SIMGFM: SIMPLIFYING DISCRETE FLOW MATCHING FOR GRAPH GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete Flow Matching (DFM) presents a promising approach for graph generation; however, existing adaptations often introduce substantial complexity by incorporating task-specific heuristics, compromising the continuity equation and significantly expanding the hyperparameter space. Moreover, their sampling efficiency remains limited, as the required number of steps is often comparable to diffusion models, diminishing DFM's practical advantages. To address these limitations, we propose SimGFM, a simplified graph DFM for graph generation. SimGFM introduces a graph-structured rate formulation based on minimalist design principles—characterized by a clear mathematical expression, free of ad-hoc heuristics, and consistent with the continuity equation. SimGFM achieves strong empirical results: on QM9, it matches prior models requiring 500–1000 steps with only 10 steps, and on most datasets, its performance at 50 steps matches or surpasses these baselines, demonstrating both efficiency and competitiveness.

1 INTRODUCTION

Graph generation is fundamental across domains from molecular chemistry to social networks, as graphs compactly represent complex relations and generate realistic structured data. Recent advances include continuous-time discrete diffusion frameworks (Xu et al., 2024; Siraudin et al., 2024) and discrete-flow frameworks (QIN et al., 2025; Campbell et al., 2024; Gat et al., 2024).

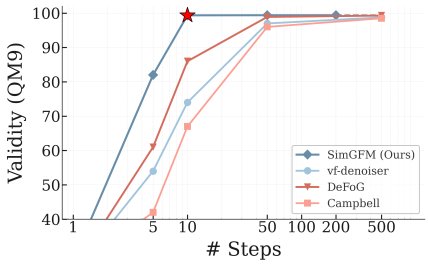


Figure 1: Validity on QM9 vs. sampling steps. Campbell (red) requires many steps, while vf-denoisers (blue) achieve higher validity with fewer steps. SimGFM further improves efficiency, reaching over 99% validity in 10 steps.

Diffusion models (Ho et al., 2020; Nichol & Dhariwal, 2021; Vignac et al., 2022) tightly couple training and sampling: once components such as the noise schedule or rate matrix are modified (Nichol & Dhariwal, 2021; Karras et al., 2022; Xu et al., 2024; Siraudin et al., 2024), retraining is typically required, incurring substantial computational cost. By contrast, discrete-flow models (Campbell et al., 2024; Gat et al., 2024) decouple training from sampling, allowing sampling adaptations without retraining and thus greater flexibility for diverse data distributions. In CV/NLP, flow matching has markedly accelerated sampling, in some cases enabling near one-step generation (Song et al., 2023; Liu et al., 2022; Lee et al., 2024; Geng et al., 2025). However, in graph generation, existing discrete-flow models remain computationally costly and require nearly as many steps as diffusion-based approaches, leaving the potential sampling efficiency of flow matching largely unrealized (QIN et al., 2025; Hou et al., 2025).

As shown in Fig. 1, Campbell et al. (2024) derive a closed-form rate matrix (Eq. 5) from the posterior endpoint $p_{1|t}(\cdot \mid X_t)$, but its posterior expectations and combinatorial bookkeeping are costly for graphs. Building on this, recent SOTA model (QIN et al., 2025) augments the Campbell field with heuristic velocity terms to gain accuracy, at the cost of (i) potential violations of the continuity equation, and (ii) added methodological complexity. By contrast, the vf-denoiser (Gat et al., 2024) offers a concise scheduler-based formulation (Eq. 6), avoids posterior expectations, and shows strong few-step performance, making it a simpler and more effective backbone for graph DFM.

Motivated by these observations, we propose SimGFM, a vf-denoiser-based method that strictly adheres to the standard DFM formulation without auxiliary modules. In particular, while the vf-denoiser is simple and flexible, its deterministic updates in complex graph tasks often collapse into template-like trajectories, limiting diversity. To address this, we introduce the *rvf-denoiser* (Eq. 11), a sampling-based variant that selects a single candidate outcome at each step, breaking symmetry and enhancing diversity while preserving theoretical consistency. On QM9 (Wu et al., 2018), SimGFM achieves 99.4% validity in just 10 steps, and across most datasets, it reaches or approaches SOTA performance with 10–50 steps, representing an *order-of-magnitude reduction* compared with diffusion/flow baselines (typically 500–1000), while also lowering hyperparameter tuning burden.

2 PRELIMINARIES

2.1 DISCRETE FLOW MATCHING

In this section, we introduce the core concepts of Discrete Flow Matching (DFM) (Campbell et al., 2024; Gat et al., 2024). Unlike diffusion models, which learn a data distribution via iterative noising and denoising, the goal of DFM is to learn a deterministic *probability path* p_t from a simple source distribution p_0 (e.g., a sequence composed of a "mask" symbol) to a target data distribution p_1. The core of the model is to train a neural network to predict the *velocity field* u_t of this probability path, which guides how samples evolve with time $t \in [0, 1]$ from the source to the target.

To build this framework, we first define a **conditional probability path** from a specific source sample $x_0 \sim p_0$ to a specific target sample $x_1 \sim p_1$. A simple and effective choice is their convex combination:

$$p_t\big(x^i \mid x_0, x_1\big) = (1 - \kappa_t)\ \delta_{x_0^i}\big(x^i\big) + \kappa_t\, \delta_{x_1^i}\big(x^i\big)\,, \tag{1}$$

where x^i is the i-th element of the sequence, δ is the Dirac delta (point mass), and κ_t is a schedule increasing monotonically from $\kappa_0 = 0$ to $\kappa_1 = 1$. This formula states that at time $t = 0$, the sample coincides with the source x_0, and at $t = 1$ it fully transforms into the target x_1.

To simulate generation along the prescribed path $p_t(x)$ for $t \in [0, 1]$, DFM adopts the **continuous-time Markov chain** (CTMC) paradigm: the sample X_t makes jumps over a state space $\mathcal{D}$ as time t evolves continuously on $[0, 1]$. DFM focuses on a model that predicts the **rate of change of probabilities** for each coordinate (token) of the current sample X_t with N tokens. Thus, for a sample $X_t \sim p_t$, each token updates independently as

$$X_{t+h}^i \ \sim\ \delta_{X_t^i}(\cdot)\ +\ h\, u_t^i(\cdot, X_t), \tag{2}$$

where $\delta_{X_t^i}$ denotes a Dirac mass at the current value and u_t^i is the probability velocity field for the i-th coordinate. If the probabilistic velocity u_t *generates* the probability path p_t, it means that for all $t \in [0, 1)$ and any sample $x_t \sim p_t$, updating each position i using the rule above equation 2 yields $x_{t+h} \sim p_{t+h} + o(h)$.

Moreover, the velocity u_t should satisfy the following **rate conditions**:

$$\sum_{x^i \in [K]} u_t^i(x^i, z) = 0, \qquad u_t^i(x^i, z) \geq 0 \quad \forall\, i \in [D],\ x^i \neq z^i. \tag{3}$$

Furthermore, prior work (Campbell et al., 2024; Gat et al., 2024) shows that a **continuity equation** (also called the Kolmogorov forward equation) holds in discrete flow matching, describing the time derivative of the state-marginal probability $\dot{p}_t(x)$, $x \in \mathcal{S}$:

$$\dot{p}_t(x)\ +\ \mathrm{div}_x\big(p_t\, u_t\big)\ =\ 0, \tag{4}$$

where $\mathrm{div}_x\big(p_t\, u_t\big) = \sum_{z \in \mathcal{S}} \sum_{i=1}^{D} \delta_x\big(\bar{z^i}\big) \Big[\, p_t(x)\, u_t^i(x^i, x)\ -\ p_t(z)\, u_t^i(x^i, z) \,\Big]$, measures the total outflow (probability flow $x \to z$) minus total inflow ($z \to x$) at state $x \in \mathcal{S}$, and $\delta_x\big(\bar{z^i}\big) = \prod_{j \neq i} \delta_{x^j}(z^j)$ indicates that only pairs (x, z) agreeing on all coordinates except possibly the i-th are considered when computing the flow. Intuitively, the continuity equation expresses that the rate of change of probability mass at x equals the net effect of the probability flow $p_t u_t$ at x. It has been shown that if the continuity equation holds, then u_t can generate the probability path p_t.

The choice of u_t is crucial. Two commonly used constructions for the rate matrix are:

(1) Campbell's construction. Campbell et al. (2024) provide a closed-form solution for the rate matrix u_t:

$$u_t^*(x, z|z_1) = \frac{\mathrm{ReLU}\left[\partial_t p_{t|1}(x \mid z_1) - \partial_t p_{t|1}(z \mid z_1)\right]}{Z_t^{>0} p_{t|1}(z \mid z_1)}, x \neq z. \tag{5}$$

where $p_{t|1}(z \mid x)$ means the state z at time t given the state x at time 1 and $Z_t^{>0} = \left|\left\{z_t : p_{t|1}(z_t \mid z_1) > 0\right\}\right|$, the diagonal case is set by $u_t^*(x, x|z_1) = -\sum_{x \neq z} u_t^*(x, z|z_1)$. Finally, the rate matrix is obtained by taking the posterior expectation: $u_t(x, z) = \mathbb{E}_{p_{1|t}(z_1|z)}\left[u_t^*(x, z \mid z_1)\right]$.

(2) Vf-denoiser. Gat et al. (2024) propose the vf-denoiser:

$$u_t^i(x^i, z) = \frac{\dot{\kappa}_t}{1 - \kappa_t}\left[p_{1|t}(x^i \mid z) - \delta_{z^i}(x^i)\right], \tag{6}$$

where $p_{1|t}(x \mid z)$ means the state x at time 1 given the state z at time t and κ_t is a scheduler (a monotone time mapping) satisfying $\dot{\kappa}_t \geq 0$, $\kappa_0 = 0$, and $\kappa_1 = 1$.

Both constructions depend on the prior $p_{1|t}(\cdot \mid z_t)$, which is typically estimated by a trained model; we denote the model output by $p_{1|t}^\theta(\cdot \mid z_t)$. The training objective is

$$\mathcal{L}(\theta) = -\sum_{i \in [N]} \mathbb{E}_{t,\,(X_0, X_1),\,X_t} \log p_{1|t}^\theta(X_1^i \mid X_t)\,. \tag{7}$$

2.2 Discrete Flow Matching on Graphs

Applying the Discrete Flow Matching (DFM) framework to graph generation requires accounting for the unique structure of graphs—namely, sets of nodes and edges. We represent a graph with N nodes as $G = (X, E)$, where $X = \{x^{(i)}\}_{i=1}^N$ is the set of node attributes and $E = \{e^{(ij)}\}_{1 \leq i < j \leq N}$ is the set of edge attributes. Based on Eq. 1, the probability path over graphs factorizes as

$$p_t(G_t \mid G_0, G_1) = \prod_{i=1}^N p_t\left(x_t^{(i)} \mid x_0^{(i)}, x_1^{(i)}\right) \prod_{1 \leq i < j \leq N} p_t\left(e_t^{(ij)} \mid e_0^{(ij)}, e_1^{(ij)}\right), \tag{8}$$

where $G_0 \sim p_0$ is a prior noise graph and $G_1 \sim p_1$ is a real data graph. Given this factorization, the sampling process for graphs follows the general update rule in Eq. 2: each node or edge is updated independently according to its velocity field,

$$G_{t+\Delta t}^{(k)} \sim \delta_{G_t^{(k)}}(\cdot) + \Delta t \cdot u_t^{(k)}(\cdot, G_t), \tag{9}$$

where k denotes either a node index (i) or an edge index (ij). Iterating this process from $t = 0$ to $t = 1$ yields a generated graph.

2.2.1 Existing Methods

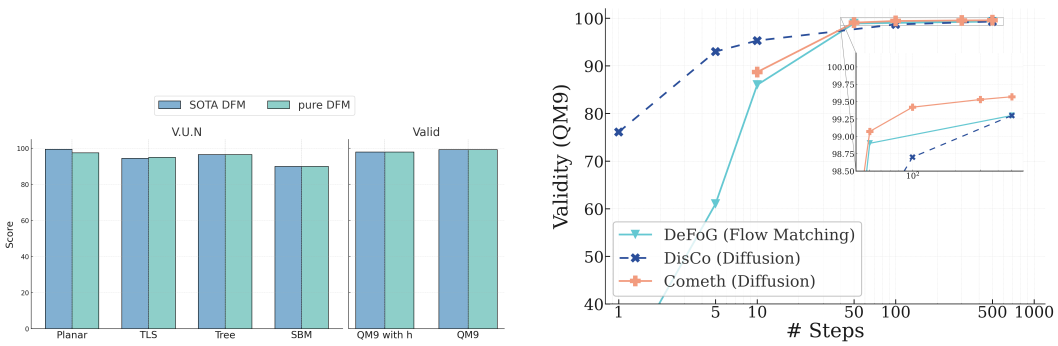
Due to the structural complexity of graphs, graph generation is inherently more challenging than image or text generation. Although DFM has solid theoretical foundations, directly applying it to complex graph structures often yields suboptimal results. Consequently, researchers have developed a range of auxiliary or heuristic techniques to improve performance.

Fine-tuning the model output. This line of work optimizes the predictor ϕ_θ to produce graphs with desired properties. For example, GGFlow (Hou et al., 2025) adopts a two-stage strategy: first pretraining with standard flow-matching loss to learn $p_\theta(G_1 \mid G_t)$, and then fine-tuning via reinforcement learning (RL). Reward functions tied to graph properties (e.g., docking scores, connectivity) guide RL, yielding an optimized policy $p_\theta^{\mathrm{RL}}(G_1 \mid G_t)$.

Modifying the velocity field. Another line directly alters the sampling dynamics. DeFoG (QIN et al., 2025), for instance, augments Campbell's base field (Eq. 5) with heuristic terms:

$$u_t(\cdot \mid G_1) = u_t^*(\cdot \mid G_1) + \omega\, u_t^\omega(\cdot \mid G_1) + \eta\, u_t^{\mathrm{DB}}(\cdot \mid G_1), \tag{10}$$

where u_t^* is the base velocity from Campbell's construction, u_t^ω is a target-guidance term weighted by ω, and u_t^{DB} is a stochastic exploration term weighted by η.



(a) Ablation Study

(b) Sampling Efficiency on QM9 dataset

Figure 2: Comparison of (a) ablation study and (b) sampling efficiency on the QM9 dataset.

2.2.2 Open Challenges in Graph DFM

Violations of the continuity equation. Directly fine-tuning the model output or modifying the velocity field (e.g., target-guidance heuristics) can break the core constraints required by Eq. 4, such as mass conservation and nonnegativity. In practice, these approaches often rely on auxiliary adjustments (e.g., normalization or clipping), which act as external interventions on the probability flow. While they may work empirically, such strategies lack a firm theoretical foundation and deviate from the standard DFM formulation.

Methodological complexity. Many enhancements to DFM introduce additional heuristics and design choices, which increase the overall modeling complexity and reduce reproducibility. These techniques expand the configuration space, making it harder to conduct systematic evaluation across tasks and datasets. As shown in Figure 2a, our experiments further indicate that, on some benchmarks, SOTA variant (QIN et al., 2025) do not consistently outperform pure baselines.

Sampling efficiency. In CV and NLP, flow models are valued for reducing the number of sampling steps compared to diffusion models. However, in graph generation, the steps required by current DFM methods remain comparable to those of diffusion approaches. This can be observed in Figure 2b, where existing graph DFM methods require nearly the same number of steps as diffusion-based models, suggesting that the efficiency advantage of DFM is not yet fully realized.

3 Proposed Framework

We propose **SimGFM**, a minimalist framework for graph DFM that adheres strictly to the standard formulation without introducing auxiliary modules, thereby preserving fidelity to flow-matching theory. The overall pipeline is illustrated in Figure 3.

3.1 Velocity Field of SimGFM

In the DFM framework, the choice of the velocity field u_t is central. Campbell's construction (Eq. 5), while theoretically sound, requires conditioning on fixed endpoints and averaging over posterior distributions, which incurs substantial computational overhead and hinders low-step generation. To alleviate this, we adopt the vf denoiser (Eq. 6) as our backbone, valued for its simplicity and scheduler-based flexibility.

However, in complex graph generation tasks such as MOSES (Polykovskiy et al., 2020) and TLS (Madeira et al., 2024), the deterministic vf-denoiser often drives the probability flow into premature, template-like trajectories, limiting diversity and causing mode collapse. Therefore, we introduce **rvf-denoiser (random vf-denoiser)**, a sampling-based variant of the vf-denoiser. Rather than using the full posterior distribution $p_{1|t}(\cdot \mid z)$, rvf-denoiser samples a single candidate ${x_{1|t}}^i \sim p_{1|t}(\cdot \mid z)$ and constructs the following velocity field:

$$u_t^{\mathbf{rvf},i}(x^i, z) = \frac{\dot{\kappa}_t}{1-\kappa_t}\Big[\delta_{{x_{1|t}}^i}(x^i) - \delta_{z^i}(x^i)\Big]. \tag{11}$$

Our update rule uniformly depends on $p_{1|t}(\cdot \mid G_t)$, which can be interpreted as the expectation over all possible terminal graphs G_1 conditioned on the current state G_t. In practice, this distribution

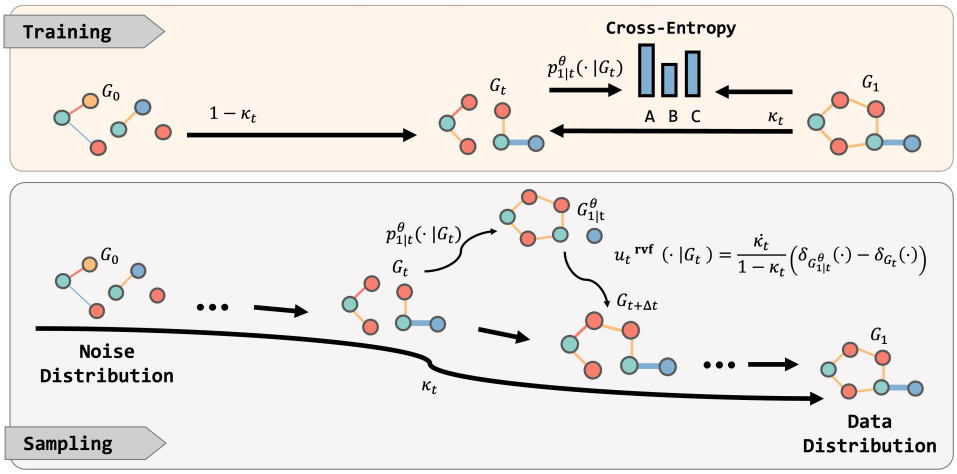


Figure 3: Our Proposed SimGFM Framework.

is approximated by the model, and we denote the resulting estimate as $p^\theta_{1|t}(\cdot \mid G_t)$. When the data contain two clearly different structures and G_t lies between them, purely following the average tends to trap the trajectory in an atypical in-between state (mean bias). To address this, we adopt a stochastic strategy that first samples according to $p_{1|t}$ and then updates (rvf-denoiser), enabling the trajectory to commit early to one alternative and break symmetry; once a preference forms, subsequent predictions reinforce that structure in a self-consistent manner. This strategy enhances exploration by avoiding convergence to ambiguous intermediate states and, within limited steps, improves both quality and diversity. When the data contain only a single dominant structure, the stochastic and deterministic updates are nearly aligned and incur negligible extra cost.

Proposition 1 (Unbiasedness of rvf-denoiser)**.** *The rvf-denoiser is an unbiased estimator of the vf-denoiser. Specifically, taking expectation over all possible candidate targets ${x_{1|t}}^i$, we have*

$$\mathbb{E}_{{x_{1|t}}^i|z}\left[u_t^{\mathbf{rvf},i}(x^i,z)\right] = u_t^i(x^i,z). \tag{12}$$

The proof is provided in Appendix A.1. It shows that rvf-denoiser behaves identically to vf-denoiser in expectation.

Corollary 1 (Consistency with DFM Updates)**.** *Due to its unbiasedness, the rvf-denoiser also satisfies the consistency requirement of DFM for one-step updates. Consequently, iterative sampling with rvf-denoiser simulates a distribution path that is consistent in expectation with the theoretical trajectory p_t, up to error $o(h)$.*

The proof is deferred to Appendix A.2. This corollary highlights that SimGFM is not a heuristic modification but a theoretically grounded alternative, fully consistent with the DFM framework in expectation. The added randomness preserves theoretical guarantees while offering practical benefits for sampling efficiency and quality.

3.2 Training and Sampling Procedures of SimGFM

Our framework follows the standard procedure of DFM (see Figure 3), but its core driving mechanism—the construction of the rate matrix—is redesigned to be more direct and efficient.

Training. We design the training procedure of **SimGFM** as shown in Algorithm 1. The entire objective centers on a single task: to teach a graph neural network f_θ to accurately predict the final, clean target graph G_1 from a halfway-evolved, ambiguous intermediate graph G_t. Each training iteration begins by sampling a real graph G_1 from the dataset and a time point t. An intermediate state G_t between pure noise and real data is then generated according to **Eq. 8**. Next, the noised graph G_t, together with the current time t, is fed into the network to produce a prediction of the posterior distribution over the original graph G_1. Finally, we optimize the model parameters by computing the cross-entropy between this predicted distribution and the ground-truth graph.

$$\mathcal{L}(\theta) = -\sum_{i\in[N]} \mathbb{E}_{t,\,(G_0,G_1),\,G_t} \log p^\theta_{1|t}\left(x_1^i \mid G_t\right) \; - \sum_{1\le i<j\le N} \mathbb{E}_{t,\,(G_0,G_1),\,G_t} \log p^\theta_{1|t}\left(e_1^{ij} \mid G_t\right) \tag{13}$$

Algorithm 1 SimGFM Training & Sampling

Input: Graph dataset $\mathcal{D} = \{G^1, \ldots, G^M\}$
while f_θ not converged **do**
  Sample $G_1 \sim \mathcal{D}$
  Sample $t \sim \mathcal{T}$
  Sample $G_0 \sim p_0(G_0)$
  Sample $G_t \sim p_t(G_t|G_0, G1)$ ▷ Noising
  $p^\theta_{1|t}(\cdot|G_t) \leftarrow f_\theta(G_t, t)$ ▷ Denoising
  loss $\leftarrow \mathrm{CE}_\lambda(G_1, p^\theta_{1|t}(\cdot|G_t))$
  optimizer.step(loss)
end while

Input: # graphs to sample S
for $i = 1$ to S **do**
  Sample N from train set ▷ # Nodes
  Sample $G_0 \sim p_0(G_0)$
  for $t = 0$ to $1 - \Delta t$ with step Δt **do**
    $p^\theta_{1|t}(\cdot|G_t) \leftarrow f_\theta(G_t, t)$ ▷ Denoising prediction
    $G^\theta_{1|t} \sim p^\theta_{1|t}(\cdot \mid G_t)$ ▷ Sample a potential graph
    $u_t(\cdot, G_t) \leftarrow \frac{\dot\kappa_t}{1-\kappa_t}\left[\delta_{G^\theta_{1|t}}(\cdot) - \delta_{G_t}(\cdot)\right]$
    $G_{t+\Delta t} \sim \delta_{G_t}(\cdot) + \Delta t \cdot u_t(\cdot, G_t)$ ▷ Update graph
  end for
  Store G_1
end for

Here, x_1^i denotes the attribute of the i-th node in G_1 (i.e., the target label), and e_1^{ij} denotes the attribute of the node pair (i, j) in G_1: the value 1 indicates that the edge is absent, while any other value represents the attribute of an existing edge.

Sampling. The sampling process of SimGFM, shown in Algorithm 1, realizes graph generation as a direct evolution from chaos to order. It starts from a noise graph G_0 sampled entirely from the prior distribution. The model then iteratively evolves from $t = 0$ to $t = 1$ through a sequence of discrete time steps Δt. At each step t, given input G_t, the model predicts the posterior distribution $p^\theta_{1|t}(\cdot \mid G_t)$ of the final target graph. Updates are performed according to Eq. 9.

Rather than averaging over the full distribution, we **sample a concrete candidate target graph** $G^\theta_{1|t}$, which provides a sharp provisional direction for the current state. The rvf-denoiser then constructs a rate matrix u_t that links G_t only to this candidate target. The graph is updated via the corresponding Markov jump process, yielding $G_{t+\Delta t}$. Repeating this predict–sample–update cycle gradually transforms pure noise into a structured graph at $t = 1$ that matches the target distribution.

3.3 Permutation Invariance Guarantees

Graph generative models should respect the permutation symmetries of graphs: both training and sampling must be independent of node indices. In our model, we ensure: (1) the loss is permutation-invariant; (2) the backbone denoiser is permutation-equivariant; (3) the one-step update kernels of both *vf*- and *rvf*-denoisers are permutation-equivariant; (4) consequently, the overall training objective and the sampling distribution are permutation-invariant. Full proofs are in the Appendix A.3.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate SimGFM across three task groups: (1) *generic graph generation* — Planar, SBM (Martinkus et al., 2022), Tree (Bergmeister et al., 2023), Ego-small, Community-small, Grid (Jo et al., 2022); (2) *molecular graph generation* — QM9 / QM9-with-H (Wu et al., 2018), MOSES (Polykovskiy et al., 2020); and (3) *conditional generation* — TLS (Madeira et al., 2024). Following prior work, we adopt the standard evaluation protocol for each dataset, reporting Valid/Unique/Novel (V.U.N.), Ratio, Fréchet ChemNet Distance (FCD), and graph statistics distances (Degree-MMD, Clustering-MMD, Orbit-MMD).

Baselines. We compare against major families of graph generative models. **Autoregressive models** include GraphRNN (You et al., 2018), GRAN (Liao et al., 2019), GraphGen (Goyal et al., 2020) BiGG (Dai et al., 2020), and AUTOGRAPH (Chen et al., 2025). **GAN models** cover GraphNVP (Madhawa et al., 2019) and SPECTRE (Martinkus et al., 2022). **Diffusion models** consist of DiGress (Vignac et al., 2022), GDSS (Jo et al., 2022), EDGE (Chen et al., 2023), BwR (Diamant et al., 2023), HSpectre (Bergmeister et al., 2023), GruM (Jo et al., 2023), DisCo (Xu et al., 2024), and Cometh (Siraudin et al., 2024). Finally, **Flow models** include DeFoG (QIN et al., 2025), Cat-

Table 1: Graph generation performance on the synthetic datasets: Planar, Tree and SBM. V.U.N. denotes Valid, Unique, and Novel, with Ratio closer to 1 indicating better alignment. Values are mean ± std from five runs of 40 graphs each. Best and second-best results are in bold and underline.

Model	Class	# Steps ↓	Planar		Tree		SBM	
			V.U.N. ↑	Ratio ↓	V.U.N. ↑	Ratio ↓	V.U.N. ↑	Ratio ↓
Train set	—	—	100	1.0	100	1.0	85.9	1.0
GraphRNN	Autoregressive	—	0.0	490.2	0.0	607.0	5.0	14.7
GRAN	Autoregressive	—	0.0	2.0	0.0	607.0	25.0	9.7
BiGG	Autoregressive	—	5.0	16.0	75.0	5.2	10.0	11.9
GraphGen	Autoregressive	—	7.5	210.3	95.0	33.2	5.0	48.8
AUTOGRAPH	Autoregressive	—	87.5	1.5	—	—	**92.5**	3.4
EDGE	Diffusion	1000	0.0	431.4	0.0	850.7	0.0	51.4
BwR (EDP-GNN)	Diffusion	1000	0.0	251.9	0.0	11.4	7.5	38.6
DiGress	Diffusion	1000	77.5	5.1	90.0	1.6	60.0	1.7
HSpectre	Diffusion	—	95.0	2.1	**100.0**	4.0	75.0	10.5
GruM	Diffusion	—	90.0	1.8	—	—	85.0	**1.1**
DisCo	Diffusion	500	83.6	—	—	—	66.2	—
Cometh	Diffusion	500	92.5	—	—	—	77.0	—
Cometh-PC	Diffusion	—	**99.5**	—	—	—	—	—
CatFlow	Flow	—	80.0	—	—	—	85.0	—
DeFoG (50 steps)	Flow	**50**	95.0	3.2	73.5	2.5	86.5	2.2
DeFoG (1000 steps)	Flow	1000	**99.5**	1.6	96.5	1.6	90.0	4.9
SimGFM (50 steps)	Flow	**50**	99.2±1.2	1.1±0.1	96.9±1.1	2.2±0.2	86.3±1.3	9.3±0.1
SimGFM (200 steps)	Flow	200	99.4±1.1	**1.0**±0.1	98.0±1.0	**1.3**±0.2	91.7±3.1	3.9±1.4

Table 2: Molecule generation on QM9. We present the results over five sampling runs of 10000 generated graphs each. We include the results of Relaxed Validity, which accounts for charged molecules, to facilitate comparison, as different methods may report varying types of validity.

Model	Without Explicit Hydrogenes					With Explicit Hydrogenes				
	# Steps ↓	Valid ↑	Relaxed Valid ↑	Unique ↑	FCD ↓	# Steps ↓	Valid ↑	Relaxed Valid ↑	Unique ↑	FCD ↓
Training set	—	99.3	99.5	99.2	0.03	—	97.8	98.9	99.9	0.01
SPECTRE	—	87.3	—	35.7	—	—	—	—	—	—
GraphNVP	—	83.1	—	99.2	—	—	—	—	—	—
GDSS	—	95.7	—	98.5	2.9	—	—	—	—	—
DiGress	—	99.0	—	96.2	—	—	95.4	—	**97.6**	—
GruM	—	99.2	—	96.7	0.11	—	—	—	—	—
CatFlow	—	—	**99.8**	**100.0**	0.44	—	—	—	—	—
DisCo	—	99.3	—	—	—	—	—	—	—	—
Cometh	—	**99.6**	—	96.8	0.25	—	—	—	—	—
GRAPHARM	—	90.25	—	95.62	1.22	—	—	—	—	—
DeFoG (50 steps)	50	98.9	99.2	96.2	0.26	50	97.1	98.1	94.8	0.31
DeFoG (500 steps)	500	99.3	99.4	96.3	0.12	500	98.0	98.8	96.7	**0.05**
SimGFM (10 steps)	**10**	99.4±0.1	99.5±0.1	96.2±0.1	0.35±0.1	—	—	—	—	—
SimGFM (50 steps)	50	99.4±0.1	99.5±0.0	95.8±0.0	0.15±0.0	**50**	97.8±0.2	98.6±0.1	97.1±0.1	**0.05**±0.0
SimGFM (200 steps)	200	99.4±0.0	99.4±0.0	96.3±0.0	**0.10**±0.0	200	**98.4**±0.1	**99.1**±0.0	96.8±0.1	**0.05**±0.0

Flow (Eijkelboom et al., 2024), and GGFlow (Hou et al., 2025). Baseline results are from official implementations or reported numbers in the corresponding papers; further details in Appendix B.

4.2 OVERALL PERFORMANCE

Requiring only **10–50 sampling steps**, SimGFM can match or even outperform state-of-the-art models across generic, molecular, and conditional graph generation tasks.

4.2.1 GENERIC GRAPH GENERATION

We evaluate SimGFM on the standard Planar, SBM, and Tree benchmarks. Table 1 reports two key metrics: (i) valid/unique/novel (V.U.N.) graphs and (ii) the Ratio of graph-statistic distances between generated and test sets relative to the train–test distance (lower is better). SimGFM demonstrates strong efficiency: on **Planar**, it achieves $\mathbf{100}\%$ V.U.N. with a Ratio of $\mathbf{1.1}$ using only $\mathbf{50}$ steps; on **Tree**, it reaches $\mathbf{98.0}\%$ V.U.N. and $\mathbf{1.3}$ Ratio at $\mathbf{200}$ steps; and on **SBM**, it matches the performance of DeFoG with $\mathbf{200}$ steps, compared to DeFoG's $\mathbf{1000}$. These results highlight that a minimalist, well-founded design can deliver both competitiveness and efficiency.

We further assess structural fidelity on Ego-small, Community-small, and Grid. Table 3 shows that SimGFM with $\mathbf{200}$ steps achieves consistently small deviations across degree, clustering, and orbit statistics, reaching or approaching the best overall scores among all compared methods.

Table 3: Generation results on the generic graph datasets. Results are the means of 3 different runs. The best results and the second-best results are marked **bold** and underline.

Model	# Steps↓	Ego-small				Community-small				Grid			
		Deg.↓	Clus.↓	Orbit↓	Avg.↓	Deg.↓	Clus.↓	Orbit↓	Avg.↓	Deg.↓	Clus.↓	Orbit↓	Avg.↓
Training Set	-	0.014	0.022	0.004	0.013	0.003	0.009	0.001	0.005	0.000	0.000	0.000	0.000
GraphRNN	-	0.090	0.220	0.003	0.104	0.080	0.120	0.040	0.080	0.064	0.043	0.021	0.043
EDP-GNN	1000	0.054	0.092	0.007	0.051	0.050	0.159	0.027	0.079	0.460	0.243	0.316	0.340
GDSS	1000	0.027	0.033	0.008	0.022	0.044	0.098	0.009	0.058	0.133	0.009	0.123	0.088
DiGress	500	0.028	0.046	0.008	0.027	0.032	0.047	0.009	0.025	0.037	0.046	0.069	0.051
GGFlow	500	0.005	0.033	0.004	0.014	**0.011**	0.030	**0.002**	**0.014**	0.030	**0.000**	0.016	0.015
DeFoG (50 steps)	**50**	0.034	0.012	0.067	0.039	0.029	0.157	0.052	0.079	0.004	**0.000**	**0.000**	0.001
DeFoG (200 steps)	200	0.056	0.149	0.068	0.091	0.022	0.040	**0.002**	0.022	0.001	**0.000**	**0.000**	**0.000**
SimGFM (50 steps)	**50**	**0.004**	0.024	0.006	0.011	0.038	0.081	0.008	0.043	**0.000**	**0.000**	**0.000**	**0.000**
SimGFM (200 steps)	200	0.006	**0.009**	**0.001**	**0.005**	0.031	**0.027**	**0.002**	0.020	**0.000**	**0.000**	**0.000**	**0.000**

Table 4: Large molecule generation performance. Only iterative denoising-based methods are reported here.

Model	MOSES						
	Val. ↑	Unique. ↑	Novelty ↑	Filters ↑	FCD ↓	SNN ↑	Scaf ↑
Training set	100.0	100.0	0.0	100.0	0.01	0.64	99.1
AUTOGRAPH	87.4	**100.0**	85.9	98.6	0.91	**0.55**	—
DiGress	85.7	**100.0**	95.0	97.1	1.19	0.52	14.8
DisCo	88.3	**100.0**	**97.7**	95.6	1.44	0.50	15.1
Cometh	90.5	99.9	92.6	**99.1**	1.27	0.54	16.0
DeFoG (50 steps)	83.9	99.9	96.9	96.5	1.87	0.50	**23.5**
DeFoG (500 steps)	**92.8**	99.9	92.1	98.9	1.95	**0.55**	14.4
SimGFM (50 steps)	88.6±0.1	**100.0**±0.1	94.7±0.1	—	**0.18**±0.1	—	—
SimGFM (200 steps)	89.4±0.0	99.8±0.1	92.9±0.1	—	**0.18**±0.0	—	—

Table 5: TLS conditional generation results.

Model	TLS Dataset	
	V.U.N. ↑	TLS Val. ↑
Train set	0.0	100
GraphGen	40.2	25.1
BiGG	0.6	16.7
SPECTRE	7.9	25.3
DiGress	13.2	12.6
ConStruct	**99.1**	92.1
DeFoG (50 steps)	44.5	93.0
DeFoG (1000 steps)	94.5	95.8
SimGFM (50 steps)	81.3	91.3
SimGFM (200 steps)	96.3	**96.3**

4.2.2 Molecular Graph Generation

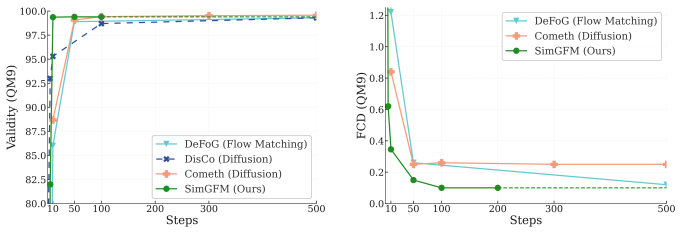
We further evaluate SimGFM on three molecular benchmarks. On **QM9**, Table 2 shows that SimGFM achieves SOTA performance at **200** steps, while already reaching $\mathbf{99.4}\%$ validity with only **10** steps, which is an order of magnitude fewer than the ~ 500 steps typically required by diffusion models, thereby demonstrating substantial gains in sampling efficiency. On **QM9-with-H**, results in Table 2 indicate that SimGFM at **200** steps matches or surpasses the best reported scores across all metrics, and at just **50** steps achieves a leading FCD of **0.05**. For the large-molecule dataset **MOSES**, Table 4 shows that SimGFM with **50** steps reduces FCD to **0.18**, the lowest among all compared methods, while maintaining strong validity and uniqueness.

4.2.3 Conditional Generation

We evaluate conditional generation on TLS dataset. Performance is assessed by (i) TLS Valid, measuring consistency between generated graphs and provided labels, and (ii) V.U.N. (validity, uniqueness, and novelty), where a graph is considered valid if it is both planar and connected. For fairness, we report the mean performance of existing methods on two subsets, as summarized in Table 5. SimGFM achieves $\mathbf{96.3}\%$ TLS Valid and $\mathbf{96.3}\%$ V.U.N. with only **200** steps, matching or surpassing DeFoG while requiring far fewer inference steps.

4.3 Sampling Efficiency

We report validity and FCD as functions of sampling steps on QM9 (Figs. 4a and 4b). SimGFM surpasses 0.99 validity with only **10** steps, whereas other methods typically require at least **50**. This advantage arises from the DFM mechanism: by following a straighter probability path, SimGFM reaches high validity with substantially fewer refinement steps.



(a) Valid (b) FCD

Figure 4: Sampling Efficiency on QM9

In terms of FCD (Fig. 4b), SimGFM decays rapidly from ~ 0.62 at 5 steps to $\mathbf{0.35}$ at 10, $\mathbf{0.15}$ at 50, and approaches $\mathbf{0.10}$ at 200, the lowest among all methods. Thus, $\mathbf{10}$ steps already attain performance once associated with $\mathbf{500 - 1000}$, while $\mathbf{200}$ steps achieve best-in-class results, underscoring a significant improvement in sampling efficiency.

4.4 Ablation Study

We study two training–sampling sensitive components: (i) the rate-matrix estimator (vf-denoiser vs. rvf-denoiser) and (ii) the DFM time scheduler κ_t.

Dataset	Vf-denoiser	Rvf-denoiser	Gain
TLS	2.50	96.25	+93.75
QM9-with-H	97.25	98.40	+1.15
MOSES	85.78	89.39	+3.61

Table 6: Rate-matrix ablation.

Table 6 summarizes the effect of replacing the vf-denoiser with the rvf-denoiser under 200 sampling steps. On TLS conditional generation, rvf-denoiser improves Valid by an absolute 93.75 points ($2.50 \rightarrow 96.25$), indicating a substantial robustness gain under conditional constraints. On MOSES, Valid rises from 85.78 to 89.39 ($+3.61$). Overall, rvf-denoiser outperforms vf-denoiser across benchmarks and is a stronger default choice.

We further analyze the time scheduler κ_t. Results across datasets show that stronger front loading (larger k) benefits small step budgets, while moderate front loading ($5 \leq k \leq 10$) is more effective for larger budgets. Detailed results are provided in Appendix C (Table 7).

5 Related Work

Autoregressive models on graphs (You et al., 2018; Liao et al., 2019) sequentially generate nodes and edges, offering high flexibility and allowing domain-specific constraints such as valence checks. Their key drawback lies in node ordering, which must be learned (Kong et al., 2023; Han et al., 2023) or predefined (You et al., 2018), leading to an enlarged search space and reduced efficiency.

Diffusion models (Ho et al., 2020; Song et al., 2020) treat generation as iterative denoising. Discrete variants like DiGress (Vignac et al., 2022) edit nodes and edges categorically while preserving marginals, achieving strong results on molecular and non-molecular datasets. Extensions such as EDGE (Chen et al., 2023), and DisCo (Xu et al., 2024) improve efficiency or structural modeling through mixture strategies, bandwidth constraints, or richer encodings. Continuous-time variants (Campbell et al., 2022; Xu et al., 2024) employ CTMCs; e.g., Cometh (Siraudin et al., 2024) integrates random-walk features to boost validity, uniqueness, and novelty. Despite these advances, diffusion remains hindered by slow sampling and error accumulation.

Flow Matching (FM) offers a more efficient refinement paradigm, transporting noise to data via ODEs or CTMCs with improved stability (Lipman et al., 2022; Liu et al., 2022) and demonstrated success in vision domains (Esser et al., 2024; Ma et al., 2024). Its discrete extension, **DFM** (Campbell et al., 2024; Gat et al., 2024), extends the framework to categorical data, including graphs, by employing linear interpolation and CTMC dynamics. Subsequent works such as CatFlow (Hou et al., 2025), DeFoG (QIN et al., 2025), and GGFlow (Hou et al., 2025) enhance performance but rely on costly optimization, heuristics, or reinforcement learning, complicating the framework.

6 Conclusion

We presented SimGFM, a minimal yet strong framework for discrete flow matching on graphs. Our approach employs a clean CTMC formulation, a simple monotone scheduler, and the unbiased rvf-denoiser, which together are sufficient to match or surpass more complex systems using only 10–50 steps. These results demonstrate that principled probabilistic design choices, free from ad-hoc heuristics, can substantially improve sampling efficiency while maintaining strong performance.

Limitations and impact. We have not fully explored the space of DFM schedulers, leaving room for improvement. As with all molecular generators, practitioners must ensure responsible downstream use; our focus is methodological efficiency, not property-targeted design.

REFERENCES

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. *arXiv preprint arXiv:2312.11529*, 2023.

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. *arXiv preprint arXiv:2502.02216*, 2025.

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.

Nathaniel Lee Diamant, Alex M Tseng, Kangway V Chuang, Tommaso Biancalani, and Gabriele Scalia. Improving graph generation by restricting graph bandwidth. In *International Conference on Machine Learning*, pp. 7939–7959. PMLR, 2023.

Floor Eijkelboom, Grigory Bartosh, Christian Andersson 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.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

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.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

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.

Xu Han, Xiaohui Chen, Francisco JR Ruiz, and Li-Ping Liu. Fitting autoregressive graph generative models through maximum likelihood estimation. *Journal of Machine Learning Research*, 24(97): 1–30, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Xiaoyang Hou, Tian Zhu, Milong Ren, Dongbo Bu, Xin Gao, Chunming Zhang, and Shiwei Sun. Improving graph generation with flow matching and optimal transport, 2025. URL https://openreview.net/forum?id=rMyfMS5nMt.

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.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B Aditya Prakash, and Chao Zhang. Autoregressive diffusion model for graph generation. In *International conference on machine learning*, pp. 17391–17408. PMLR, 2023.

Sangyun Lee, Zinan Lin, and Giulia Fanti. Improving the training of rectified flows. *Advances in neural information processing systems*, 37:63082–63109, 2024.

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.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pp. 23–40. Springer, 2024.

Manuel Madeira, Clement Vignac, Dorina Thanou, and Pascal Frossard. Generative modelling of structurally constrained graphs. *Advances in Neural Information Processing Systems*, 37:137218–137262, 2024.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

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.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KPRIwWhqAZ.

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

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

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.

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.

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.

A PROOF

A.1 PROOF OF UNBIASEDNESS OF RVF-DENOISER

Proposition 1 (Unbiasedness of rvf-denoiser)**.** *The rvf-denoiser is an unbiased estimator of the vf-denoiser. Specifically, taking expectation over all possible candidate targets $x^i_{1|t}$, we have*

$$\mathbb{E}_{x^i_{1|t}|z}\big[u_t^{\mathbf{rvf}}(x^i, z)\big] = u_t^i(x^i, z). \tag{14}$$

Proof. The derivation follows directly. Starting from the definition:

$$\mathbb{E}_{x^i_{1|t}|X_t}\big[u_t^{\mathbf{rvf}}\big] = \mathbb{E}_{x^i_{1|t}|X_t}\left[\frac{\dot\kappa_t}{1-\kappa_t}\Big(\delta_{x^i_{1|t}}(x^i) - \delta_{X_t^i}(x^i)\Big)\right]. \tag{15}$$

Moving constants outside of the expectation:

$$= \frac{\dot\kappa_t}{1-\kappa_t}\Big(\mathbb{E}_{x^i_{1|t}|X_t}[\delta_{x^i_{1|t}}(x^i)] - \delta_{X_t^i}(x^i)\Big). \tag{16}$$

By definition of expectation, $\mathbb{E}_{x^i_{1|t}|X_t}[\delta_{x^i_{1|t}}(x^i)]$ equals $p_{1|t}(x^i \mid X_t)$. Substituting, we obtain:

$$= \frac{\dot\kappa_t}{1-\kappa_t}\Big[p_{1|t}(x^i \mid X_t) - \delta_{X_t^i}(x^i)\Big] = u_t^i(x^i, X_t). \tag{17}$$

□

A.2 PROOF OF CONSISTENCY OF SIMFMG UPDATES WITH DFM

Corollary 1 (Consistency with DFM Updates)**.** *Due to its unbiasedness, the rvf-denoiser also satisfies the consistency requirement of DFM for one-step updates. Consequently, iterative sampling with rvf-denoiser simulates a distribution path that is consistent in expectation with the theoretical trajectory p_t, up to error $o(h)$.*

Proof. The validity of DFM relies on ensuring that each update approximately pushes the sample distribution from p_t to p_{t+h} in expectation. Itai et al. proved that vf-denoiser satisfies:

$$\mathbb{E}_{X_t}\left[\delta_{X_t}(x) \;+\; h\sum_{i=1}^{N}\delta_{X_t}(x^{\bar i})\,u_t^i(x^i, X_t)\right] \;=\; p_{t+h}(x) + o(h). \tag{18}$$

Applying the law of total expectation and the unbiasedness property, we obtain:

$$\mathbb{E}_{X_t, X_{1|t}}\left[\delta_{X_t}(x) \;+\; h\sum_{i=1}^{N}\delta_{X_t}(x^{\bar i})\,u_t^{\mathbf{rvf}}(x^i, X_t)\right] \tag{19}$$

$$= \mathbb{E}_{X_t}\left[\mathbb{E}_{X_{1|t}|X_t}\left[\delta_{X_t}(x) \;+\; h\sum_{i=1}^{N}\delta_{X_t}(x^{\bar i})\,u_t^{\mathbf{rvf}}(x^i, X_t)\right]\right] \tag{20}$$

$$= \mathbb{E}_{X_t}\left[\delta_{X_t}(x) \;+\; h\sum_{i=1}^{N}\delta_{X_t}(x^{\bar i})\,\underbrace{\mathbb{E}_{X_{1|t}{}^i|X_t}\big[u_t^{\mathbf{rvf}}(x^i, X_t)\big]}_{=\,u_t^i(x^i, X_t)}\right] \tag{21}$$

$$= p_{t+h}(x) + o(h). \tag{22}$$

□

A.3 Proof of Permutation Invariance for RVF-denoiser

A.3.1 Notation and Setup

We denote an undirected graph by $G = (x_{1:N}, e_{1\le i<j\le N})$, where node variables take values in $\mathcal{X}$ and edge variables in $\mathcal{E}$. For any node permutation with matrix $P \in \{0,1\}^{N\times N}$, define the relabeling action on graph-indexed tensors by

$$\pi_P(X) = PX, \qquad \pi_P(A) = PAP^\top, \qquad [\pi_P(E)]_{\{i,j\}} = E_{\{P^{-1}(i),P^{-1}(j)\}}.$$

Let δ_G denote the Dirac measure at G, and $(\pi_P)_\# \mu$ the pushforward of a measure μ via π_P. Scalars such as $t, \Delta t, \kappa_t, \dot\kappa_t$ are invariant under π_P. We write G_t for a noisy state, f_θ for the denoiser, and $p^\theta_{1|t}(\cdot \mid G_t)$ for the predicted clean-graph distribution.

A.3.2 Backbone Equivariance

Proposition 2. *The attention-based Graph Transformer denoiser satisfies*

$$f_\theta(\pi_P(G_t), t) = \pi_P\big(f_\theta(G_t, t)\big), \qquad p^\theta_{1|t}(\cdot \mid \pi_P(G_t)) = \pi_P\big(p^\theta_{1|t}(\cdot \mid G_t)\big).$$

Proof. With shared projections $Q = XW_Q$, $K = XW_K$, $V = XW_V$, relabeling yields $Q' = PQ$, $K' = PK$, $V' = PV$. Shared edge bias/mask obeys $B' = PBP^\top$, $M' = PMP^\top$. The score matrix satisfies $L' = \frac{Q'K'^\top}{\sqrt{d_k}} + B' + M' = PLP^\top$. Row-softmax commutes with row permutations, hence $\mathrm{Att}' = P\,\mathrm{Att}\,P^\top$. Aggregation gives $Y' = \mathrm{Att}'V' = P(\mathrm{Att}V) = PY$. Pointwise residuals, layer normalizations, and MLPs commute with P. Multi-head attention and stacking preserve equivariance. □

A.3.3 Loss Invariance

Proposition 3. *The training loss*

$$\mathcal{L}(\theta; G_t, G_1) = -\sum_{i\in[N]} \log p^\theta_{1|t}(x_1^i \mid G_t) \; - \sum_{1\le i<j\le N} \log p^\theta_{1|t}(e_1^{ij} \mid G_t)$$

is permutation-invariant:

$$\mathcal{L}(\theta; \pi_P(G_t), \pi_P(G_1)) = \mathcal{L}(\theta; G_t, G_1).$$

Proof. By backbone equivariance, $p^\theta_{1|t}(\cdot \mid \pi_P(G_t)) = \pi_P\big(p^\theta_{1|t}(\cdot \mid G_t)\big)$. The node sum reindexes via $i \mapsto P(i)$; the unordered edge sum reindexes via $\{i,j\} \mapsto \{P(i), P(j)\}$. Reindexing does not change the sums, proving invariance. □

A.3.4 One-step Kernel Equivariance

Define the vector fields and one-step kernels (with global scalars $\kappa_t, \dot\kappa_t, \Delta t$):

$$\widehat G \sim p^\theta_{1|t}(\cdot \mid G_t), \qquad u_t^{\mathrm{rvf}}(\cdot, G_t) = \frac{\dot\kappa_t}{1-\kappa_t}\big[\delta_{\widehat G}(\cdot) - \delta_{G_t}(\cdot)\big], \qquad K_t^{\mathrm{rvf}} = \delta_{G_t} + \Delta t\, u_t^{\mathrm{rvf}},$$

$$u_t^{\mathrm{vf}}(\cdot, G_t) = \frac{\dot\kappa_t}{1-\kappa_t}\Big[\mathbb{E}_{\widehat G\sim p^\theta_{1|t}(\cdot|G_t)}\delta_{\widehat G}(\cdot) - \delta_{G_t}(\cdot)\Big], \qquad K_t^{\mathrm{vf}} = \delta_{G_t} + \Delta t\, u_t^{\mathrm{vf}}.$$

Proposition 4. ***Vf-denoiser.*** *For any measurable set $\mathcal{S}$,*

$$K_t^{\mathrm{vf}}(\pi_P(G_t), \pi_P(\mathcal{S})) = K_t^{\mathrm{vf}}(G_t, \mathcal{S}).$$

Proof. From backbone equivariance, $\pi_P(\widehat G) \overset{d}{=} \widehat G' \sim p^\theta_{1|t}(\cdot \mid \pi_P(G_t))$. Pushforward gives $(\pi_P)_\#\mathbb{E}[\delta_{\widehat G}] = \mathbb{E}[\delta_{\pi_P(\widehat G)}]$ and $(\pi_P)_\#\delta_{G_t} = \delta_{\pi_P(G_t)}$, hence $K_t^{\mathrm{vf}}(\pi_P(G_t), \cdot) = (\pi_P)_\# K_t^{\mathrm{vf}}(G_t, \cdot)$. □

Proposition 5. ***Rvf-denoiser.*** *For any measurable set $\mathcal{S}$,*

$$K_t^{\mathrm{rvf}}(\pi_P(G_t), \pi_P(\mathcal{S})) = K_t^{\mathrm{rvf}}(G_t, \mathcal{S}) \quad \textit{in distribution.}$$

Proof. With $\pi_P(\widehat{G}) \overset{d}{=} \widehat{G}' \sim p^{\theta}_{1|t}(\cdot \mid \pi_P(G_t))$ and $(\pi_P)_{\#}\delta_{\widehat{G}} = \delta_{\pi_P(\widehat{G})}$, the claim follows immediately. □

A.3.5 Sampling Trajectory and Training Objective

Sampling invariance. If the initial distribution p_0 and the noising kernel $p_t(G_t \mid G_0, G_1)$ are compatible with permutations (relabeling only changes indices, not structural dependence), then kernel equivariance implies, by the Markov property and induction over time steps, that for any time grid and finite set of times,

$$\Pr\big(G_{t_1} \in \mathcal{S}_1, \ldots, G_{t_k} \in \mathcal{S}_k\big) = \Pr\big(\pi_P(G_{t_1}) \in \mathcal{S}_1, \ldots, \pi_P(G_{t_k}) \in \mathcal{S}_k\big),$$

so the terminal sampling distribution over isomorphism classes is permutation-invariant.

Training invariance. Taking expectation over $(t, (G_0, G_1), G_t)$ in the loss shows that the overall training objective is permutation-invariant; the expected gradient is unchanged under node relabeling.

B Experimental Details

B.1 Computing Environment

Our implementation is based on PyG (Fey & Lenssen, 2019). The experiments are conducted on a single workstation with 8 A100 GPUs.

B.2 Implementation Details

We adopt the Graph Transformer backbone from DiGress (Vignac et al., 2022), with further experimental details available in our source code at https://anonymous.4open.science/r/SimGFM-F9C5.

C Further Results

C.1 Scheduler Sensitivity

We adopt the power-accelerated family $\kappa_t = f_k(t) = 1 - (1-t)^k$ with $k \in \{1, 2, 5, 10, 20\}$, where larger k front loads progress. Table 7 reports Valid under three representative settings: QM9 with a small step budget (10 steps), MOSES with a large step budget (200 steps), and TLS conditional generation (200 steps). On QM9 (10 steps), Valid improves monotonically with k and peaks at $k = 20$, suggesting strong front loading is preferred when steps are scarce. On MOSES (200 steps), Valid peaks at $k = 10$ and remains close at $k = 20$, indicating that moderate front loading balances early progress and late refinement. On TLS (200 steps), the best results occur at $k = 2$ and $k = 20$, while $k = 10$ underperforms, reflecting task-dependent optima.

Table 7: Scheduler sensitivity on QM9 (10 steps), MOSES (200 steps), and TLS (200 steps).

Dataset	Steps	Valid ↑				
		$\kappa_t = t$	$\kappa_t = f_2(t)$	$\kappa_t = f_5(t)$	$\kappa_t = f_{10}(t)$	$\kappa_t = f_{20}(t)$
QM9	10	95.72	95.84	98.83	99.09	**99.37**
MOSES	200	81.92	87.52	88.24	**89.39**	89.29
TLS	200	93.75	**96.25**	95.00	93.75	**96.25**

D Further Discussion

D.1 Reproducibility Statement

We have taken several steps to ensure reproducibility. Source code, datasets, and detailed instructions are available at https://anonymous.4open.science/r/SimGFM-F9C5.

D.2 LLM Usage

We used large language models (LLMs) for language editing and polishing only.

D.3 Ethics Statement

Our study does not involve human subjects, sensitive personal data, or applications with foreseeable harmful impact. All datasets used are publicly available, and we follow community standards regarding data usage, fairness, and privacy.