^{(c)})}{\sum_{k \neq i} \exp(\ell_k^{(c)})}, & j \neq i, \\ 0, & j = i. \end{cases}$$

We use separate node and edge output MLPs. Each MLP shares its hidden features between the hazard and jump projections. Edge logits are symmetrized across node pairs.

Settings. The shared componentwise heads, permutation-compatible structural features, and symmetric edge outputs preserve the permutation equivariance of the graph-transformer backbone. We provide an overview of the training hyperparameters of H-Glow in Table 8. We note that we use as prior distribution p_0 the empirical marginal prior over node and edge attributes (Vignac et al., 2022). We further use the train time distortion setting from Qin et al. (2025). We note that for sampling, we use H-Glow's inferred time discretization, as opposed to the sample time distortions used in Qin et al. (2025), see Section C.

C SAMPLING DETAILS

C.1 DERIVATION OF THE HAZARD-BASED TIME ALLOCATION

For readability, we initially suppress the dependence of componentwise generators on θ, t, and G_t. For each component c, we follow the uniformization construction in (Rao & Teg, 2013) and define:

$$\Lambda^{(c)} = \sum_{\tilde{z}} \max_{z \neq \tilde{z}} R^{(c)}(z, \tilde{z}). \tag{40}$$

This sums the largest incoming off-diagonal rate for each destination and bounds every row's exit hazard:

$$\sum_{\tilde{z} \neq z} R^{(c)}(z, \tilde{z}) \leq \Lambda^{(c)}, \quad \forall z. \tag{41}$$

For a component with only one possible attribute value, we set $\Lambda^{(c)} = 0$. For $\Lambda^{(c)} > 0$, we define

$$\mathbf{P}^{(c)} = \mathbf{I} + \frac{\mathbf{R}^{(c)}}{\Lambda^{(c)}}. \tag{42}$$

The bound above ensures that all entries of $\mathbf{P}^{(c)}$ are nonnegative. Its rows sum to one because the rows of $\mathbf{R}^{(c)}$ sum to zero. Thus, $\mathbf{P}^{(c)}$ is a stochastic matrix, with

$$P^{(c)}(z_t^{(c)}, z_t^{(c)}) = 1 - \frac{\tau^{(c)}}{\Lambda^{(c)}}, \qquad \tau^{(c)} = -R^{(c)}(z_t^{(c)}, z_t^{(c)}). \tag{43}$$

This diagonal entry is the probability of retaining the current attribute at one uniformization event for the fixed component matrix. If $\Lambda^{(c)} = 0$, then $\mathbf{R}^{(c)} = \mathbf{0}$, and we set $\mathbf{P}^{(c)} = \mathbf{I}$ and define the normalized hazard to be zero.

As a sampling policy, we choose the target time by assigning equal weight to the mean node and edge retention probabilities:

$$\widehat{t} = 1 - \frac{1}{2}\left(\frac{1}{N}\sum_{i=1}^{N} \frac{\tau_{\theta,t}^{(i)}}{\Lambda_{\theta,t}^{(i)}} + \frac{1}{\binom{N}{2}} \sum_{1 \leq i < j \leq N} \frac{\tau_{\theta,t}^{(ij)}}{\Lambda_{\theta,t}^{(ij)}}\right). \tag{44}$$

Since every normalized hazard lies in $[0, 1]$, this definition yields $\widehat{t} \in [0, 1]$.

Denoiser-based DFM generators. We first instantiate the policy under the base DFM generator construction (Gat et al., 2024). Let $a_t = \dot{\alpha}_t/(1 - \alpha_t)$, with $\alpha_t < 1$ and $\dot{\alpha}_t > 0$. We evaluate the denoiser at G_t and hold its predicted distribution fixed when constructing the component matrix:

$$R_{\theta,t}^{(c)}(z, \tilde{z}; G_t) = a_t \left[p_{1|t}^{\theta,(c)}(\tilde{z} \mid G_t) - \delta(z, \tilde{z})\right]. \tag{45}$$

For $z \neq \tilde{z}$, these entries are independent of the dummy source attribute z. Thus, for a component with at least two possible attribute values,

$$\begin{aligned} \Lambda_{\theta,t}^{(c)} &= \sum_{\tilde{z}} \max_{z \neq \tilde{z}} R_{\theta,t}^{(c)}(z, \tilde{z}; G_t) \\ &= a_t \sum_{\tilde{z}} p_{1|t}^{\theta,(c)}(\tilde{z} \mid G_t) = a_t. \end{aligned} \tag{46}$$

Components with only one possible attribute value have zero hazard and contribute zero. Substituting into Eq. (44) gives

$$\widehat{t} = 1 - \frac{1 - \alpha_t}{2\dot{\alpha}_t}\left(\frac{1}{N}\sum_{i=1}^{N} \tau_{\theta,t}^{X,(i)} + \frac{1}{\binom{N}{2}} \sum_{1 \leq i < j \leq N} \tau_{\theta,t}^{E,(ij)}\right). \tag{47}$$

Here, $\tau_{\theta,t}^{(c)}/a_t \in [0, 1]$, so $\widehat{t} \in [0, 1]$ without clipping.

Table 10: NFE across datasets and sampling steps.

Dataset	Steps T	NFE
QM9	20	19.9 ± 0.4
QM9	50	22.4 ± 2.3
MOSES	50	35.9 ± 7.0
MOSES	100	47.1 ± 15.7
MOSES	200	57.2 ± 14.2
GuacaMol	50	40.4 ± 8.1
GuacaMol	100	49.2 ± 13.7
GuacaMol	200	78.8 ± 35.6

C.2 Additional sampling details

In our experiments, we set the early stopping threshold to 1. Stopping when the predicted graph activity falls below this threshold improves generation quality compared with continuing to edit the graph until the full sampling budget T is exhausted.

D Additional experiments

D.1 Parameterization.

In the following, we conduct a controlled experiment on QM9 to compare the denoiser parametrization trained with cross-entropy against the hazard–jump parametrization trained with the decomposed objective in Eq. (22). Both models use identical data splits, a graph-transformer backbone, and matched initialization and optimization. We sample 10K samples using Euler and H-Glow. H-Glow sampling substantially improves the validity–FCD trade-off over Euler sampling, particularly at small sampling budgets. Combined with our decomposed parameterization, H-Glow sampling delivers consistently strong performance across the evaluated sampling budgets.

Table 9: Parameterization comparison.

		$T = 20$		$T = 50$		$T = 100$	
Model	Sampler	Val. ↑	FCD ↓	Val. ↑	FCD ↓	Val. ↑	FCD ↓
Denoiser	Euler	97.0	0.75	98.9	0.28	99.4	0.14
H-Glow	Euler	96.7	0.66	99.2	0.24	99.5	0.13
Denoiser	H-Glow	98.9	0.31	99.0	0.21	99.0	0.11
H-Glow	H-Glow	99.2	0.37	99.1	0.10	99.2	0.10

D.2 Clock comparisons

In Fig. 3, we compare different time distortions, including time distortions from prior work, such as polydec-2 and polydec-10 (Qin et al., 2025; Luo et al., 2026), as well as our hazard-guided inferred time. We omit the early stopping and hazard annealing here.

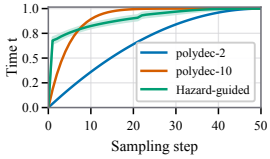


Figure 3: Schedulers.

D.3 Early stopping

As discussed in Section 4, H-Glow performs early stopping during inference to avoid disrupting node or edge updates that can render the graph invalid. We report the average number of function evaluations (NFE) in Table 10 for different benchmarks and budgets T.

D.4 Time annealing

To moderate large changes in the inferred time, we interpolate between the current model query time t_k and the hazard-guided target $\widehat{t}_k$:

$$t_{k+1} = (1 - \gamma_k)t_k + \gamma_k \widehat{t}_k, \qquad \gamma_k \in (0, 1],$$

where smaller γ_k can limit the magnitude of each time update, allowing further graph refinement. This is particularly useful during the initial sampling steps, when changes in the graph can produce

Table 11: Sampler comparison using the HGLow trained model on MOSES.

T	Sampler	Validity ↑	Unique ↑	Novelty ↑	Filters ↑	FCD ↓
50	RVF (default)	86.4	100.0	94.6	99.1	1.03
50	DeFoG (default)	80.3	100.0	95.3	98.0	1.38
50	HGLow (ours)	89.5	100.0	94.4	99.1	0.98
100	RVF (default)	87.7	100.0	94.2	99.2	1.08
100	DeFoG (default)	85.8	100.0	90.7	99.4	1.38
100	HGLow (ours)	89.9	99.9	93.0	99.2	0.96

Table 12: Large molecule generation performance on MOSES and GuacaMol.

			MOSES							GuacaMol				
Model	Class	Steps	Val. ↑	Unique. ↑	Novelty ↑	Filters ↑	FCD ↓	SNN ↑	Scaf. ↑	Val. ↑	V.U. ↑	V.U.N. ↑	KLD. ↑	FCD ↑
Training set	–	–	100.0	100.0	0.0	100.0	0.01	0.64	99.1	100.0	100.0	0.0	99.9	92.8
DiGress (Vignac et al., 2022)	Diff.	500	85.7	100.0	95.0	97.1	1.19	0.52	14.8	85.2	85.2	85.1	92.9	68.0
DisCo (Xu et al., 2024)	Diff.	500	88.3	100.0	97.7	95.6	1.44	0.50	15.1	86.6	86.6	86.5	92.6	59.7
Cometh (Siraudin et al., 2024)	Diff.	500	86.1	100.0	96.0	97.3	1.27	0.50	16.4	94.4	94.4	93.5	94.1	67.4
Cometh-PC (Siraudin et al., 2024)	Diff.	500	90.5	99.9	92.6	99.1	1.27	0.54	16.0	98.9	98.9	97.6	96.7	72.7
SimGFM (Luo et al., 2026)	Flow	50	86.1	100.0	95.1	98.6	1.44	0.51	16.9	98.5	98.5	98.3	93.6	77.1
SimGFM (Luo et al., 2026)	Flow	200	89.4	100.0	94.6	98.9	1.08	0.55	17.3	99.2	99.2	98.5	94.6	76.4
SimGFM (Luo et al., 2026)	Flow	500	91.1	100.0	93.9	99.2	1.19	0.56	14.4	99.2	99.2	98.6	95.5	73.5
DeFoG (Qin et al., 2025) ($\eta = 0, \omega = 0$)	Flow	50	88.7	100.0	95.9	98.0	3.49	0.53	11.8	84.2	84.2	83.8	91.0	32.7
DeFoG (Qin et al., 2025) ($\eta = 0, \omega = 0$)	Flow	100	89.5	100.0	95.6	98.2	2.51	0.54	12.6	92.3	92.3	91.8	92.9	51.4
DeFoG (Qin et al., 2025) ($\eta = 0, \omega = 0$)	Flow	200	89.6	100.0	94.7	98.6	1.93	0.55	12.8	94.6	94.6	94.1	93.8	62.7
DeFoG (Qin et al., 2025) ($\eta = 0, \omega = 0$)	Flow	500	89.4	100.0	94.9	98.6	1.66	0.55	13.3	96.1	96.1	95.5	94.1	71.9
DeFoG (Qin et al., 2025)	Flow	50	82.9	99.9	95.1	97.8	1.79	0.53	19.8	91.7	91.7	91.2	92.3	57.9
DeFoG (Qin et al., 2025)	Flow	100	88.5	99.9	90.7	99.2	1.75	0.58	16.7	96.7	96.7	95.7	96.0	74.2
DeFoG (Qin et al., 2025)	Flow	200	90.9	99.9	89.7	99.3	1.67	0.58	11.9	98.0	98.0	96.9	98.3	70.1
DeFoG (Qin et al., 2025)	Flow	500	91.5	99.9	90.0	99.2	1.47	0.55	11.5	99.0	99.0	97.9	97.7	73.8
H-Glow-DeFoG(Ours)	Flow	50	86.2	100.0	92.7	97.4	1.84	0.56	13.9	92.5	92.5	91.2	92.6	70.4
H-Glow-DeFoG (Ours)	Flow	100	91.1	99.9	92.0	99.2	1.46	0.57	16.7	97.9	97.9	96.3	97.0	72.4
H-Glow-DeFoG (Ours)	Flow	200	91.1	100.0	93.0	99.2	1.32	0.56	14.6	99.1	99.1	97.5	97.3	74.1
H-Glow-DeFoG (Ours)	Flow	500	91.0	100.0	92.0	99.3	1.37	0.57	15.7	99.3	99.3	98.0	98.0	74.2
H-Glow (Ours)	Flow	50	89.5	100.0	94.4	99.1	0.98	0.55	18.8	95.2	95.2	93.6	89.9	69.3
H-Glow (Ours)	Flow	100	89.9	99.9	93.0	99.2	0.96	0.56	16.0	99.2	99.2	98.1	96.9	77.7
H-Glow (Ours)	Flow	200	91.9	99.9	93.1	99.4	1.08	0.58	15.5	99.4	99.4	97.9	96.8	77.3

substantial fluctuations in the predicted activity. We use a piecewise-constant schedule,

$$\gamma_k = \begin{cases} \gamma_0, & k < K_{\text{prefix}}, \\ 1, & k \geq K_{\text{prefix}}, \end{cases}$$

which damps the initial time updates and subsequently adopts the inferred target directly. This interpolation preserves the possibility of backward time updates, with renoising applied whenever $t_{k+1} < t_k$. We use $\gamma_0 = 0.1$. For QM9, Moses, and GuacaMol, we set K_{prefix} to 20% of the total steps T. For Planar and Tree, K_{prefix} is 30% of T.

D.5 Sampler comparison

We provide the full quantitative comparison of the samplers comparisons under the same trained H-Glow model in Table 11.

D.6 Accelerating pretrained models

In Table 12, we provide the full quantitative comparison of DeFoG (base), without added stochasticity and target guidance during inference, DeFoG, using the sampling hyperparameters as reported in Qin et al. (2025), H-Glow-DeFoG, which applies the H-Glow hazard-guided sampler to accelerate the pretrained base DeFoG model. We further include our H-Glow DFM model.

D.7 Synthetic datasets

We summarize the parallel H-Glow sampler in Algorithm 3. For planar and tree, we set B to 1024 $T = 20$, and use $B = 512$ for $T = 50$, and $B = 128$ for $T = 100$ and $T = 200$. We compare H-Glow to autoregressive, diffusion, and flow-based approaches in Table 13.

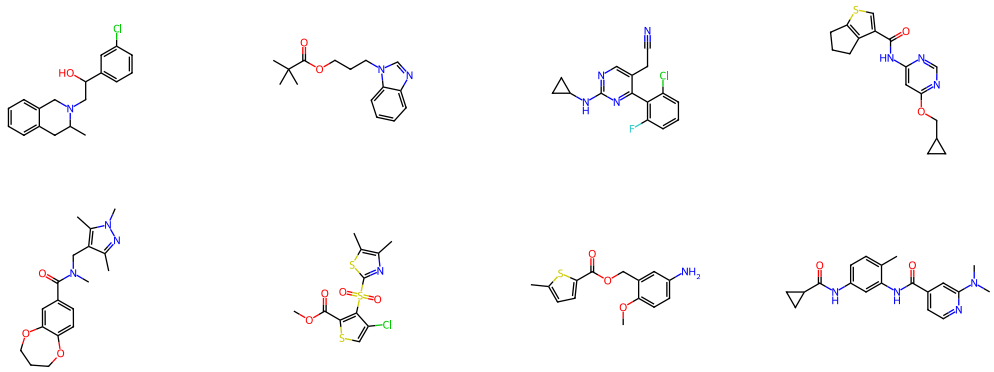
Table 13: Performance on the synthetic datasets Planar and Tree. AR denotes Autoregressive approaches.

Model	Class	Steps	Planar V.U.N. ↑	Planar Ratio ↓	Tree V.U.N. ↑	Tree Ratio ↓
Training set	–	–	100.0	1.0	100.0	1.0
GraphRNN (You et al., 2018)	AR	–	0.0	490.2	0.0	607.0
GRAN (Liao et al., 2019)	AR	–	0.0	2.0	0.0	607.0
BiGG (Dai et al., 2020)	AR	–	5.0	16.0	75.0	5.2
GraphGen (Goyal et al., 2020)	AR	–	7.5	210.3	95.0	33.2
AUTOGRAPH (Chen et al., 2026)	AR	–	87.5	1.5	–	–
EDGE (Chen et al., 2023)	Diff.	1000	0.0	431.4	0.0	850.7
DiGress (Vignac et al., 2022)	Diff.	1000	77.5	5.1	90.0	1.6
HSpectre (Bergmeister et al., 2024)	Diff.	–	95.0	2.1	100.0	4.0
GruM (Jo et al., 2023)	Diff.	–	90.0	1.8	–	–
DisCo (Xu et al., 2024)	Diff.	500	83.6	–	–	–
Cometh (Siraudin et al., 2024)	Diff.	500	92.5	–	–	–
Cometh-PC (Siraudin et al., 2024)	Diff.	–	99.5	–	–	–
CatFlow (Eijkelboom et al., 2024)	Flow	–	80.0	–	–	–
SimGFM (Luo et al., 2026)	Flow	20	94.0 ± 4.4	2.3 ± 0.6	88.0 ± 4.8	2.5 ± 0.9
SimGFM (Luo et al., 2026)	Flow	50	99.5 ± 1.0	1.8 ± 0.5	97.0 ± 1.0	2.0 ± 0.7
SimGFM (Luo et al., 2026)	Flow	200	100.0 ± 0.0	9.3 ± 2.6	99.5 ± 1.0	1.5 ± 0.2
SimGFM (Luo et al., 2026)	Flow	1000	100.0 ± 0.0	21.0 ± 1.5	100.0 ± 0.0	2.1 ± 0.8
DeFoG (Qin et al., 2025)	Flow	50	95.0 ± 3.2	3.2 ± 1.1	73.5 ± 9.0	2.5 ± 1.0
DeFoG (Qin et al., 2025)	Flow	1000	99.5 ± 1.0	1.6 ± 0.4	96.5 ± 2.6	1.6 ± 0.4
H-Glow (Ours)	Flow	20	95.0 ± 3.5	1.9 ± 0.5	93.5 ± 4.6	2.2 ± 0.8
H-Glow (Ours)	Flow	50	99.0 ± 1.2	1.9 ± 0.4	98.0 ± 2.9	2.7 ± 0.9
H-Glow (Ours)	Flow	100	99.0 ± 1.2	1.8 ± 0.3	98.5 ± 2.0	1.8 ± 0.4
H-Glow (Ours)	Flow	200	100.0 ± 0.0	2.4 ± 0.2	100.0 ± 0.0	2.0 ± 0.8

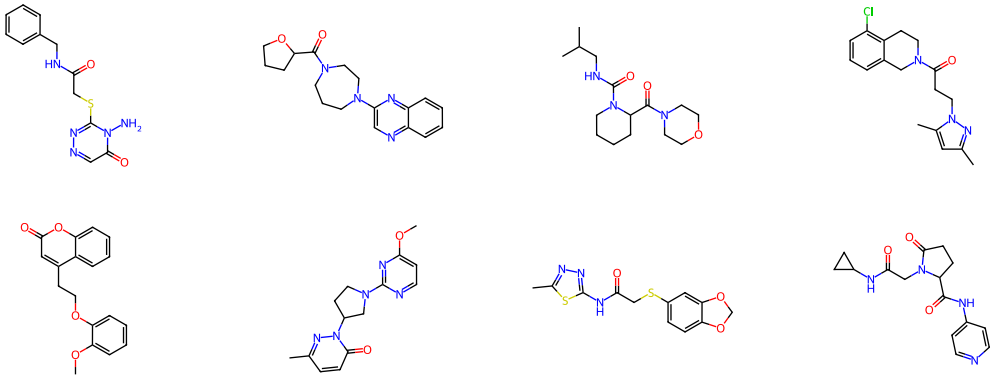
Algorithm 3 H-Glow: Parallel hazard-guided sampling

Input: model f_θ, budget T, maximum parallel updates B, threshold $T_{\text{stop}} > 0$
Initialize $G \sim p_0$, $t \leftarrow 0$, $k \leftarrow 0$
while $k < T$ **do**
  Construct rates $\mathrm{R}^{(c)}_{\theta,t}$ from (f_θ, G, t) and obtain $\{(\tau^c_\theta, \pi^c)\}_{c \in \mathcal{C}}$
  $H \leftarrow \sum_{c \in \mathcal{C}} \tau^c_\theta$
  if $H < T_{\text{stop}}$ **then return** G
  $\mathcal{C}_+ \leftarrow \{c \in \mathcal{C} : \tau^c_\theta > 0\}$, $m \leftarrow \min(B, |\mathcal{C}_+|)$
  Select $S \subseteq \mathcal{C}_+$, $|S| = m$, without replacement with weights $\{\tau^c_\theta\}_{c \in \mathcal{C}_+}$
  Sample $\widetilde{z}^c \sim \pi^c_\theta$ for all $c \in S$
  $G \leftarrow \mathrm{Update}(G, S, \{\widetilde{z}^c\}_{c \in S})$
  $t \leftarrow 1 - \frac{1-\alpha_t}{2\dot{\alpha}_t}\left(\overline{\tau}^X_{\theta,t} + \overline{\tau}^E_{\theta,t}\right)$
  $k \leftarrow k + 1$
end while
return G

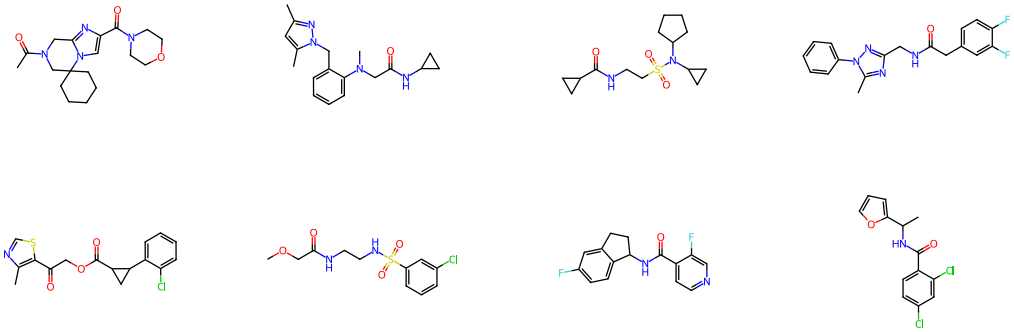
E QUALITATIVE RESULTS



(a) MOSES, $T = 50$.

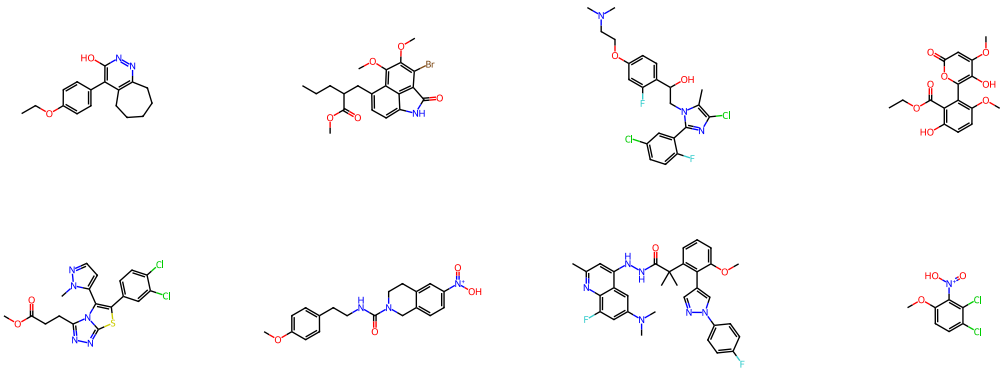


(b) MOSES, $T = 100$.

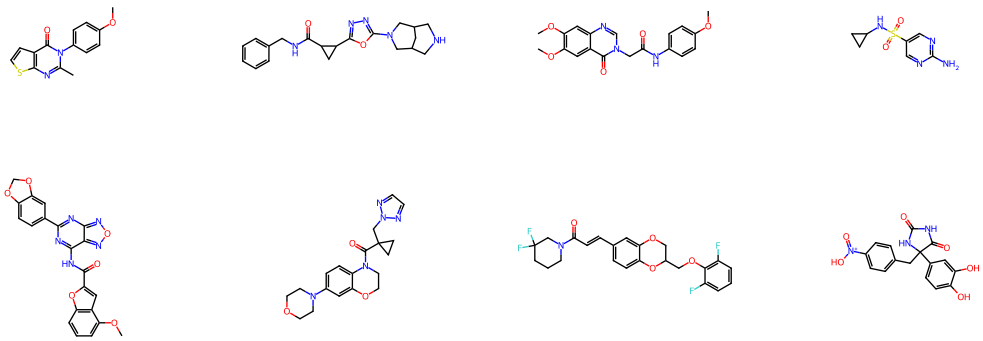


(c) MOSES, $T = 200$.

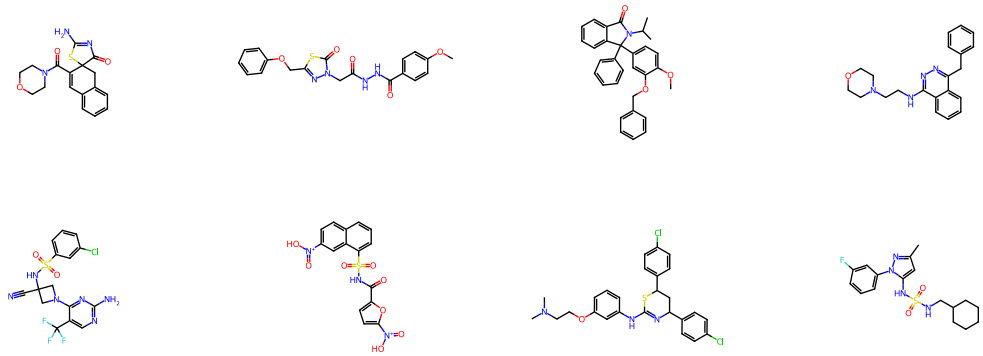
Figure 4: Uncurated sets of molecular graphs generated by H-Glow for the MOSES dataset for $T \in \{50, 100, 200\}$ steps.



(a) guacamol, $T = 50$.

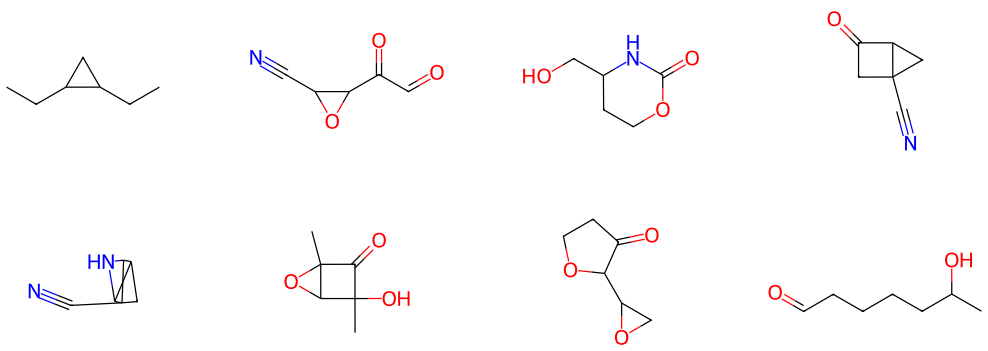


(b) guacamol, $T = 100$.

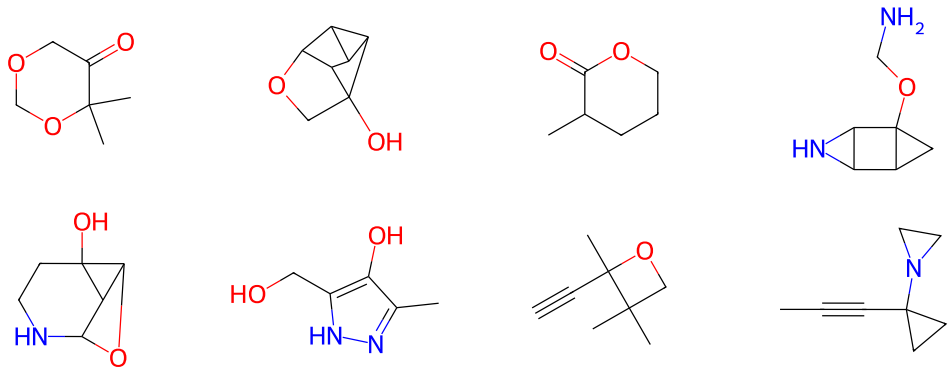


(c) guacamol, $T = 200$.

Figure 5: Uncurated sets of molecular graphs generated by H-Glow for the Guacamol dataset for $T \in \{50, 100, 200\}$ steps.

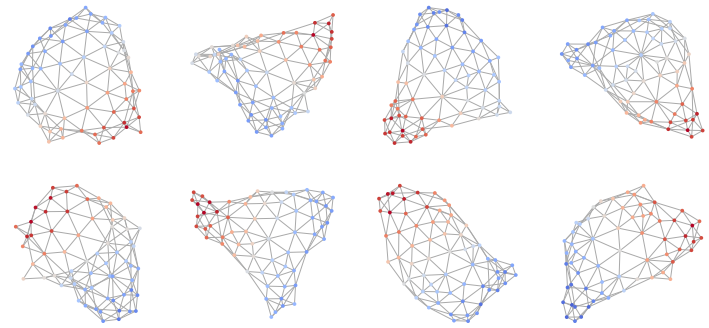


(a) QM9, $T = 20$.

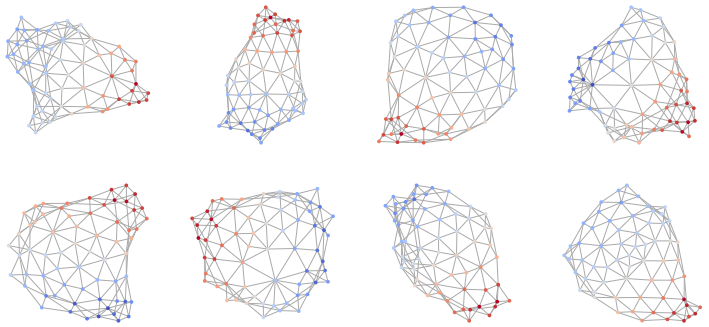


(b) QM9, $T = 50$.

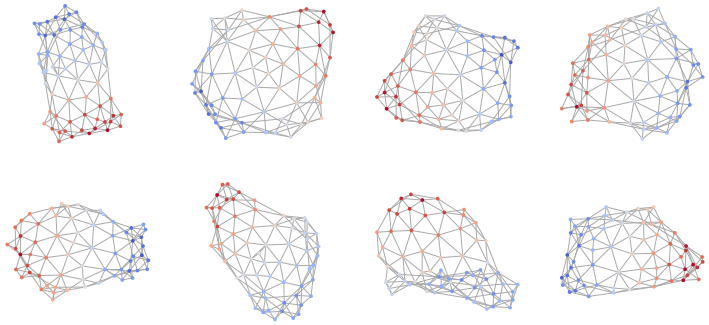
Figure 6: Uncurated sets of molecular graphs generated by H-Glow for the QM9 dataset for $T \in \{20, 50\}$ steps.



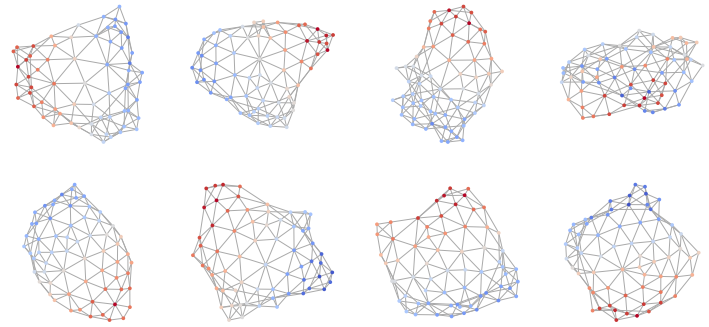
(a) Planar, $T = 20$.



(b) Planar, $T = 50$.



(c) Planar, $T = 100$.



(d) Planar, $T = 200$.

Figure 7: Uncurated sets of molecular graphs generated by H-Glow for the Planar dataset for $T \in \{20, 50, 100, 200\}$ steps.