Conditional Diffusion Based on Discrete Graph Structures for Molecular Graph Generation

Han Huang, Leilei Sun, Bowen Du[*], Weifeng Lv

State Key Laboratory of Software Development Environment, Beihang University, China
{h-huang, leileisun, dubowen, lwf}@buaa.edu.cn

Abstract

Learning the underlying distribution of molecular graphs and generating high-fidelity samples is a fundamental research problem in drug discovery and material science. However, accurately modeling distribution and rapidly generating novel molecular graphs remain crucial and challenging goals. To accomplish these goals, we propose a novel Conditional Diffusion model based on discrete Graph Structures (CDGS) for molecular graph generation. Specifically, we construct a forward graph diffusion process on both graph structures and inherent features through stochastic differential equations (SDE) and derive discrete graph structures as the condition for reverse generative processes. We present a specialized hybrid graph noise prediction model that extracts the global context and the local node-edge dependency from intermediate graph states. We further utilize ordinary differential equation (ODE) solvers for efficient graph sampling, based on the semi-linear structure of the probability flow ODE. We also combine the solvers with gradient guidance from the molecule property predictor for similarity-constrained molecule optimization. Experiments on diverse datasets validate the effectiveness of our framework. Particularly, the proposed method still generates high-quality molecular graphs in a limited number of steps.

Introduction

Dating back to the early works of Erdős Rényi random graphs (Erdős, Rényi et al. 1960), graph generation has been extensively studied for applications in biology, chemistry, and social science. Recent graph generative models make great progress in graph distribution learning by exploiting the capacity of neural networks. Models for molecular graph generation are notable for their success in representing molecule structures and restricting molecule search space, which facilitates drug discovery and material design. In terms of the sampling process of graph generative models, autoregressive generation constructs molecular graphs step-by-step with decision sequences (You et al. 2018a; Jin, Barzilay, and Jaakkola 2018; Shi et al. 2020; Luo, Yan, and Ji 2021), whereas one-shot generation builds all graph components at once (Zang and Wang 2020; Lippe and Gavves 2021; Liu et al. 2021). Recently, diffusion-based models have been applied effectively to one-shot molecular graph generation (Jo, Lee, and Hwang 2022), highlighting the advantages of flexible model architecture requirements and graph permutation-invariant distribution modeling.

However, current diffusion-based models for molecular graphs still suffer from generation quality and sampling speed issues. In the work of Jo, Lee, and Hwang, the generated graph distribution faces an obvious distance from the true distribution of datasets. Furthermore, their sampling process relies heavily on extra Langevin correction steps (Song et al. 2021) to diminish approximation errors, which largely increases computational cost and inference time, implying insufficient expressiveness of the graph score estimate model. We argue that two major factors hinder the practice of diffusion-based models for molecular graph generation. One is to focus on real-number graph formulation (*i.e.*, representing molecules as node feature and edge feature matrices) while neglecting the discrete graph structures, making it difficult to extract accurate local motifs from noisy real-number matrices for denoising and staying close to the true graph distribution. The other is that a straightforward graph neural network design may not be strong enough to fully model the node-edge dependency from corrupted graphs and further satisfy the complex generation requirements, such as local chemical valency constraints, atom type proportion closeness, and global structure pattern similarity.

To address these issues, we propose a novel Conditional Diffusion model based on discrete Graph Structures (CDGS) for molecular graph generation. We find that considering graph discreteness and designing suitable graph noise prediction models could boost the ability of diffusion models in the graph domain, allowing for faster sampling and downstream applications.

Graph discreteness. We develop a simple yet effective method for incorporating discrete graph structures without using special discrete state spaces. Along with variables for node and edge features, additional one-bit discrete variables are added to indicate the existence of edges. We convert them to real numbers and determine the quantization threshold. In our diffusion framework, the continuous forward process is applied directly to edge existence variables, but for the reverse process, discrete graph structures are decoded first and serve as the condition for each sampling step.

Graph noise prediction model. We design a hybrid graph

[*]Corresponding author

noise prediction model composed of standard message passing layers on discrete graphs and attention-based message passing layers on fully-connected graphs. The first concentrates on neighbor node-edge dependency modeling, and the second on global information extraction and transmission. Unlike (Jo, Lee, and Hwang 2022) which utilizes separate networks for node and edge denoising, we apply the unified graph noise prediction model to explicitly interact with the node and edge representations from both real-valued matrices and discrete graph structures.

Fast sampling and downstream applications. We employ stochastic differential equations (SDEs) to describe the graph diffusion process. With the simple Euler-Maruyama method, our diffusion-based model can obtain high-fidelity samples in 200 steps of network evaluations, much fewer steps than the previous method. We can benefit from recent research on probability flow ordinary differential equations (ODE) (Zhang and Chen 2022; Lu et al. 2022) to further promote fast graph sampling because we preserve the real-number graph description as an integral part of SDE. Therefore, we introduce fast ODE solvers utilizing the semi-linear structure of probability flow ODEs for graphs. Exploiting ODE solvers, we also construct a useful pipeline for similarity-constrained molecule optimization based on latent space determined by the parameterized ODE and gradient guidance from the graph property predictor.

Our main contributions are summarized as follows:

- We propose a novel conditional diffusion framework based on discrete graph structures. Leveraging a specialized graph noise prediction model, our framework accurately models the complex dependency between graph structures and features during the generative process.
- We promote high-quality rapid graph sampling by adapting ODE solvers that utilize the semi-linear structure of the probability flow ODE. These ODE solvers also serve as the foundation for our effective similarity-constrained molecule optimization pipeline.
- Experimental results demonstrate that our method outperforms the state-of-the-art baselines in both molecular graph and generic graph generation.

Methodology

Conditional Graph Diffusion

The first step in constructing diffusion probabilistic models (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020; Song et al. 2021; Kingma et al. 2021) is to define a forward process that perturbs data with a sequence of noise until the output distribution becomes a known prior distribution. Assuming a continuous random variable $\boldsymbol{x}_0 \in \mathbb{R}^d$ and a well-defined forward process $\{\boldsymbol{x}_t\}_{t\in[0,T]}$, we have

$$q_{0t}(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\alpha_t\boldsymbol{x}_0, \sigma_t^2\boldsymbol{I}) , \qquad (1)$$

where $\alpha_t, \sigma_t \in \mathbb{R}^+$ are time-dependant differentiable functions. α_t and σ_t are usually chosen to ensure that $q_T(\boldsymbol{x}_T) \approx \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ with the decreasing signal-to-noise ratio α_t^2/σ_t^2. By learning to reverse such a process, the diffusion model generates new samples from the prior distribution.

It is a simple way to apply diffusion models to the graph domain by formulating graphs as high-dimensional variables $\boldsymbol{G} \in \mathbb{R}^{N\times F} \times \mathbb{R}^{N\times N}$ composed of N node features with F dimensions and an edge type matrix (Jo, Lee, and Hwang 2022). We argue that overlooked discrete graph structures, including motifs like rings and stars, may provide extra clues for node-edge dependency modeling and graph denoising. We propose to separate the edge existence matrix from the edge type matrix and utilize a one-bit discrete variable representing the existence of a possible edge, forming $\bar{\boldsymbol{A}} \in \{0,1\}^{N\times N}$ for the whole graph. Instead of designing special discrete state spaces for discrete variables like (Hoogeboom et al. 2021; Austin et al. 2021), we turn bits into real numbers and determine a quantization threshold. Thus, we can conveniently apply the continuous diffusion process to these variables and decode them with quantization back to discrete graph structure $\bar{\boldsymbol{A}}_t$ for $t \in [0, T]$. The discrete graph structures can be plugged into the reverse process and function as conditions.

We redefine the graph $\boldsymbol{G}$ by real-number node features $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ and edge information $\boldsymbol{A} \in \mathbb{R}^{2\times N\times N}$ (one channel for edge existence which can be quantized to $\bar{\boldsymbol{A}}$ and the other for edge types). The forward diffusion process for graphs shown in Figure 1 can be described by the stochastic differential equation (SDE) sharing the same transition distribution in Eq. 1 (Kingma et al. 2021) with $t \in [0, T]$ as

$$\mathrm{d}\boldsymbol{G}_t = f(t)\boldsymbol{G}_t\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t , \qquad (2)$$

where $f(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}$ is the drift coefficient, $g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2$ is the diffusion coefficient, and $\boldsymbol{w}_t$ is a standard Wiener process. The reverse-time SDE from T to 0 (Song et al. 2021) corresponding to Eq. 2 is denoted as:

$$\mathrm{d}\boldsymbol{G}_t = [f(t)\boldsymbol{G}_t - g^2(t)\nabla_{\boldsymbol{G}}\log q_t(\boldsymbol{G}_t)]\mathrm{d}_t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t , \qquad (3)$$

where $\nabla_{\boldsymbol{G}}\log q_t(\boldsymbol{G}_t)$ is the graph score function and $\bar{\boldsymbol{w}}_t$ is the reverse-time standard Wiener process. We further split the reverse-time SDE into two parts that share the drift and diffusion coefficients as

$$\begin{cases} \mathrm{d}\boldsymbol{X}_t = [f(t)\boldsymbol{X}_t - g^2(t)\nabla_{\boldsymbol{X}}\log q_t(\boldsymbol{X}_t, \boldsymbol{A}_t)]\mathrm{d}_t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t^1 \\ \mathrm{d}\boldsymbol{A}_t = [f(t)\boldsymbol{A}_t - g^2(t)\nabla_{\boldsymbol{A}}\log q_t(\boldsymbol{X}_t, \boldsymbol{A}_t)]\mathrm{d}_t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t^2 \end{cases} . \qquad (4)$$

We use a neural network $\boldsymbol{\epsilon_\theta}(\boldsymbol{G}_t, \bar{\boldsymbol{A}}_t, t)$ with discrete graph structure conditioning to parameterize the σ_t-scaled partial scores in Eq. 4, where the node output of the neural network is denoted by $\boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{X}}(\boldsymbol{G}_t, \bar{\boldsymbol{A}}_t, t)$ to estimate $-\sigma_t\nabla_{\boldsymbol{X}}\log q_t(\boldsymbol{X}_t, \boldsymbol{A}_t)$, and the edge output is denoted by $\boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{A}}(\boldsymbol{G}_t, \bar{\boldsymbol{A}}_t, t)$ to estimate $-\sigma_t\nabla_{\boldsymbol{A}}\log q_t(\boldsymbol{X}_t, \boldsymbol{A}_t)$. The model is optimized by the objective (Ho, Jain, and Abbeel 2020; Song et al. 2021) as follows:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_t\{w(t)\mathbb{E}_{\boldsymbol{G}_0}\mathbb{E}_{\boldsymbol{G}_t|\boldsymbol{G}_0}[||\boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{X}}(\boldsymbol{G}_t, \bar{\boldsymbol{A}}_t, t) - \boldsymbol{\epsilon}_{\boldsymbol{X}}||_2^2 + ||\boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{A}}(\boldsymbol{G}_t, \bar{\boldsymbol{A}}_t, t) - \boldsymbol{\epsilon}_{\boldsymbol{A}}||_2^2]\} , \qquad (5)$$

where $w(t)$ is a given positive weighting function, $\boldsymbol{\epsilon_X}$ and $\boldsymbol{\epsilon_A}$ are the sampled Gaussian noise, and $\boldsymbol{G}_t = (\alpha_t\boldsymbol{X}_0 + \sigma_t\boldsymbol{\epsilon_X}, \alpha_t\boldsymbol{A}_0 + \sigma_t\boldsymbol{\epsilon_A})$. The training process is summarized in Algorithm 1. In practice, we use the Variance-Preserving (VP) SDE for implementation, with the definition that $f(t) = -\frac{1}{2}\beta(t)$, $g(t) = \sqrt{\beta(t)}$, and $\beta(t) =$

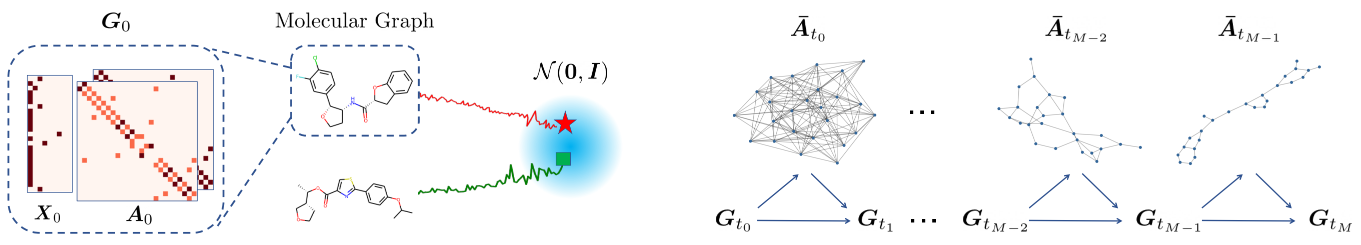

$\boldsymbol{G}_0$
Molecular Graph
$\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$
$\boldsymbol{X}_0$
$\boldsymbol{A}_0$
$\bar{\boldsymbol{A}}_{t_0}$
$\bar{\boldsymbol{A}}_{t_{M-2}}$
$\bar{\boldsymbol{A}}_{t_{M-1}}$
$\boldsymbol{G}_{t_0}$
$\boldsymbol{G}_{t_1}$
$\boldsymbol{G}_{t_{M-2}}$
$\boldsymbol{G}_{t_{M-1}}$
$\boldsymbol{G}_{t_M}$

Figure 1: (Left) Forward diffusion process that perturbs molecular graphs towards a known prior distribution. A graph $\boldsymbol{G}_0$ is denoted by a node feature matrix $\boldsymbol{X}_0$ and a two-channel edge matrix $\boldsymbol{A}_0$ for edge types and existence. (Right) Discretized reverse generative process with discrete graph structure conditioning.

$\bar{\beta}_{min} + t(\bar{\beta}_{max} - \bar{\beta}_{min})$. With the optimized $\boldsymbol{\epsilon_\theta}$ and numerical solvers discretizing the SDE trajectory, shown in the right of Figure 1, new graph samples can be generated by solving the parameterized reverse-time SDE.

Graph Noise Prediction Model

Since $\boldsymbol{\epsilon_\theta}(\boldsymbol{G}_t, \bar{\boldsymbol{A}}_t, t)$ can be considered to predict the noise that is added to the original graphs, we refer to it as the graph noise prediction model. The design of noise prediction models plays a key role in diffusion-based generation, but it is still an open problem for the graph domain. Applying the standard graph neural networks used in graph classification and link prediction tasks is not an appropriate choice due to the immediate real-number graph states and the complicated requirements for graph distribution learning. In the case of molecular graphs, the model should focus on local node-edge dependence for chemical valency rules and attempt to recover global graph patterns like edge sparsity, frequent ring subgraphs, and even atom-type distribution.

To meet these challenges, we propose a hybrid message passing block (HMPB) consisting of two different kinds of message passing layers to explicitly model structure and feature dependency in both real-valued matrices ($\boldsymbol{X}_t$ and $\boldsymbol{A}_t$) and discrete graphs ($\bar{\boldsymbol{A}}_t$). One is a standard message passing layer like GINE (Hu et al. 2020) to aggregate local neighbor node-edge features, relying on the decoded discrete graph structures. The other one is a fully-connected attention-based message passing layer to focus on global information extraction and transmission. We denote the node and edge representation update process in the l-th HMPB as

$$\begin{aligned} \boldsymbol{H}^{l+1}, \boldsymbol{E}^{l+1} &= \mathrm{HMPB}^l(\boldsymbol{H}^l, \boldsymbol{E}^l, \bar{\boldsymbol{A}}), \\ \text{with} \quad \boldsymbol{M}^{l+1} &= \mathrm{GINE}^l(\boldsymbol{H}^l, \boldsymbol{E}^l, \bar{\boldsymbol{A}}) + \mathrm{ATTN}^l(\boldsymbol{H}^l, \boldsymbol{E}^l), \\ \boldsymbol{H}^{l+1} &= \mathrm{FFN}_0^l(\boldsymbol{M}^{l+1}), \\ \boldsymbol{E}_{i,j}^{l+1} &= \mathrm{FFN}_1^l(\boldsymbol{M}_i^{l+1} + \boldsymbol{M}_j^{l+1}), \end{aligned} \tag{6}$$

where $\boldsymbol{H}^l \in \mathbb{R}^{N\times d}$ and $\boldsymbol{E}^l \in \mathbb{R}^{N\times N\times d}$ are node and edge inputs, $\boldsymbol{M}^{l+1} \in \mathbb{R}^{N\times d}$ is the aggregated message for nodes, $\boldsymbol{E}_{i,j}^{l+1} \in \mathbb{R}^d$ is the (i,j)-indexed edge output; ATTN^l is the full-connected attention layer; FFN^l is Feed Forward Network composed of the multilayer perceptron (MLP) and normalization layers. Here, the time t and residual connections are omitted for clarity. In particular, different from (Dwivedi and Bresson 2020; Ying et al. 2021; Kreuzer et al. 2021), our attention layer takes edge features as the gate for both the message and dot-product calculation to thoroughly interact with node features and bias the message passing. The key attention mechanism is denoted by

$$\begin{aligned} a_{i,j} &= \mathrm{softmax}\Big(\frac{(\tanh(\phi_0(\boldsymbol{E}_{i,j}))\cdot Q_i)K_j^\top}{\sqrt{d}}\Big), \\ \mathrm{ATTN}_i(\boldsymbol{H}, \boldsymbol{E}) &= \sum_{j=0}^{N-1} a_{i,j}(\tanh(\phi_1(\boldsymbol{E}_{i,j}))\cdot V_j), \end{aligned} \tag{7}$$

where Q, K, V are projected from node feature $\boldsymbol{H}$; $\boldsymbol{E}$ is the corresponding edge feature, ϕ_0 and ϕ_1 are learnable projections, and $\tanh$ is the activation layer.

For the initial features $\boldsymbol{H}^0$ and $\boldsymbol{E}^0$, we not only consider $\boldsymbol{X}_t$ and $\boldsymbol{A}_t$, but also extract structural encodings and relative positional encodings from $\bar{\boldsymbol{A}}_t$. Using the m-step random walk matrix from the discrete adjacency matrix, we adopt the arrival probability vector as node features and obtain the truncated shortest-path distance from the same matrix as edge features. Time information is added to the initial features with the sinusoidal position embedding (Vaswani et al. 2017). The final node and edge representations are respectively input to MLPs for graph noise prediction. Note that without any node ordering dependent operations, our graph noise prediction model built upon message passing mechanisms is permutation equivariant and implicitly defines the permutation invariant graph log-likelihood function.

ODE Solvers for Few-Step Graph Sampling

To generate graphs from the parameterized SDE in Eq. 4, the SDE trajectory needs to be stimulated with numerical solvers. The Euler-Maruyama (EM) solver is one of the simple and general solvers for SDEs, shown in Algorithm 2. Although our diffusion-based model can generate high-fidelity graphs in 200 steps (a.k.a., number of function evaluation (NFE)) using the EM solver shown in Figure 3, such a solver still needs relatively long steps to achieve convergence in the high-dimensional data space and fails to meet the fast sampling requirement. Since we preserve the continuous real-number graph diffusion formulation, one promising fast sampling method is to use the mature black-box ODE solvers for the probability flow ODE (Song et al. 2021) that shares the same marginal distribution at time t with the

SDE. Accordingly, the parameterized probability flow ODE for graphs is defined as

$$\mathrm{d}\boldsymbol{G}_t/\mathrm{d}t = f(t)\boldsymbol{G}_t + \frac{g^2(t)}{2\sigma_t}\boldsymbol{\epsilon_\theta}(\boldsymbol{G}_t, \bar{\boldsymbol{A}}_t, t)\,. \tag{8}$$

Recent works (Zhang and Chen 2022; Lu et al. 2022) claim that the general black-box ODE solvers ignore the semi-linear structure of the probability flow ODE and introduce additional discretization errors. Therefore, new fast solvers are being developed to take advantage of the special structure of the probability flow ODE.

For our graph ODE in Eq. 8, we further extend fast solvers based on the semi-linear ODE structure to generate high-quality graphs within a few steps. By introducing $\lambda_t := \log(\alpha_t/\sigma_t)$ and its inverse function $t_\lambda(\cdot)$ that satisfies $t = t_\lambda(\lambda(t))$, we change the subscript t to λ and get $\hat{\boldsymbol{G}}_\lambda := \boldsymbol{G}_{t_\lambda(\lambda)}$, $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\hat{\boldsymbol{G}}_\lambda, \bar{\boldsymbol{A}}'_\lambda, \lambda) := \boldsymbol{\epsilon_\theta}(\boldsymbol{G}_{t_\lambda(\lambda)}, \bar{\boldsymbol{A}}_{t_\lambda(\lambda)}, \lambda)$. We can derive the exact solution of the semi-linear probability flow ODE from time s to time t (Lu et al. 2022) as

$$\boldsymbol{G}_t = \frac{\alpha_t}{\alpha_s}\boldsymbol{G}_s - \alpha_t \int_{\lambda_s}^{\lambda_t} e^{-\lambda}\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\hat{\boldsymbol{G}}_\lambda, \bar{\boldsymbol{A}}'_\lambda, \lambda)\mathrm{d}\lambda\,. \tag{9}$$

With the analytical linear part, we only need to approximate the exponentially weighted integral of $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}$. This approximation can be achieved by various methods (Hochbruck and Ostermann 2005, 2010), and we follow the derivation from (Lu et al. 2022) to apply DPM-Solvers to graphs (denoted as GDPMS). Given the initial graph sampled from the prior distribution $\tilde{\boldsymbol{G}}_{t_0} := \boldsymbol{G}_T = (\boldsymbol{X}_T, \boldsymbol{A}_T)$ with the predefined time step schedules $\{t_i\}_{i=0}^M$, the sequence $\{\tilde{\boldsymbol{G}}_{t_i} = (\tilde{\boldsymbol{X}}_{t_i}, \tilde{\boldsymbol{A}}_{t_i})\}_{i=1}^M$ is calculated iteratively by the first-order GDPMS as follows:

$$\begin{cases} \tilde{\boldsymbol{X}}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{X}}_{t_{i-1}} - \gamma_i \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta},\boldsymbol{X}}(\tilde{\boldsymbol{G}}_{t_{i-1}}, \bar{\boldsymbol{A}}'_{t_{i-1}}, t_{i-1}) \\ \tilde{\boldsymbol{A}}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{A}}_{t_{i-1}} - \gamma_i \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta},\boldsymbol{A}}(\tilde{\boldsymbol{G}}_{t_{i-1}}, \bar{\boldsymbol{A}}'_{t_{i-1}}, t_{i-1}) \end{cases}, \tag{10}$$

where $\gamma_i = \sigma_{t_i}(e^{\lambda_{t_i} - \lambda_{t_{i-1}}} - 1)$, and discrete graph structure $\bar{\boldsymbol{A}}'_{t_{i-1}}$ is decoded from $\tilde{\boldsymbol{G}}_{t_{i-1}}$. The final graph sample is derived from $\tilde{\boldsymbol{G}}_{\boldsymbol{t_M}}$ with discretization. More details on high-order ODE samplers for graphs are provided in Appendix.

ODE-Based Graph Optimization Besides efficient sampling, the probability flow ODE offers latent representations for flexible data manipulation (Song et al. 2021). Based on the latent space determined by the parameterized ODE and the graph DPM-Solvers assisted by gradient guidance, we propose a useful optimization pipeline for the meaningful similarity-constrained molecule optimization task.

Specifically, we first train an extra time-dependent graph property predictor $\boldsymbol{R_\psi}(\boldsymbol{G}_t, t)$ on noisy graphs. Then we set up a solver for the parameterized ODE in Eq. 8 to map the initial molecular graphs at time 0 to the latent codes $\mathcal{G}_{t_\xi}$ at the time $t_\xi \in (0, T]$. Following the common optimization manipulation on latent space like (Jin, Barzilay, and Jaakkola 2018; Zang and Wang 2020), we use the predictor to predict properties on the graph latent representation and lead the optimization towards molecules with desired properties through the gradient ascent, producing a latent graph sequence $\{\mathcal{G}_{t_\xi}^k\}_{k=0}^K$. Instead of using the same ODE as in the forward encoding process, we introduce the gradient-guided ODE to further drive the sampling process to the high-property region during the decoding process from the latent space to the molecular graph space. The ODE with guidance can be modified from Eq. 8 as

$$\begin{cases} \mathrm{d}\boldsymbol{X}_t/\mathrm{d}t = f(t)\boldsymbol{X}_t + \frac{g^2(t)}{2\sigma_t}[\boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{X}} - r\sigma_t\nabla^*_{\boldsymbol{X}}\boldsymbol{R_\psi}] \\ \mathrm{d}\boldsymbol{A}_t/\mathrm{d}t = f(t)\boldsymbol{A}_t + \frac{g^2(t)}{2\sigma_t}[\boldsymbol{\epsilon}_{\boldsymbol{\theta},\boldsymbol{A}} - r\sigma_t\nabla^*_{\boldsymbol{A}}\boldsymbol{R_\psi}] \end{cases}, \tag{11}$$

where r is the guidance weight, ∇^* refers to the unit normalized gradients, and the input $(\boldsymbol{G}_t, \bar{\boldsymbol{A}}_t, t)$ for $\boldsymbol{\epsilon_\theta}$ and $(\boldsymbol{G}_t, t)$ for $\boldsymbol{R_\psi}$ are omitted for simplicity. Notably, the GDPMS in Eq. 10 can still work for the gradient-guided ODE by constructing the $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}$ with the predictor gradients accordingly. The proposed pipeline can also be flexibly extended for multi-objective optimization by expanding the gradient guidance from multiple property prediction networks.

Algorithm 1: Optimizing CDGS

Require: original graph data $\boldsymbol{G_0} = (\boldsymbol{X_0}, \boldsymbol{A_0})$, graph noise prediction model $\boldsymbol{\epsilon_\theta}$, schedule function $\alpha(\cdot)$ and $\sigma(\cdot)$, quantized function $quantize(\cdot)$

```
1: Sample $t \sim \mathcal{U}(0,1]$, $\boldsymbol{\epsilon_X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\boldsymbol{\epsilon_A} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
2: $\boldsymbol{G}_t = (\boldsymbol{X}_t, \boldsymbol{A}_t) \leftarrow (\alpha(t)\boldsymbol{X}_0 + \sigma(t)\boldsymbol{\epsilon_X}, \alpha(t)\boldsymbol{A}_0 + \sigma(t)\boldsymbol{\epsilon_A})$
3: $\bar{\boldsymbol{A}}_t \leftarrow quantize(\boldsymbol{A}_t)$
4: $\boldsymbol{\epsilon_\theta^X}, \boldsymbol{\epsilon_\theta^A} \leftarrow \boldsymbol{\epsilon_\theta}(\boldsymbol{G}_t, \bar{\boldsymbol{A}}_t, t)$
5: Minimize $||\boldsymbol{\epsilon_\theta^X} - \boldsymbol{\epsilon_X}||_2^2 + ||\boldsymbol{\epsilon_\theta^A} - \boldsymbol{\epsilon_A}||_2^2$
```

Algorithm 2: Sampling from CDGS with the Euler-Maruyama method

Require: number of time steps N, graph noise prediction model $\boldsymbol{\epsilon_\theta}$, drift coefficient function $f(\cdot)$, diffusion coefficient function $g(\cdot)$, schedule function $\sigma(\cdot)$, quantized function $quantize(\cdot)$, post-processing function $post(\cdot)$

```
1: Sample initial graph $\boldsymbol{G} \leftarrow (\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \boldsymbol{A} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}))$,
2: $\Delta t = \frac{T}{N}$
3: for $i \leftarrow N$ to $1$ do
4:     $\bar{\boldsymbol{A}} \leftarrow quantize(\boldsymbol{A})$
5:     $\boldsymbol{\epsilon_X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\boldsymbol{\epsilon_A} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
6:     $t \leftarrow i\Delta t$
7:     $\boldsymbol{\epsilon_\theta^X}, \boldsymbol{\epsilon_\theta^A} \leftarrow \boldsymbol{\epsilon_\theta}(\boldsymbol{G}, \bar{\boldsymbol{A}}, t)$
8:     $\boldsymbol{X} \leftarrow \boldsymbol{X} - (f(t)\boldsymbol{X} + \frac{g(t)^2}{\sigma(t)}\boldsymbol{\epsilon_\theta^X})\Delta t + g(t)\sqrt{\Delta t}\boldsymbol{\epsilon_X}$
9:     $\boldsymbol{A} \leftarrow \boldsymbol{A} - (f(t)\boldsymbol{A} + \frac{g(t)^2}{\sigma(t)}\boldsymbol{\epsilon_\theta^A})\Delta t + g(t)\sqrt{\Delta t}\boldsymbol{\epsilon_A}$
10: return $post(\boldsymbol{X}, \boldsymbol{A})$
```

Related Work

Molecule Generation Early attempts for molecule generation introduce sequence-based generative models and represent molecules as SMILES strings (Gómez-Bombarelli et al. 2018; Kusner, Paige, and Hernández-Lobato 2017; Dai et al. 2018). Besides the challenge from long dependency modeling, these methods may exhibit low validity rates since the SMILES string does not ensure absolute validity. Therefore, graphs are more commonly used to represent molecule structures in recent studies. Various graph generative models have been proposed to construct graphs autoregressively or in a one-shot form, based on different types of generative models, including variational auto-encoders (Simonovsky and Komodakis 2018; Liu et al. 2018), generative adversarial networks (De Cao and Kipf 2018; Assouel et al. 2018), and normalizing flows (Shi et al. 2020; Luo, Yan, and Ji 2021; Lippe and Gavves 2021; Zang and Wang 2020). Compared to these models, our diffusion-based model advances in stable training and flexible model architecture to consider discrete graph structures for complicated dependency modeling. In addition, (Jin, Barzilay, and Jaakkola 2018; Ahn et al. 2022) adopt an effective tree-based graph formulation for molecules, while our method keeps the general graph settings and models permutation invariant distributions.

Diffusion Models This new family of generative models (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020) correlated with score-based models (Song et al. 2021; Song and Ermon 2019) has demonstrated great power in the generation of high-dimensional data such as images. For molecule science, in addition to molecular graph generation (Jo, Lee, and Hwang 2022), diffusion models have also been applied to generate molecular conformations (Xu et al. 2022; Jing et al. 2022) and 3D molecular structures (Hoogeboom et al. 2022). Our framework greatly differs from the previous diffusion-based molecule generation in the conditional reverse process and the unified model design instead of separate models for nodes and edges. Moreover, we promote efficient molecular graph generation with training-free samplers, which is primarily investigated in the image domain (Liu et al. 2022; Zhang and Chen 2022; Lu et al. 2022).

Experiment

In this section, we display the experimental results of the proposed discrete graph structure assisted diffusion framework on multiple datasets. We provide more experiment details in Appendix, and we release the code at https://github.com/GRAPH-0/CDGS.

Molecular Graph Generation

Experimental Setup We train and evaluate models on two molecule datasets, ZINC250k (Irwin et al. 2012) and QM9 (Ramakrishnan et al. 2014). Before converting to graphs, all molecules are processed to the kekulized form using RDKit (Landrum 2016), where hydrogen atoms are removed and aromatic bonds are replaced by double bonds. We evaluate generation quality on $10,000$ generated molecules with the following widely used metrics. **Fréchet ChemNet Distance (FCD)** (Preuer et al. 2018) calculates the distance between the reference molecule set and the generated set with the activations of the penultimate layer of ChemNet. Lower FCD values indicate higher similarity between the two distributions. Following (Jo, Lee, and Hwang 2022), we report FCD values after validity checking and valency correction

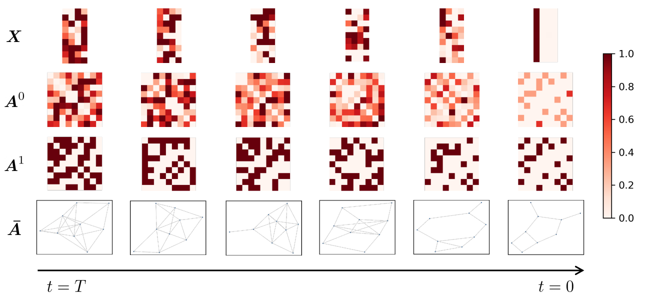


Figure 2: Molecular graph normalized visualization at different steps in the reverse generative process from the model trained on QM9. $\boldsymbol{X}$ is the node feature matrix, $\boldsymbol{A}^0$ is the edge type matrix, $\boldsymbol{A}^1$ is the quantized edge existence matrix, and $\bar{\boldsymbol{A}}$ is the discrete graph structure visualization.

since FCD is only calculated on valid molecules. **Neighborhood subgraph pairwise distance kernel (NSPDK)** is the distance measured by mean maximum discrepancy (MMD), which incorporates node and edge features along with the underlying graph structure. FCD and NSPDK, one from the perspective of molecules and the other from the perspective of graphs, are crucial for the evaluation of molecular graph distribution learning (Jo, Lee, and Hwang 2022). **VALID w/o check** is the percentage of valid molecules without post-hoc chemical valency correction. Here, we follow the setting of (Zang and Wang 2020; Jo, Lee, and Hwang 2022) to consider the formal charges for valency checking. We also report the results of three metrics that are used commonly but have obvious marginal effects, *i.e.*, the ratio of valid molecules (**VALID**), the ratio of unique molecules (**UNIQUE**), and the ratio of novel molecules with reference to the training set (**NOVEL**).

Baselines We compare our CDGS with several autoregressive and one-shot molecular graph generative models, including **GraphAF** (Shi et al. 2020), **GraphDF** (Luo, Yan, and Ji 2021), **MoFlow** (Zang and Wang 2020), **GraphCNF** (Lippe and Gavves 2021), **EDP-GNN** (Niu et al. 2020), **GraphEBM** (Liu et al. 2021), and **GDSS** (Jo, Lee, and Hwang 2022). **GraphAF+FC** and **GraphDF+FC** are the modified versions considering formal charges for fair comparison. **GDSS-EM** is the result sampled with the EM solver, and **GDSS-VP-EM** is retrained with VPSDE, sharing the same SDE parameters with our model.

Generation Quality The molecular graph generation quality benchmark results on ZINC250k and QM9 are reported in Table 1. We run three times for our method and report the mean performance. We provide the performance bound on two distribution metrics by measuring the distance between preprocessed training molecules and original test molecules. In the first three non-trivial metrics across two different molecule datasets, CDGS with the EM solver outperforms state-of-the-art molecular graph generative models. The high validity rate before valency checking shows that CDGS learns the chemical valency rule successfully and avoids unrealistically frequent valency correction. Further-

	Method	VALID w/o check (%) ↑	NSPDK ↓	FCD ↓	VALID (%) ↑	UNIQUE (%) ↑	NOVEL (%) ↑
	Train	-	*5.91e-5*	*0.985*	-	-	-
Autoreg.	GraphAF	68.00	0.044	16.289	100.00	99.10	100.00
	GraphAF+FC	68.47	0.044	16.023	100.00	98.64	99.99
	GraphDF	89.03	0.176	34.202	100.00	99.16	100.00
	GraphDF+FC	90.61	0.177	33.546	100.00	99.63	100.00
One-shot	MoFlow	63.11	0.046	20.931	100.00	99.99	100.00
	GraphCNF	96.35	0.021	13.532	100.00	99.98	100.00
	EDP-GNN	82.97	0.049	16.737	100.00	99.79	100.00
	GraphEBM	5.29	0.212	35.471	99.96	98.79	100.00
	GDSS	97.01	0.019	14.656	100.00	99.64	100.00
	GDSS-EM	15.97	0.075	24.310	100.00	100.00	100.00
	GDSS-VP-EM	33.01	0.048	24.471	100.00	100.00	100.00
	CDGS-EM	**98.13**	**7.03e-4**	**2.069**	100.00	99.99	99.99
	CDGS-GDPMS-200	96.19	0.001	3.037	100.00	99.98	99.99
	CDGS-GDPMS-50	95.56	0.002	3.567	100.00	99.98	99.99
	CDGS-GDPMS-30	93.49	0.003	4.498	100.00	99.99	99.99

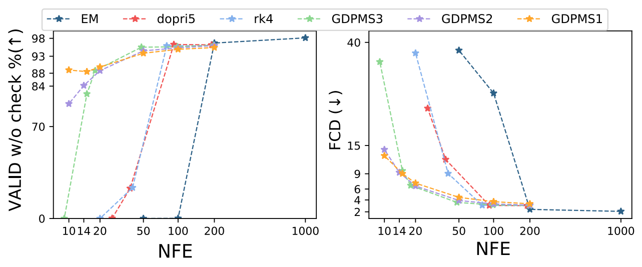
	Method	VALID w/o check (%) ↑	NSPDK ↓	FCD ↓	VALID (%) ↑	UNIQUE (%) ↑	*NOVEL (%)* ⋆
	Train	-	*1.36e-4*	*0.057*	-	-	-
Autoreg.	GraphAF	67.00	0.020	5.268	100.00	94.51	88.83
	GraphAF+FC	74.43	0.021	5.625	100.00	88.64	86.59
	GraphDF	82.67	0.063	10.816	100.00	97.62	98.10
	GraphDF+FC	93.88	0.064	10.928	100.00	98.58	98.54
One-shot	MoFlow	91.36	0.017	4.467	100.00	98.65	94.72
	EDP-GNN	47.52	0.005	2.680	100.00	99.25	86.58
	GraphEBM	8.22	0.030	6.143	100.00	97.90	97.01
	GDSS	95.72	0.003	2.900	100.00	98.46	86.27
	GDSS-EM	66.01	0.016	5.112	100.00	90.05	94.24
	GDSS-VP-EM	86.02	0.013	4.588	100.00	89.03	88.63
	CDGS-EM	**99.68**	**3.08e-4**	**0.200**	100.00	96.83	69.62
	CDGS-GDPMS-200	99.54	3.68e-4	0.269	100.00	97.20	72.52
	CDGS-GDPMS-50	99.47	3.85e-4	0.289	100.00	97.27	72.38
	CDGS-GDPMS-30	99.18	4.13e-4	0.326	100.00	97.42	72.52

Table 1: Generation performance on ZINC250k (Up) and QM9 (Down). The best results in first three metrics are highlighted in bold. The novelty metric on QM9 dataset denoted with ⋆ is debatable due to its contradiction with distribution learning.

more, with much lower NSPDK and FCD values, CDGS learns the underlying distribution more faithfully in both graph and chemical space. CDGS achieves such performance without any Langevin correction steps in sampling, while previous diffusion-based GDSS drops off obviously with the pure EM solver. Using the same SDE parameters, the performance gap between GDSS-VP-EM and CDGS-EM further demonstrates the effectiveness of our framework design. Another noteworthy point is that, equipped with the 3rd-order GDPMS, our proposed model maintains excellent generation ability with limited NFE decreasing from 200 to 30. Extra visualization of generated molecules is provided in Appendix.

We also point out that the novelty metric on the QM9 dataset seems debatable because the QM9 dataset is almost an exhaustive list of molecules that adhere to a predetermined set of requirements (Vignac and Frossard 2022; Hoogeboom et al. 2022). Therefore, a molecule that is thought to be novel violates the constraints, which means the model is unable to capture the dataset properties. This metric is kept for experiment completeness.

Fast Sampling To explore fast and high-quality few-step molecular graph sampling, we compare the sampling quality of CDGS with different types of numerical solvers, including GDPMS with different orders, the EM solver, and black-box ODE solvers. For black-box ODE solvers, we pick out an adaptive-step and a fixed-step neural ODE solver implemented by (Chen et al. 2018), that is, Runge-Kutta of order 5 of Dormand-Prince-Shampine (dopri5) and Fourth-order Runge-Kutta with 3/8 rule (rk4). As shown in Figure 3, based on our conditional diffusion framework, the EM solver generates high-quality graphs between 200 NFE and 1000 NFE, but fails to converge under fewer NFE. The black-box neural ODE solvers can obtain acceptable quality



Method	GraphAF	GraphDF	MoFlow	GDSS	Our-50	Our-30	Our-10
Time (s)	$2.89e^2$	$3.19e^3$	1.54	$1.42e^2$	$3.79e^1$	$2.38e^1$	8.38

Figure 3: (Up) Few-step molecular graph sampling results for various numerical solvers. (Down) The wall-clock time taken to generate 512 molecular graphs.

	GraphAF-RL		MoFlow	
δ	**Improvement**	**Success**	**Improvement**	**Success**
0.0	13.13±6.89	100%	8.61±5.44	99%
0.2	11.90±6.86	100%	7.06±5.04	97%
0.4	8.21±6.51	100%	4.71±4.55	86%
0.6	4.98±6.49	97%	2.10±2.86	58%
	GraphEBM		CDGS	
δ	**Improvement**	**Success**	**Improvement**	**Success**
0.0	15.75±7.40	99%	12.83±7.01	100%
0.2	8.40±6.38	94%	11.70±6.84	100%
0.4	4.95±5.90	79%	9.56±6.33	100%
0.6	3.15±5.08	45%	5.10±5.80	98%

Table 2: Similarity-constrained molecule property optimization performance.

at around 50 NFE. The GDPMS displays clear superiority in the range below 50 NFE. Notably, the 1st-order GDPMS still generates reasonable molecular graphs with 10 NFE. For the running time comparison, CDGS equipped with GDPMS takes much less time compared to autoregressive GraphAF and GraphDF, and makes an obvious improvement towards GDSS. MoFlow spends the least time but fails to generate high-fidelity samples according to Table 1. In conclusion, benefiting from the framework design and the ODE solvers utilizing the semi-linear structure, we achieve great advancement in fast sampling for complex molecular graphs.

Ablation Studies We conduct ablation analysis on the ZINC250k dataset to verify the effectiveness of our framework. In Figure 4, with the goal to generate high-quality molecular graphs efficiently, we report the results using GDPMS with 50 NFE, which is sufficient to obtain converged samples. Taking CDGS with 64 hidden dimensions (**64ch**) as reference, we first remove the discrete graph structure related components and remain with our edge-gated attention layers (**ATTN**), then further remove the edge existence variable (**W-ADJ**). The variant using GINE without attention layers is denoted as **GINE**.

We emphasize that VALID w/o check and FCD metrics are complementary and should be combined to assess molecule generation quality, because the former only reflects the valency validity of local atom and bond connections, whereas the latter is obtained after valency corrections and focuses more on global molecule similarity. It can be observed from Figure 4 that: (1) Compared to 64ch, ATTN has a lower validity rate and gets a close FCD after more undesirable corrections, while GINE achieves high validity rates but fails to capture more global information. It proves that the proposed attention module is crucial for global distribution learning and that discrete graph structures greatly help to capture the chemical valency rule. (2) The comparison of W-ADJ and ATTN shows that separating the edge existence in the formulation also makes contributions to molecule validity. In addition, W-ADJ outperforms GDSS-VP-EM in Table 1, showing the effectiveness of explicitly interacting node and edge representations using a unified graph noise prediction model. (3) It is necessary to increase hidden dimensions (**128ch**, **256ch**) to better handle the complexity of drug-like molecules in the ZINC250k dataset.

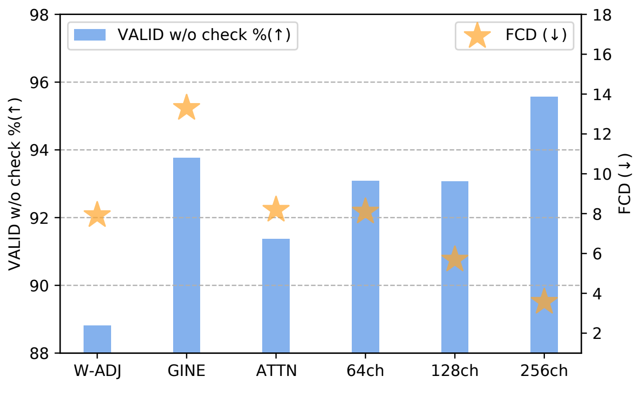


Figure 4: Ablation on the ZINC250k dataset.

Similarity-Constrained Property Optimization We also show how our diffusion framework can be used for similarity-constrained property optimization. Following (Shi et al. 2020; Zang and Wang 2020), we select 800 molecules with low p-logP scores (*i.e.*, the octanol-water partition coefficients penalized by synthetic accessibility and number of long cycles) as the initial molecules for optimization. We aim to generate new molecules with a higher p-logP while keeping similarity to the original molecules with a threshold δ. The similarity metric is defined as Tanimoto similarity with Morgan fingerprints (Rogers and Hahn 2010). The property predictor is composed of 6 hybrid message passing blocks with RGCN (Schlichtkrull et al. 2018) as the non-attention layer for differentiation. We pretrain the time-dependent predictor on perturbed graphs of the ZINC250k dataset for 200 epochs. Each initial molecular graph is encoded into latent codes at the middle time $t_\xi = 0.3$ through the forward-time ODE solver. After 50 gradient ascent steps, all latent codes are decoded back to molecules with another gradient-guided reverse-time ODE solver. This procedure is repeated 20 times with a different number of atoms to search for the highest property molecule that satisfies the similarity constraint.

	Community-small				Ego-small				Enzymes				Ego			
	Deg.	Clus.	Spec.	GIN.	Deg.	Clus.	Spec.	GIN.	Deg.	Clus.	Spec.	GIN.	Deg.	Clus.	Spec.	GIN.
Train/Test	*0.035*	*0.067*	*0.045*	*0.037*	*0.025*	*0.029*	*0.027*	*0.016*	*0.011*	*0.011*	*0.011*	*0.007*	*0.009*	*0.009*	*0.009*	*0.005*
ER	0.300	0.239	0.100	0.278	0.200	0.094	0.361	0.230	0.844	0.381	0.104	0.808	0.738	0.397	0.868	0.118
VGAE	0.391	0.257	0.095	0.360	0.146	0.046	0.249	0.089	0.811	0.514	0.153	0.716	0.873	1.210	0.935	0.520
GraphRNN	0.106	0.115	0.091	0.353	0.155	0.229	0.167	0.472	0.397	0.302	0.260	1.495	0.140	0.755	0.316	1.283
GraphRNN-U	0.410	0.297	0.103	0.970	0.471	0.416	0.398	0.915	0.932	1.000	0.367	1.263	1.413	1.097	1.110	1.317
GRAN	0.125	0.164	0.111	0.196	0.096	0.072	0.095	0.106	0.215	0.147	0.034	0.069	0.594	0.425	1.025	0.244
GRAN-U	0.106	0.127	0.083	0.164	0.155	0.229	0.167	0.094	0.343	0.122	0.041	0.242	0.099	0.170	0.179	0.128
EDP-GNN	0.100	0.140	0.085	0.125	0.026	0.032	0.037	0.031	0.120	0.644	0.070	0.119	0.553	0.605	0.374	0.295
GDSS	0.102	0.125	0.087	0.137	0.041	0.036	0.041	0.041	0.118	0.071	0.053	0.028	0.314	0.776	0.097	0.156
CDGS-EM	**0.052**	**0.080**	**0.064**	**0.062**	**0.025**	**0.031**	**0.033**	**0.025**	**0.048**	**0.070**	**0.033**	**0.024**	**0.036**	**0.075**	**0.026**	**0.026**
CDGS-GDPM	0.100	0.121	0.084	0.120	0.116	0.064	0.141	0.052	0.140	0.127	0.041	0.040	0.157	0.109	0.153	0.064

Table 3: Generation performance on generic graph datasets. The better results are indicated by a closer value with the performance of training graphs, and the best results are in bold.

Results for the similarity-constrained optimization are summarized in Table 2. **GraphAF-RL** is the representative method combined with reinforcement learning, **MoFlow** is a flow-based method, and **GraphEBM** is an energy-based method for molecule optimization. With the similarity constraint ($\delta > 0$), CDGS outperforms MoFlow and GraphEBM in terms of success rate and mean property improvement, showing competitive performance to the RL-based method. Since RL-based methods require heavy property evaluator calls, which is unrealistic in some optimization scenarios, our framework could serve as a useful supplement for drug discovery tasks.

Generic Graph Generation

Experimental Setup To display the graph structure distribution learning ability, we validate CDGS on four common generic graph datasets with various graph sizes and characteristics: (1) *Community-small*, 100 two-community graphs generated by the Erdős-Rényi model (E-R) (Erdős, Rényi et al. 1960) with $p = 0.7$, (2) *Ego-small*, 200 one-hop ego graphs extracted from Citeseer network (Sen et al. 2008), (3) *Enzymes*, 563 protein graphs with more than 10 nodes from BRENDA database (Schomburg et al. 2004), (4) *Ego*, 757 three-hop ego graphs extracted from Citeseer network (Sen et al. 2008). We use $8 : 2$ as the split ratio for train/test. We generate 1024 graphs for the evaluation on Community-small and Ego-small, and generate the same number of graphs as the test set on Enzymes and Ego. We follow the advice from (O'Bray et al. 2022) to evaluate discrete graph structure distribution. Three graph-level structure descriptor functions are selected: degree distribution (**Deg.**), clustering coefficient distribution (**Clus.**) and Laplacian spectrum histograms (**Spec.**). We use MMD with the radial basis function kernel (RBF) to calculate the distance on features extracted by graph descriptors. To accurately evaluate distribution distance, different from (You et al. 2018b; Liao et al. 2019; Niu et al. 2020) using a static smoothing hyperparameter for MMD, we provide a set of hyperparameters and report the largest distance (Thompson et al. 2022; Huang et al. 2022). We also consider a well-established comprehensive neural-network-based metric (**GIN.**) from (Thompson et al. 2022).

Baselines Apart from scored-based models (EDP-GNN and GDSS), we compare CDGS with a classical method (**ER** (Erdős, Rényi et al. 1960)), a VAE-based method (**VGAE** (Kipf and Welling 2016)), and two strong autoregressive graph generative models (**GraphRNN** (You et al. 2018b), **GRAN** (Liao et al. 2019)). **GraphRNN-U** and **GRAN-U** are trained with uniform node orderings to alleviate the bias from specific ordering strategies.

Sampling Quality Table 3 displays that CDGS consistently achieves better performance than score-based models and autoregressive models among four datasets. Especially for the large Ego dataset, CDGS still generates graphs with high fidelity while the diffusion-based GDSS fails in Deg. and Clus. metrics. Using GDPMS with 30 steps, our method also generates graph structures with acceptable quality. Thanks to the appropriate framework design and the emphasis on evolving discrete graph structures during the generative process, CDGS effectively captures the underlying distribution of graph topology.

Conclusion

We present a novel conditional diffusion model for molecular graph generation that takes advantage of discrete graph structure conditioning and delicate graph noise prediction model design. Our model markedly outperforms existing molecular graph generative methods in both graph space and chemical space for distribution learning, and also performs well for generic graph generation. By adapting fast ODE solvers for graphs, we utilize our framework to make advances in efficient graph sampling and facilitate similarity-constrained optimization. In the future, we plan to apply our model to molecule generation with complex conditions, such as target proteins.

Acknowledgments

We would like to thank Le Yu for the helpful discussion. This work was supported in part by the National Natural Science Foundation of China (62272023, 51991395 and U21A20516).

References

Ahn, S.; Chen, B.; Wang, T.; and Song, L. 2022. Spanning Tree-based Graph Generation for Molecules. In *ICLR*.

Assouel, R.; Ahmed, M.; Segler, M. H. S.; Saffari, A.; and Bengio, Y. 2018. DEFactor: Differentiable Edge Factorization-based Probabilistic Graph Generation. *arXiv preprint arXiv: 1811.09766*.

Austin, J.; Johnson, D. D.; Ho, J.; Tarlow, D.; and van den Berg, R. 2021. Structured Denoising Diffusion Models in Discrete State-Spaces. In *NeurIPS*, 17981–17993.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural Ordinary Differential Equations. In *NeurIPS*.

Dai, H.; Tian, Y.; Dai, B.; Skiena, S.; and Song, L. 2018. Syntax-Directed Variational Autoencoder for Structured Data. In *ICLR*.

De Cao, N.; and Kipf, T. 2018. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*.

Dwivedi, V. P.; and Bresson, X. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

Erdős, P.; Rényi, A.; et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1): 17–60.

Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; and Aspuru-Guzik, A. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising Diffusion Probabilistic Models. In *NeurIPS*.

Hochbruck, M.; and Ostermann, A. 2005. Explicit Exponential Runge-Kutta Methods for Semilinear Parabolic Problems. *SIAM J. Numer. Anal.*, 43(3): 1069–1090.

Hochbruck, M.; and Ostermann, A. 2010. Exponential integrators. *Acta Numer.*, 19: 209–286.

Hoogeboom, E.; Nielsen, D.; Jaini, P.; Forré, P.; and Welling, M. 2021. Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions. In *NeurIPS*, 12454–12465.

Hoogeboom, E.; Satorras, V. G.; Vignac, C.; and Welling, M. 2022. Equivariant Diffusion for Molecule Generation in 3D. In *ICML*, 8867–8887.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V. S.; and Leskovec, J. 2020. Strategies for Pre-training Graph Neural Networks. In *ICLR*.

Huang, H.; Sun, L.; Du, B.; Fu, Y.; and Lv, W. 2022. GraphGDP: Generative Diffusion Processes for Permutation Invariant Graph Generation. In *ICDM*.

Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; and Coleman, R. G. 2012. ZINC: A Free Tool to Discover Chemistry for Biology. *J. Chem. Inf. Model.*, 52(7): 1757–1768.

Jin, W.; Barzilay, R.; and Jaakkola, T. S. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. In Dy, J. G.; and Krause, A., eds., *ICML*, 2328–2337.

Jing, B.; Corso, G.; Chang, J.; Barzilay, R.; and Jaakkola, T. S. 2022. Torsional Diffusion for Molecular Conformer Generation. *arXiv preprint arXiv:2206.01729*.

Jo, J.; Lee, S.; and Hwang, S. J. 2022. Score-based Generative Modeling of Graphs via the System of Stochastic Differential Equations. In *ICML*, 10362–10383.

Kingma, D.; Salimans, T.; Poole, B.; and Ho, J. 2021. Variational Diffusion Models. In *NeurIPS*.

Kipf, T. N.; and Welling, M. 2016. Variational graph autoencoders. *arXiv preprint arXiv:1611.07308*.

Kreuzer, D.; Beaini, D.; Hamilton, W.; Létourneau, V.; and Tossou, P. 2021. Rethinking graph transformers with spectral attention. In *NeurIPS*, volume 34.

Kusner, M. J.; Paige, B.; and Hernández-Lobato, J. M. 2017. Grammar Variational Autoencoder. In *ICML*, 1945–1954.

Landrum, G. 2016. RDKit: Open-Source Cheminformatics Software.

Liao, R.; Li, Y.; Song, Y.; Wang, S.; Hamilton, W. L.; Duvenaud, D.; Urtasun, R.; and Zemel, R. S. 2019. Efficient Graph Generation with Graph Recurrent Attention Networks. In *NeurIPS*, 4257–4267.

Lippe, P.; and Gavves, E. 2021. Categorical Normalizing Flows via Continuous Transformations. In *ICLR*.

Liu, L.; Ren, Y.; Lin, Z.; and Zhao, Z. 2022. Pseudo Numerical Methods for Diffusion Models on Manifolds. In *ICLR*.

Liu, M.; Yan, K.; Oztekin, B.; and Ji, S. 2021. GraphEBM: Molecular graph generation with energy-based models. *arXiv preprint arXiv:2102.00546*.

Liu, Q.; Allamanis, M.; Brockschmidt, M.; and Gaunt, A. L. 2018. Constrained Graph Variational Autoencoders for Molecule Design. In *NeurIPS 2018*, 7806–7815.

Lu, C.; Zhou, Y.; Bao, F.; Chen, J.; Li, C.; and Zhu, J. 2022. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. *arXiv preprint arXiv:2206.00927*.

Luo, Y.; Yan, K.; and Ji, S. 2021. GraphDF: A Discrete Flow Model for Molecular Graph Generation. In *ICML*, 7192–7203.

Niu, C.; Song, Y.; Song, J.; Zhao, S.; Grover, A.; and Ermon, S. 2020. Permutation invariant graph generation via score-based generative modeling. In *AISTATS*, 4474–4484.

O'Bray, L.; Horn, M.; Rieck, B.; and Borgwardt, K. 2022. Evaluation Metrics for Graph Generative Models: Problems, Pitfalls, and Practical Solutions. In *ICLR*.

Preuer, K.; Renz, P.; Unterthiner, T.; Hochreiter, S.; and Klambauer, G. 2018. Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery. *J. Chem. Inf. Model.*, 58(9): 1736–1741.

Ramakrishnan, R.; Dral, P. O.; Rupp, M.; and von Lilienfeld, O. A. 2014. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1.

Rogers, D.; and Hahn, M. 2010. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.*, 50(5): 742–754.

Schlichtkrull, M. S.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*, 593–607.

Schomburg, I.; Chang, A.; Ebeling, C.; Gremse, M.; Heldt, C.; Huhn, G.; and Schomburg, D. 2004. BRENDA, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1): D431–D433.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Shi, C.; Xu, M.; Zhu, Z.; Zhang, W.; Zhang, M.; and Tang, J. 2020. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. In *ICLR*.

Simonovsky, M.; and Komodakis, N. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *ICANN*, 412–422. Springer.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2256–2265.

Song, Y.; and Ermon, S. 2019. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, volume 32.

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *ICLR*.

Thompson, R.; Knyazev, B.; Ghalebi, E.; Kim, J.; and Taylor, G. W. 2022. On Evaluation Metrics for Graph Generative Models. In *ICLR*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*.

Vignac, C.; and Frossard, P. 2022. Top-N: Equivariant Set and Graph Generation without Exchangeability. In *ICLR*.

Xu, M.; Yu, L.; Song, Y.; Shi, C.; Ermon, S.; and Tang, J. 2022. GeoDiff: A Geometric Diffusion Model for Molecular Conformation Generation. In *ICLR*.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T. 2021. Do Transformers Really Perform Badly for Graph Representation? In *NeurIPS*, 28877–28888.

You, J.; Liu, B.; Ying, Z.; Pande, V. S.; and Leskovec, J. 2018a. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *NeurIPS*, 6412–6422.

You, J.; Ying, R.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018b. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*, 5708–5717.

Zang, C.; and Wang, F. 2020. MoFlow: an invertible flow model for generating molecular graphs. In *SIGKDD*, 617–626.

Zhang, Q.; and Chen, Y. 2022. Fast Sampling of Diffusion Models with Exponential Integrator. *arXiv preprint arXiv:2204.13902*.