Diffusion Twigs with Loop Guidance for Conditional Graph Generation

Giangiacomo Mercatali [†] [*]
HES-SO Genève
University of Manchester
giangiacomo.mercatali@hesge.ch

Yogesh Verma [†]
Aalto University
yogesh.verma@aalto.fi

Andre Freitas
Idiap Research Institute
University of Manchester
NBC, CRUK Manchester Institute
andre.freitas@idiap.ch

Vikas Garg
YaiYai Ltd & Aalto University
vgarg@csail.mit.edu

Abstract

We introduce a novel score-based diffusion framework named *Twigs* that incorporates multiple co-evolving flows for enriching conditional generation tasks. Specifically, a central or *trunk* diffusion process is associated with a primary variable (e.g., graph structure), and additional offshoot or *stem* processes are dedicated to dependent variables (e.g., graph properties or labels). A new strategy, which we call *loop guidance*, effectively orchestrates the flow of information between the trunk and the stem processes during sampling. This approach allows us to uncover intricate interactions and dependencies, and unlock new generative capabilities. We provide extensive experiments to demonstrate strong performance gains of the proposed method over contemporary baselines in the context of conditional graph generation, underscoring the potential of Twigs in challenging generative tasks such as inverse molecular design and molecular optimization. Code is available at https://github.com/Aalto-QuML/Diffusion_twigs.

1 Introduction

Conditional graph generation is a fundamental problem in scientific domains such as *de novo* drug design [21, 43, 74] and material design [39]. However, searching for new molecules with desired physicochemical properties poses significant challenges to traditional brute-force methods due to the vast combinatorial spaces [64]. With the advent of neural networks [44], deep generative models have emerged as a powerful tool for learning informative conditional representations of molecules, facilitating the development of *in silico* methods for chemical design [16, 31, 61, 73].

Score-based diffusion generative models (SGMs) and denoising probabilistic diffusion models (DDPMs) [24, 67] have recently emerged as powerful techniques for training deep networks on graph-structured data, with applications spanning molecular design [37, 53, 36, 81], molecular docking [6], molecular dynamics simulations [78], protein folding [79], and backbone modeling [70]. Notably, diffusion models exhibit superior capabilities for *conditional* graph generation, excelling in both discrete [26, 75, 49] and continuous [3, 28, 45, 11] settings. The training of the mentioned conditional diffusion models is achieved by two types of diffusion guidance algorithms: *classifier-based guidance*

[†]Equal Contribution. Order decided via coin flip.
[*]Work done while at the University of Manchester

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

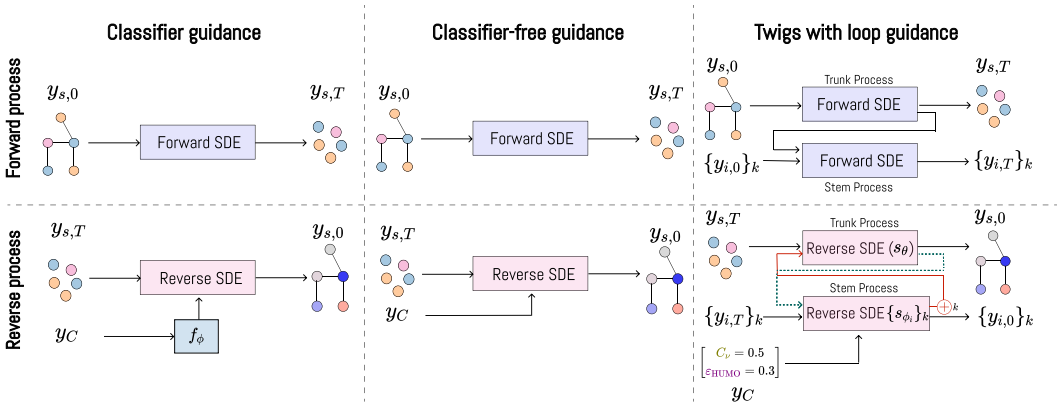

Figure 1: **Overview of the proposed method (`Twigs`).** We define two types of diffusion processes: (1) multiple *Stem* processes (s_{ϕ_i}), which unravel the interactions between graph structure and single properties, and (2) the *Trunk* process, which orchestrates the combination of the graph structure score from s_θ with the stem process contributions from s_{ϕ_i}. During the forward process, the structure $\mathbf{y}_s$ and the properties $\{\mathbf{y}_i\}_k$ co-evolve toward noise. In each step of the reverse process, the structure is first denoised and subsequently used to denoise the properties (indicated by the green-dashed line). Such de-noised properties are then utilized, in turn, to further denoise the structure (red line), in a process that resembles a *guidance loop*.

[8], which involves training a separate property predictor model alongside the diffusion model; and *classifier-free guidance* [23], which integrates scores from both unconditional and conditional diffusion models. While these guidance techniques have been found to be effective, the algorithm design is not tailored to encompass the intricate hierarchical or multi-resolution elements inherent in conditional generation. Consequently, it is plausible that this inadequacy may contribute to suboptimal representations, particularly notable in tasks such as conditional graph generation. The recent success of hierarchical diffusion flows in various domains, such as modeling interactions between node and edge features [37], multi-resolution modeling [25], decision-making [47], and conditional image generation [4, 71] underscores the need to integrate hierarchical information beyond the capabilities of classifier-based and classifier-free guidance.

We assert that conditional diffusion models for structured spaces, such as graphs, could be enhanced with *hierarchical conditional processes*. Specifically, rather than treating heterogeneous structural and label information uniformly within the hierarchy, we advocate for the co-evolution of multiple processes with *distinct roles* (asymmetric). These roles encompass a primary process governing the structural evolution alongside multiple secondary processes responsible for driving conditional content. We aim to propose an alternative to existing conditional graph diffusion techniques (outlined in Table 1) by bestowing the models with finer control over two key aspects: 1) the evolution of structural graph components, including nodes and edges, and 2) the co-adaptation of the graph structure in conjunction with one or more associated properties.

Towards this objective, we present a novel diffusion framework for conditional generation named `Twigs`, drawing analogies from the trunk and offshoots of a tree. Concretely, we establish a central *trunk* process governing a primary variable, which interacts with several *stem* processes, each associated with a secondary variable. In contrast with classifier-free and classifier-based methodologies, a novel conditional mechanism, termed *loop guidance*, orchestrates information exchange between the trunk and the stem processes (refer to Figure 1). Our methodology facilitates the acquisition of flexible representations, capitalizing on the disentanglement of intricate interactions and dependencies. We formalize our framework by drawing upon the theory of denoising score matching [67] and leveraging tools derived from stochastic differential equations (SDEs) [1]. The effectiveness of `Twigs` is substantiated through compelling empirical validation across various conventional constrained generation tasks, utilizing both molecular and generic graph datasets.

1.1 Contributions

In summary, this paper makes the following key contributions:

Table 1: **Comparison of related methodologies**. `Twigs` is the first method that enables a seamless orchestration of multiple asymmetric property-oriented hierarchical diffusion processes via SDEs.

Method	Conditional	Asymmetric	Multiple flows	Continuous (SDEs)
GDSS [37]	✗	✗	✓	✓
EEGSDE [3]	✓	✗	✗	✓
MOOD [45]	✓	✗	✗	✓
JODO [28]	✓	✗	✗	✓
EDGE [5]	✗	✗	✓	✗
GraphMaker [46]	✓	✗	✓	✗
Nisonoff et al. [52]	✓	✗	✗	✗
Gruver et al. [18]	✓	✗	✗	✓
Klarner et al. [40]	✓	✓	✗	✗
Twigs (ours)	✓	✓	✓	✓

- **(Conceptual and methodological)** The introduction of a new score-based, end-to-end trainable, non-autoregressive generative model `Twigs` designed for acquiring conditional representations. Our approach enables precise guidance of multiple property-conditioned diffusion processes.
- **(Technical)** We present a robust mathematical framework, including a novel strategy called *loop guidance*, that employs tools from Stochastic Differential Equations (SDEs) to derive both the forward diffusion process and its corresponding reverse SDE for conditional generation. This framework is designed to seamlessly integrate additional contexts as conditioning information.
- **(Empirical)** We showcase the versatility of the proposed diffusion mechanism (`Twigs`) through extensive empirical evidence across various challenging conditional graph generation tasks, consistently surpassing contemporary baselines.

2 Related works

In Table 1 we provide an overview of the similarities and differences between `Twigs` and related methods. We refer the reader to Appendix E for additional related work.

Diffusion guidance is typically applied to regulate the diffusion process for conditional generation. Previous approaches that perform class-conditional generation are divided into classifier-based [8], and classifier-free guidance [23]. While some works model diffusion with multiple flows [5, 37, 46], they treat nodes and edges in a symmetric way; i.e., they associate multiple flows for nodes and edges that have equivalent contributions (in other words, these flows have the same roles). We instead abstract graph properties as secondary processes that branch from, and interact with, the main process that pertains to the graph structure. In addition, while other guidance methods are related [18, 40, 52], they do not leverage multiple diffusion flows. To our knowledge, the proposed method is the first to incorporate multiple diffusion flows in a hierarchical fashion for conditional generation. We formalize in Table 2 how `Twigs` differs, mathematically, from classifier-free and classifier-based methods.

Conditional Diffusion for Graphs Recent advancements in generative modeling have prominently featured score-based techniques (SGM), utilizing diffusion or stochastic differential equations (SDEs) [19, 32, 35, 37, 48], including for graph generation [3, 5, 13, 14, 15, 18, 26, 40, 45, 46, 52, 72, 75, 82]. Guidance methods have been adopted in conditional molecule generation settings. The works from Hoogeboom et al. [26], Huang et al. [28, 28], Xu et al. [82] are classifier-free approaches, while Bao et al. [3], Vignac et al. [75], Lee et al. [45] focus on classifier-based methods. Diverging from these approaches, we explicitly model the dynamic interaction between primary variables (e.g., graph structure) and dependent variables (e.g., graph properties) using dedicated diffusion processes to achieve more expressive representations and improve performance for conditional generation.

3 Diffusion Twigs

Method overview We extend score-based techniques [67] for training conditional diffusion models over graphs. Differently from current guidance methods, as summarised in Table 2, we leverage a finer control over the structure and graph properties to diffuse multiple hierarchical processes, toward achieving a more robust representation. Our method, `Twigs`, defines a *trunk* process over the primary

Table 2: `Twigs` comparison to Classifier-based [8] and Classifier-free [23] guidance, applied for conditional generation in Diffusion models. Here $\mathbf{y}_s$ represents the graph structure, $\{\mathbf{y}_i\}_k$ represent the k-properties of graph. The f_ϕ function is the classifier, ϵ_θ and $s_{\theta,\phi}$ are learnable score models.

Method	**Diffusion Scheme**	**Approach**
Class. based	$\mathrm{d}\mathbf{y}_s = \mathbf{f}(\mathbf{y}_{s,t}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$ $d\mathbf{y}_s = [\mathbf{f}(\mathbf{y}_{s,t}, t) - g_t^2 \nabla_{\mathbf{y}_{s,t}} \log p_t(\mathbf{y}_{s,t}, \{\mathbf{y}_i\}_k)]dt + g_t d\bar{\mathbf{w}}$	$\nabla_{\mathbf{y}_{s,t}} \log p\left(\mathbf{y}_{s,t}, \{\mathbf{y}_i\}_k\right) = \nabla_{\mathbf{y}_{s,t}} \log p\left(\mathbf{y}_{s,t}\right) + \nabla_{\mathbf{y}_{s,t}} \log p\left(\{\mathbf{y}_i\}_k \mid \mathbf{y}_{s,t}\right)$ $\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta\left(\mathbf{y}_{s,t}\right) + \nabla_{\mathbf{y}_{s,t}} \log f_\phi\left(\{\mathbf{y}_i\}_k \mid \mathbf{y}_{s,t}\right)$
Class. free	$\mathrm{d}\mathbf{y}_s = \mathbf{f}(\mathbf{y}_{s,t}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$ $d\mathbf{y}_s = [\mathbf{f}(\mathbf{y}_{s,t}, t) - g_t^2 \nabla_{\mathbf{y}_{s,t}} \log p_t(\mathbf{y}_{s,t}, \{\mathbf{y}_i\}_k)]dt + g_t d\bar{\mathbf{w}}$	$\nabla_{\mathbf{y}_{s,t}} \log p\left(\{\mathbf{y}_i\}_k \mid \mathbf{y}_{s,t}\right) = \nabla_{\mathbf{y}_{s,t}} \log p\left(\mathbf{y}_{s,t} \mid \{\mathbf{y}_i\}_k\right) - \nabla_{\mathbf{y}_{s,t}} \log p\left(\mathbf{y}_{s,t}\right)$ $= -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \left(\epsilon_\theta\left(\mathbf{y}_{s,t}, t, \{\mathbf{y}_i\}_k\right) - \epsilon_\theta\left(\mathbf{y}_{s,t}, t\right)\right)$
`Twigs`	$\mathrm{d}\mathbf{y}_s = \mathbf{f}(\mathbf{y}_{s,t}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$, $\{\mathrm{d}\mathbf{y}_i\}_k = \mathbf{f}(\mathbf{y}_{s,t}, \mathbf{y}_{i,t}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$ $d\mathbf{y}_s = [\mathbf{f}(\mathbf{y}_{s,t}, t) - g_t^2 \nabla_{\mathbf{y}_{s,t}} \log p_t(\mathbf{y}_{s,t}, \{\mathbf{y}_{i,t}\}_k)]dt + g_t d\bar{\mathbf{w}}$ $\{\mathrm{d}\mathbf{y}_i\}_k = [\mathbf{f}(\mathbf{y}_{s,t}, \mathbf{y}_{i,t}, t) - g_t^2 \nabla_{\mathbf{y}_{i,t}} \log p_t(\mathbf{y}_{s,t}, \mathbf{y}_{i,t})]dt + g_t d\bar{\mathbf{w}}$	$\nabla_{\mathbf{y}_{s,t}} \log p_t(\mathbf{y}_{s,t}, \{\mathbf{y}_{i,t}\}_k) = \nabla_{\mathbf{y}_{s,t}} \log p_t(\mathbf{y}_{s,t}) + \sum_i \nabla_{\mathbf{y}_{s,t}} \log p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t})$ $\nabla_{\mathbf{y}_{s,t}} \log p_t(\mathbf{y}_{s,t}) \approx s_{\theta,t}(\mathbf{y}_{s,t})$, $\nabla_{\mathbf{y}_{s,t}} \log p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}) \approx s_{\phi,t}(\mathbf{y}_{s,t}, \mathbf{y}_{i,t})$ $\nabla_{\mathbf{y}_{s,t}} \log p_t(\mathbf{y}_{s,t}, \{\mathbf{y}_{i,t}\}_k) = s_{\theta,t}(\mathbf{y}_{s,t}) + \sum_i s_{\phi,t}(\mathbf{y}_{s,t}, \mathbf{y}_{i,t})$

variable (graph structure) $\mathbf{y}_s$, and a *stem* process over each dependent variable $\mathbf{y}_i \in \mathbb{R}$ (e.g., graph property). We achieve the desired flexibility with a variable $\mathbf{y}_s$ that encompasses both node features and the adjacency matrix as well as the coordinates. The details of the dimensions of $\mathbf{y}_s$ are given in Section B.1 for the 3D case, and in Section B.2 for the 2D case.

Forward process We define multiple forward processes within a hierarchy that co-evolves data and properties into noise. The *trunk* forward process for the graph structure $\mathbf{y}_s$ is defined as

$$\mathrm{d}\mathbf{y}_s = \mathbf{f}_s(\mathbf{y}_{s,t}, t)\mathrm{d}t + g_s(t)\mathrm{d}\mathbf{w} \tag{1}$$

where $\mathbf{f}_s$ and g_s are corresponding diffusion and drift functions, and $\mathrm{d}\mathbf{w}$ is the Wiener noise. The *stem* forward process over the k dependent variables $\mathbf{y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ is defined as

$$\mathrm{d}\mathbf{y}(t) = \begin{pmatrix} \mathrm{d}\mathbf{y}_1(t) \\ \vdots \\ \mathrm{d}\mathbf{y}_k(t) \end{pmatrix} = \begin{pmatrix} \mathbf{f}_p(\mathbf{y}_{1,t}, \mathbf{y}_{s,t}, t)\mathrm{d}t + g_p(t)\mathrm{d}\mathbf{w} \\ \vdots \\ \mathbf{f}_p(\mathbf{y}_{k,t}, \mathbf{y}_{s,t}, t)\mathrm{d}t + g_p(t)\mathrm{d}\mathbf{w} \end{pmatrix} \tag{2}$$

Here, $\mathbf{f}_p$ and g_p denote the diffusion and drift functions, respectively, for the k stem processes. Collectively, along with the trunk forward process, they constitute `Twigs`. These operations introduce random Gaussian noise, iteratively, to the data toward a prior (typically Gaussian) distribution.

Reverse Process The `Twigs` reverse process starts from the prior distribution (Gaussian noise) towards the data distribution. A key difference with Song et al. [67] is that here our variable $\mathbf{y}_t$ comprises both structure *and properties*, leading to the following modification of the overall diffusion process:

$$d\mathbf{y}_t = [f(\mathbf{y}_t, t) - g_t^2 \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_t)]dt + g_t d\bar{\mathbf{w}} \qquad \text{where} \quad \mathbf{y}_t = \{\mathbf{y}_{s,t}, \{\mathbf{y}_{i,t}\}_{i=1}^k\}\,. \tag{3}$$

We derive Equation (3) in Section A.1. The joint distribution over the trunk and stem processes is assumed to factorize as

$$p_t(\mathbf{y}_{s,t}, \mathbf{y}_{1,t}, ..., \mathbf{y}_{k,t}) = p_t(\mathbf{y}_{s,t}) \prod_{i=1}^k\ p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t})\,. \tag{4}$$

In turn, the score function simplifies as in Equation (5), leading to the decomposition in Equation (6).

$$\nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{s,t}, \mathbf{y}_{1,t}, \ldots, \mathbf{y}_{k,t}) = \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{s,t}) + \sum_{i=1}^k\ \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}) \tag{5}$$

$$\mathrm{d}\mathbf{y}_t = [\mathbf{f}(\mathbf{y}_t, t) - g_t^2(\nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{s,t}) + \textstyle\sum_{i=1}^k \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}))]\mathrm{d}t + g_t \mathrm{d}\bar{\mathbf{w}} \tag{6}$$

Conditional modeling We expand our proposed approach to enable conditional generation with an external context $\mathbf{y}_C = \{\mathbf{y}_c \mid c \in C\}$, where $C \subseteq \{1, \ldots, k\}$. The context can be represented as a scalar or vector, describing a particular value associated with a data-dependent variable. For example, in case of molecules, it could represent one or more of the k properties such as the Synthetic Accessibility (SA) score or the Quantitative Estimate of Drug likeness (QED). This extension modifies the joint distribution for the score function in Equation (5).

Reverse SDE under conditioning context

The reverse SDE for $\mathbf{y}_t = \{\mathbf{y}_{s,t}, \{\mathbf{y}_{i,t}\}_k\}$ give an external conditioning context $\mathbf{y}_C$ is shown below (details in Appendix A.2).

$$\mathrm{d}\mathbf{y}_t = [\mathbf{f}(\mathbf{y}_t, t) - g_t^2 \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_t, \mathbf{y}_C)]\mathrm{d}t + g_t \mathrm{d}\bar{\mathbf{w}} \tag{7}$$

We resort to the following factorization of the distribution, conditioned on the context $\mathbf{y}_C$:

$$p_t(\mathbf{y}_{s,t}, \{\mathbf{y}_{i,t}\}_k, \mathbf{y}_C) = \prod_i^k p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}, \mathbf{y}_C) p_t(\mathbf{y}_{s,t}, \mathbf{y}_C)$$

As a result, the factorization of the score function $\nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{s,t}, \{\mathbf{y}_{i,t}\}_k, \mathbf{y}_C)$ amounts to

$$\nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{s,t}, \mathbf{y}_C) + \sum_{i \notin C}^k \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}) + \sum_c^C \sum_i^k \delta_{i=c} \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}, \mathbf{y}_c) \tag{8}$$

The above-factorized score function parameterizes our reverse diffusion process, thus offering a novel approach to integrate external contextual information into conditional generation.

Training We propose to train `Twigs` by incorporating the factorization from Equation (8) within a score-matching objective function [30, 67]. Algorithm 1 shows the training procedure to learn two types of time-dependent score-based models: $\boldsymbol{s}_{\theta,t}$, which approximates the trunk variable, and $\boldsymbol{s}_{\phi_i,t}$ which approximates the coupling between the stem variable and the trunk variable. The objective function for optimizing the score networks $\boldsymbol{s}_\theta, \boldsymbol{s}_{\phi_i}$, is given as follows:

$$\min_{\theta,\phi_i} \mathbb{E}_t \{\lambda_{\mathbf{y}_t}(t) \mathbb{E}_{\mathbf{y}_0} \mathbb{E}_{\mathbf{y}_t|\mathbf{y}_0} \| \boldsymbol{s}_{\theta,t}(\mathbf{y}_{s,t}, \mathbf{y}_c) + \sum_i^k \boldsymbol{s}_{\phi_i,t}(\mathbf{y}_{i,t}, \mathbf{y}_{s,t}, \mathbf{y}_c) - \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_t, \mathbf{y}_C) \|_2^2\} \tag{9}$$

where $\mathbb{E}_{\mathbf{y}_0} = \mathbb{E}_{\mathbf{y}_{s,0}, \mathbf{y}_{i,0}}$ and $\mathbb{E}_{\mathbf{y}_t} = \mathbb{E}_{\mathbf{y}_{s,t}, \mathbf{y}_{i,t}}$. It is worth noting that the influence introduced by the variable s_{ϕ_i} provides the directions for the diffusion model to converge into distributions with the desired properties. Such property-oriented knowledge operates in conjunction with the structural information provided by $\boldsymbol{s}_\theta$, resulting in a novel form of guidance that is orchestrated by a branching diffusion process, named *Loop guidance*.

Algorithm 1 Training `Twigs`

Input: Dataset $\mathcal{D}$, iterations n_{iter}, batch size B, number of batches n_B, K properties to consider
Initialize parameters $s_{\theta,t}, \{s_{\phi_i,t}\}_{i=1}^K$ for Score Networks
for $k = 1, \ldots, n_{\text{iter}}$ **do**
  for $b = 1, \ldots, n_B$ **do**
    $t \sim \mathcal{U}(0, 1]$
    $\mathcal{D}_b = \{(\boldsymbol{y}_{s,l}, \{\boldsymbol{y}_{i,l}\}_{i=1}^K)_{l=1}^B, \boldsymbol{y}_C\} \sim \mathcal{D}$
    $\mathcal{L}_b \leftarrow$ Eq. 9
  end for
  $\theta, \{\phi_i\}_{i=i}^K \leftarrow \texttt{optim}(\frac{1}{n_B} \sum_{b=1}^{n_B} \mathcal{L}_b)$
end for

Algorithm 2 Generating with `Twigs`

Input: Score-based models $s_{\theta,t}, \{s_{\phi_i,t}\}_{i=1}^K$, Time step schedule $\{t\}_{t=T}^0$, Langevin MCMC step size α, External context $\boldsymbol{y}_C$

$\boldsymbol{y}_{s_T}, \{\boldsymbol{y}_{i,T}\}_{i=1}^K \sim \mathcal{N}(0, I)$
for $t = T, \ldots, 0$ **do**
  $s_{\theta,t} \leftarrow s_{\theta,t}(\boldsymbol{y}_{s_t}, \{\boldsymbol{y}_{i,t}\}_{i=1}^K, \boldsymbol{y}_C)$
  $\{s_{\phi_i,t}\}_{i=1}^K \leftarrow \{s_{\phi_i,t}(\boldsymbol{y}_{s_t}, \boldsymbol{y}_{i,t}, \boldsymbol{y}_C)\}_{i=1}^K$
  $\mathbf{y}_{s_t} \leftarrow \mathbf{y}_{s_t} + \frac{\alpha}{2} s_{\theta,t} + \sqrt{\alpha} z_s;\ z_s \sim \mathcal{N}(0, I)$
  $\mathbf{y}_{i_t} \leftarrow \mathbf{y}_{i_t} + \frac{\alpha}{2} s_{\phi_i,t} + \sqrt{\alpha} z_i;\ z_i \sim \mathcal{N}(0, I)$
end for

Sampling Given a trained conditional `Twigs` model, our generative process begins by sampling an external context or conditioning value $\mathbf{y}_C$, which can also be supplied externally. We then simulate the reverse diffusion process, similar to the one described in Equation 8, but with a modified score function to generate the data. The proposed algorithm for generating new data samples with `Twigs` is given in Algorithm 2 and involves a loop of updates between processes: the stem score network s_{ϕ_i} evolves the property $\mathbf{y}_i$, integrating information from the structure $\mathbf{y}_s$, and subsequently, the updated property information from s_{ϕ_i} is integrated into the main process by the score network s_θ.

4 Experiments

We conduct a set of comprehensive experiments to demonstrate that `Twigs` improves over contemporary conditional generation methods. Benchmarks include: molecule generation conditioned over single (§ 4.1), and multiple (§ 4.2) properties on QM9, as well as molecule optimization on ZINC250K (§ 4.3), and network-graph generation conditioned on desired properties (§ 4.4).

Figure 2: First row: Samples by `Twigs` for 3D molecules conditioned on single properties on QM9. Second row: KDE and KL divergence results between target and predicted properties.

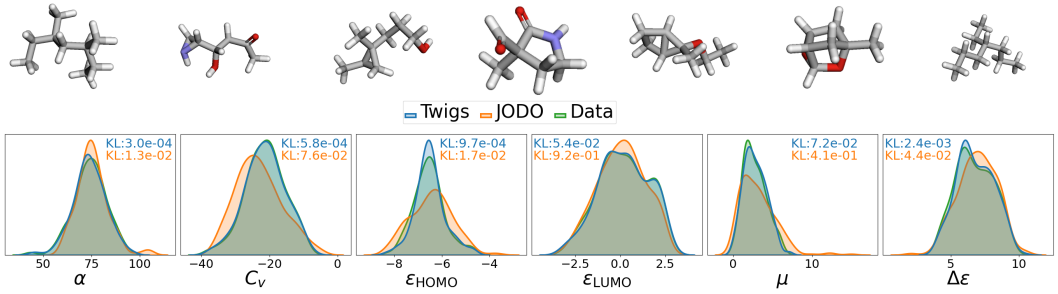

Table 3: MAE↓ results on single target quantum property for the QM9 dataset.

Method	C_v	μ	α	$\Delta\epsilon$	ϵ_{HOMO}	ϵ_{LUMO}
EDM	1.065 (± 0.010)	1.123 (± 0.013)	2.78 (± 0.04)	671 (± 5)	371 (± 2)	601 (± 7)
GeoLDM	1.025 (± na)	1.108 (± na)	2.37 (± na)	587 (± na)	340 (± na)	522 (± na)
EEGSDE	0.941 (± 0.005)	0.777 (± 0.007)	2.50 (± 0.02)	487 (± 3)	302 (± 2)	447 (± 6)
EquiFM	1.033 (± na)	1.106 (± na)	2.41 (± na)	591 (± na)	337 (± na)	530(± na)
TEDMol	0.847 (± na)	0.840 (± na)	2.24 (± na)	443 (± na)	279 (± na)	412 (± na)
JODO	0.581 (± 0.001)	0.628 (± 0.003)	1.42 (± 0.01)	335 (± 3)	226 (± 1)	256 (± 1)
`Twigs`	**0.559** (± 0.002)	**0.627** (± 0.001)	**1.36** (± 0.01)	**323** (± 2)	**225** (± 1)	**244** (± 3)

Table 4: Novelty, atom & molecule stability for QM9 single property.

	Novelty↑	Atom Stability↑	Mol Stability↑	Novelty↑	Atom Stability↑	Mol Stability↑
		C_v			μ	
EDM	83.64(± 0.30)	98.25(± 0.02)	80.82(± 0.32)	83.93(± 0.11)	98.17(± 0.04)	80.25(± 0.40)
EEGSDE	83.53(± 0.18)	98.25(± 0.06)	80.83(± 0.33)	83.85(± 0.20)	98.18(± 0.02)	80.25(± 0.18)
TEDMol	83.82(± na)	98.27(± na)	80.83(± na)	84.88(± na)	98.22(± na)	80.31(± na)
JODO	91.21(± 0.22)	97.74(± 0.29)	91.75(± 0.11)	91.22(± 0.02)	99.02(± 0.02)	92.86(± 0.15)
Twigs	**93.16**(± 0.16)	**99.14**(± 0.04)	**92.72**(± 0.07)	**92.90**(± 0.08)	**99.25**(± 0.05)	**93.91**(± 0.03)
		$\Delta\varepsilon$			$\varepsilon_{\text{HOMO}}$	
EDM	83.93(± 0.45)	98.30(± 0.04)	81.95(± 0.27)	84.35(± 0.31)	98.17(± 0.07)	79.61(± 0.32)
EEGSDE	84.09(± 0.27)	98.18(± 0.06)	80.99(± 0.29)	84.44(± 0.33)	98.19(± 0.03)	79.81(± 0.20)
TEDMol	84.92(± na)	98.19(± na)	79.82(± na)	84.58(± na)	98.22(± na)	80.97(± na)
JODO	91.02(± 0.19)	98.42(± 0.02)	93.32(± 0.04)	91.38(± 0.02)	98.19(± 0.38)	92.02(± 0.03)
Twigs	**92.70**(± 0.04)	**99.31**(± 0.01)	**94.12**(± 0.31)	**93.02**(± 0.21)	**99.26**(± 0.04)	**94.11**(± 0.26)
		α			$\varepsilon_{\text{LUMO}}$	
EDM	84.56(± 0.47)	98.13(± 0.04)	79.33(± 0.30)	84.62(± 0.28)	98.26(± 0.04)	81.34(± 0.29)
EEGSDE	84.19(± 0.32)	98.26(± 0.03)	80.95(± 0.35)	84.83(± 0.30)	98.14(± 0.01)	80.00(± 0.21)
TEDMol	85.82(± na)	98.42(± na)	82.03(± na)	84.90(± na)	98.31(± na)	81.40(± na)
JODO	90.15(± 0.02)	98.74(± 0.05)	94.03(± 0.32)	90.78(± 0.42)	98.84(± 0.04)	94.02(± 0.03)
Twigs	**92.88**(± 0.13)	**99.28**(± 0.12)	**94.12**(± 0.02)	**92.48**(± 0.15)	**99.29**(± 0.17)	**94.11**(± 0.33)

4.1 Single Quantum properties on QM9

Setup. We evaluate the effectiveness of `Twigs` for generating molecules with a single desired quantum property, sourced from the QM9 dataset [58], specifically, we consider C_v, μ, α, $\Delta\epsilon$, ϵ_{LUMO} and ϵ_{HOMO}. To ensure consistency and comparability with the baselines, which include JODO [28], EDM [26], EEGSDE [3], GeoLDM [82], TEDMol [49], EquiFM [68], we adhere to the identical dataset preprocessing, training/test data partitions, and evaluation metrics outlined by Huang et al. [28]. Regarding parameterization of `Twigs`, we follow the attention architecture defined in Section B.1 with a single stem process.

Results. In Table 3, we report the Mean Absolute Error (MAE) results, and in Table 4, the Novelty, Atom stability and Molecule stability. Our method outperforms all the evaluated baselines across the specified properties. In Figure 2, the bottom row provides a Kernel Density Estimation (KDE) visualization which shows that `Twigs` achieves a more accurate distribution for the property values when compared with JODO, while the top row shows some 3D molecule samples by our model.

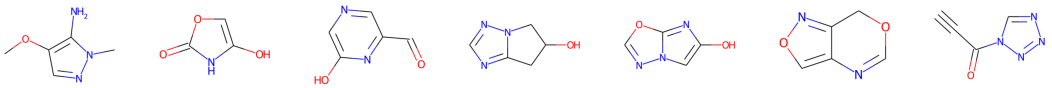

Figure 3: Samples of multiple-property conditional molecules by `Twigs` (C_v and μ) for QM9.

Table 5: MAE (↓) for conditional generation on QM9 with multiple properties.

	C_v	μ	$\Delta\epsilon$	μ	α	μ
EDM	1.097(± 0.007)	1.156(± 0.011)	683(± 1)	1.130(± 0.007)	2.76(± 0.01)	1.158(± 0.002)
EEGSDE	0.981(± 0.008)	0.912(± 0.006)	563(± 3)	0.866(± 0.003)	2.61(± 0.01)	0.855(± 0.007)
TEDMol	0.645(± n/a)	0.836(± n/a)	489(± n/a)	0.843(± n/a)	2.27(± n/a)	0.809(± n/a)
JODO	0.634(± 0.002)	0.716(± 0.006)	350(± 4)	0.752(± 0.006)	1.52(± 0.01)	0.717(± 0.006)
`Twigs`	**0.602**(± 0.001)	**0.708**(± 0.002)	**343**(± 2)	**0.740**(± 0.003)	**1.46**(± 0.01)	**0.712**(± 0.002)

4.2 Multiple Quantum properties on QM9

Setup. This experiment evaluates the capability to combine multiple desired properties in the generated molecule. Specifically we follow Huang et al. [29] and consider all possible combinations of properties involving μ: (C_v, μ), $(\Delta\epsilon, \mu)$, (α, μ). Since we model two properties, we test our `Twigs` with two stem networks within the attention architecture described in Section B.1. We benchmark against several contemporary baselines, including EDM [26], EEGSDE [3] and JODO [28].

Results. In Table 5, we present the Mean Absolute Error (MAE) results obtained from the property predictors introduced by Huang et al. [28] for the various property pairs under consideration. The superior performance of `Twigs` across all baselines reinforces the findings from the single property experiment (Section 4.1), emphasizing the benefits of learning multiple hierarchical stem processes.

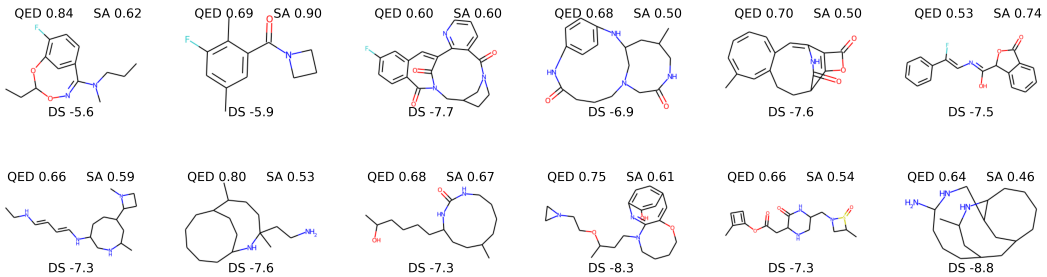

Figure 4: Molecules generated by `Twigs` from ZINC250k conditioned on fa7 (top), parp1 (bottom).

4.3 Molecule optimization on ZINC250K

Setup. The goal is to generate molecules from the ZINC250K dataset that exhibit optimal binding affinity, drug-likeness, and synthesizability for the following five target proteins: *parp1, fa7, 5ht1b, braf, jak2*. We adhere to the evaluation protocol established by Lee et al. [45], which involves generating 3000 molecules and assessing them using two metrics that constrain the desired properties, including docking score (DS), drug-likeness (QED), and synthetic accessibility (SA).

The first metric, *Novel hit ratio (%)*, represents the fraction of unique *hit molecules* that have a maximum Tanimoto similarity of less than 0.4 with the training molecules. Hit molecules are defined as those meeting the criteria: DS < (the median DS of the known active molecules), QED > 0.5, and SA < 5. The second metric, *Novel top 5% docking score*, is the average DS of the top 5% unique molecules that satisfy QED > 0.5 and SA < 5, with a maximum similarity of less than 0.4 to the training molecules.

Baselines. We consider REINVENT [55]: a reinforcement learning (RL) model that utilizes a prior sequence model, MORLD [33]: a RL model that uses QED and SA scores as intermediate rewards and docking scores as final rewards, HierVAE [34]: a VAE-based model that utilizes hierarchical molecular representation and active learning, GDSS [37]: a score-based diffusion model that evolves nodes and edge information with a system of SDEs, MOOD [45]: a score-based diffusion model based on GDSS that trains an additional property predictor to improve conditional generation. For MOOD we consider the version without the out-of-distribution (OOD) control, to have a fair comparison with our method. For `Twigs` we follow the GCN-based architecture described in Section B.2, with multiple stem processes (one for each target protein).

Results. In Table 6 we report the results for top 5% docking scores. We observe that `Twigs` achieves the highest score across all properties, excluding braf, where it achieves the second-best score after MOOD. In Table 7 we report the results for Novel hit ratio. The outcomes confirm that our model is improving the performance substantially over all the considered properties, except for braf, on which `Twigs` is the second-best performing model after MOOD. In Figure 4, we provide some samples of the molecules obtained by `Twigs` with the respective QED, SA, and docking score. Additionally, in Table 13 we report the MAE values for generating molecules with a desired target protein property, and in Table 14 we compare the inference cost of `Twigs` against MOOD.

Table 6: Novel top 5% docking score on ZINC250K. Best is **boldfaced**, second-best is in gray.

Model	*parp1*	*fa7*	*5ht1b*	*braf*	*jak2*
REINVENT	8.702(± 0.523)	7.205(± 0.264)	8.770(± 0.316)	8.392(± 0.400)	8.165(± 0.277)
MORLD	7.532(± 0.260)	6.263(± 0.165)	7.869(± 0.650)	8.040(± 0.337)	7.816(± 0.133)
HierVAE	9.487(± 0.278)	6.812(± 0.274)	8.081(± 0.252)	8.978(± 0.525)	8.285(± 0.370)
GDSS	9.967(± 0.028)	7.775(± 0.039)	9.459(± 0.101)	9.224(± 0.068)	8.926(± 0.089)
MOOD	10.409(± 0.030)	7.947(± 0.034)	10.487(± 0.069)	**10.421**(± 0.050)	9.575(± 0.075)
Twigs	**10.449**(± 0.009)	**8.182**(± 0.012)	**10.542**(± 0.025)	10.343(± 0.024)	**9.678**(± 0.032)

Table 7: Novel hit ratio (↑) results on ZINC250K.

Model	*parp1*	*fa7*	*5ht1b*	*braf*	*jak2*
REINVENT	0.480(± 0.344)	0.213(± 0.081)	2.453(± 0.561)	0.127(± 0.088)	0.613(± 0.167)
MORLD	0.047(± 0.050)	0.007(± 0.013)	0.880(± 0.735)	0.047(± 0.040)	0.227(± 0.118)
HierVAE	0.553(± 0.214)	0.007(± 0.013)	0.507(± 0.278)	0.207(± 0.220)	0.227(± 0.127)
GDSS	1.933(± 0.208)	0.368(± 0.103)	4.667(± 0.306)	0.167(± 0.134)	1.167(± 0.281)
MOOD	3.400(± 0.117)	0.433(± 0.063)	11.873(± 0.521)	**2.207**(± 0.165)	3.953(± 0.383)
Twigs	**3.733**(± 0.081)	**0.900**(± 0.012)	**16.366**(± 0.029)	1.933(± 0.023)	**5.100**(± 0.312)

4.4 Generation of Network graphs with desired properties

Setup. We follow the data processing delineated by Jo et al. [37] and provide results for the Community-small [60] and Enzymes datasets [62]. To test the capabilities to generate conditional graphs, we extract four properties via the NetworkX library [20], including density, clustering, assortativity, and transitivity. Considering a graph G with n nodes and m edges, we have: (1) Density: $d = \frac{2m}{n(n-1)}$, (2) Clustering coefficient: the average $C = \frac{1}{n}\sum_{v \in G} c_v$. (3) Assortativity: measures the similarity of connections in the graph with respect to the node degree. (4) Transitivity: the fraction of all possible triangles present in G. Possible triangles are identified by the number of "triads" (two edges with a shared vertex). The transitivity is $T = 3\frac{\#triangles}{\#triads}$.

Baselines. In terms of baselines, we first consider two versions of MOOD [45] (two OOD coefficients), and we train the property predictors using the codes from the authors. Our second baseline is GDSS [37], which we modify to be equipped with a classifier-free guidance scheme. We also consider the version of GDSS based on transformers, which leverages the graph-multi-head attention [2]. Finally, we consider Digress [75], which is a classifier-based guidance diffusion model based on attention mechanisms. We parameterize our `Twigs` model with our GCN architecture described in Section B.2, with a single stem process.

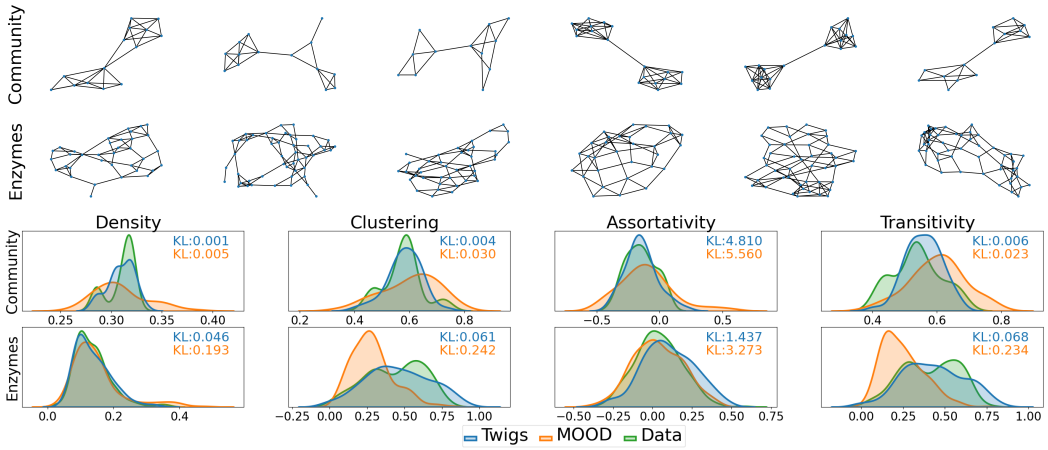

Figure 5: Visualization of Community-small and Enzymes datasets. First and second rows: samples generated by `Twigs`. Third and fourth rows: KDE plots and corresponding KL divergence values.

Table 8: MAE (↓) values on Community-small and Enzymes, conditioned on single properties.

	Community Small				Enzymes			
Model	Density	Clustering	Assortativity	Transitivity	Density	Clustering	Assortativity	Transitivity
GDSS	2.95	12.1	19.6	11.4	8.04	2.53	1.98	2.55
GDSS-T	2.30	11.5	19.2	10.1	9.25	3.27	2.03	2.68
Digress	2.34	10.6	17.8	9.42	8.04	2.39	1.95	2.55
MOOD-1	2.35	11.1	18.8	10.5	7.94	2.34	1.83	2.12
MOOD-4	2.12	11.3	16.7	8.76	7.98	2.44	1.99	2.43
`Twigs`	**2.07**	**9.67**	**15.2**	**8.35**	**7.35**	**2.23**	**1.72**	**2.03**

Results. Table 8 reports the MAE average of three runs, demonstrating that `Twigs` consistently outperforms the considered baselines on all cases across the two datasets. MOOD is the second-best performing model in the majority of the cases. We further strengthen the MAE results by providing in Figure 5 (bottom) the KDE plots of the property distributions of the graph generated by `Twigs` and MOOD. The Figure demonstrates that `Twigs` can achieve a higher fidelity to the data, which is also confirmed by the lower KL divergence values. Figure 5 (top) depicts some random graph samples generated by `Twigs`.

4.5 Ablation study on multiple properties

Setup. Assuming conditional independence among the properties α, ϵ_{HOMO}, ϵ_{LUMO}, $\Delta\epsilon$, μ, and C_v given the molecular graph can simplify the modeling process. This assumption leverages the fact that the molecular graph captures the essential structural dependencies, allowing us to treat the properties as independent for computational efficiency and ease of interpretation, even if slight interdependencies exist.

Results. Here we show that such modeling assumption can work practically. Table 9 reports the MAE on molecular graphs for QM9 on three properties, showing that our method consistently achieves lower error on all the properties. Table 10 shows that on generic graphs `Twigs` can achieve lower MAE on all the considered cases, in the cases of two and three properties.

Table 9: MAE values over three properties for QM9.

Model	α	μ	$\Delta\epsilon$
JODO	2.749 (± 0.03)	1.162 (± 0.04)	717 (± 5)
Twigs	**2.544** (± 0.05)	**1.094** (± 0.02)	**640** (± 3)

Table 10: MAE results for two and three properties on community small.

Model	Pair1		Pair2		Triplet		
	Density	Clustering	Density	Assortativity	Density	Clustering	Assortativity
GDSS	2.95	13.3	2.61	19.8	2.97	12.5	19.4
Digress	2.82	12.1	2.52	18.1	2.65	11.2	18.2
MOOD	2.43	12.0	2.40	17.2	2.53	11.4	17.3
`Twigs`	**2.34**	**11.0**	**2.39**	**16.7**	**2.27**	**10.6**	**16.1**

4.6 Training time

In Table 11 we study the impact of multiple diffusion flows on the community-small and Enzymes datasets. Specifically, we report the average time for the overall training for `Twigs` with one and three secondary diffusion flows. We observe that our models encounter a small overhead compared to GDSS and Digress, however, we believe it is a good tradeoff because it achieves a lower MAE.

Table 11: Overall training time for 5,000 epochs (hours and minutes) for `Twigs` with different secondary diffusion flows, GDSS, and Digress on the Community-small and Enzymes datasets.

Dataset	`Twigs` $p = 1$	`Twigs` $p = 3$	GDSS	Digress
Community-small	0h 22m	0h 24m	0h 19m	0h 20m
Enzymes	6h 45m	6h 59m	6h 42m	6h 43m

5 Conclusion, Broader Implications, and Limitations

We introduced a novel approach to model conditional information within generative models tailored for graph data. `Twigs` incorporates the novel mechanism of *loop guidance* to control the overall generative process by first bifurcating the diffusion flow into multiple stem processes and then re-integrating them into the trunk process, resembling a loop. Our experimental results showcase the performance gains of `Twigs` when compared to current state-of-the-art baselines across various conditional graph generation tasks.

Conditional generation is fast emerging as one of the most exciting avenues within machine learning and would benefit from techniques beyond classifier-based and classifier-free schemes, making our method applicable to settings beyond this work. Indeed, while the current work has focused on graph settings, `Twigs` might find use in other domains (e.g., image, text, and audio). However, whether `Twigs` is effective in such settings needs to be investigated in future works.

Training multiple properties (stem processes) might require training additional parameters, incurring additional computation and training time. Our ablation study on training time due to multiple processes (Section 4.6) suggests that `Twigs` could provide a good tradeoff (lower MAE compared to some prominent existing methods at the expense of small additional computational overhead).

Finally, assuming factorization of the distribution over stem processes conditioned on the trunk process might not always be realistic. Our experiments in Section 4.5 suggest that `Twigs` might still be able to achieve a strong performance when considering multiple properties. In case some prior knowledge is available about some properties that violate this assumption, we could, in principle, adapt `Twigs` by grouping them into a single stem process while factorizing with the remaining ones.

Acknowledgments

YV and VG acknowledge support from the Research Council of Finland for the "Human-steered next-generation machine learning for reviving drug design" project (grant decision 342077). VG also acknowledges Jane and Aatos Erkko Foundation (grant 7001703) for "Biodesign: Use of artificial intelligence in enzyme design for synthetic biology". GM acknowledges support from the Engineering and Physical Sciences Research Council (EPSRC) and the BBC under iCASE. AF is partially funded by the CRUK National Biomarker Centre, by the Manchester Experimental Cancer Medicine Centre and the NIHR Manchester Biomedical Research Centre.

References

[1] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. ISSN 0304-4149. doi: https://doi.org/10.1016/0304-4149(82)90051-5. URL https://www.sciencedirect.com/science/article/pii/0304414982900515.

[2] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=JHcqXGaqiGn.

[3] Fan Bao, Min Zhao, Zhongkai Hao, Peiyao Li, Chongxuan Li, and Jun Zhu. Equivariant energy-guided SDE for inverse molecular design. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=r0otLtOwYW.

[4] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation. *arXiv preprint arXiv:2302.08113*, 2023.

[5] Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *International Conference on Machine Learning*, pages 4585–4610. PMLR, 2023.

[6] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. *arXiv preprint arXiv:2210.01776*, 2022.

[7] Alex O Davies, Nirav S Ajmeri, et al. Hierarchical gnns for large graph generation. *arXiv preprint arXiv:2306.11412*, 2023.

[8] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[9] Anh-Dung Dinh, Daochang Liu, and Chang Xu. Rethinking conditional diffusion sampling with progressive guidance. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=gThGBHhqcU.

[10] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International Conference on Machine Learning*, pages 8489–8510. PMLR, 2023.

[11] Alexandru Dumitrescu, Dani Korpela, Markus Heinonen, Yogesh Verma, Valerii Iakovlev, Vikas Garg, and Harri Lähdesmäki. Field-based molecule generation. *arXiv preprint arXiv:2402.15864*, 2024.

[12] Peter Eckmann, Kunyang Sun, Bo Zhao, Mudong Feng, Michael K Gilson, and Rose Yu. Limo: Latent inceptionism for targeted molecule generation. *arXiv preprint arXiv:2206.09010*, 2022.

[13] Niklas Gebauer, Michael Gastegger, and Kristof Schütt. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. *Advances in neural information processing systems*, 32, 2019.

[14] Niklas WA Gebauer, Michael Gastegger, Stefaan SP Hessmann, Klaus-Robert Müller, and Kristof T Schütt. Inverse design of 3d molecular structures with conditional generative neural networks. *Nature communications*, 13(1):973, 2022.

[15] Zijie Geng, Shufang Xie, Yingce Xia, Lijun Wu, Tao Qin, Jie Wang, Yongdong Zhang, Feng Wu, and Tie-Yan Liu. De novo molecular generation via connection-aware motif mining. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Q_Jexl8-qDi.

[16] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[17] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 35:14715–14728, 2022.

[18] Nate Gruver, Samuel Stanton, Nathan Frey, Tim GJ Rudner, Isidro Hotzel, Julien Lafrance-Vanasse, Arvind Rajpal, Kyunghyun Cho, and Andrew G Wilson. Protein design with guided discrete diffusion. *Advances in neural information processing systems*, 36, 2023.

[19] Florentin Guth, Simon Coste, Valentin De Bortoli, and Stephane Mallat. Wavelet score-based generative modeling, 2022.

[20] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[21] Philip J Hajduk and Jonathan Greer. A decade of fragment-based drug design: strategic advances and lessons learned. *Nature reviews Drug discovery*, 6(3):211–219, 2007.

[22] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. *arXiv preprint arXiv:2311.16424*, 2023.

[23] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020.

[25] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *The Journal of Machine Learning Research*, 23(1):2249–2281, 2022.

[26] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pages 8867–8887. PMLR, 2022.

[27] Han Huang, Leilei Sun, Bowen Du, Yanjie Fu, and Weifeng Lv. Graphgdp: Generative diffusion processes for permutation invariant graph generation. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 201–210. IEEE, 2022.

[28] Han Huang, Leilei Sun, Bowen Du, and Weifeng Lv. Learning joint 2d & 3d diffusion models for complete molecule generation. *arXiv preprint arXiv:2305.12347*, 2023.

[29] Yinan Huang, Xingang Peng, Jianzhu Ma, and Muhan Zhang. 3dlinker: an e (3) equivariant variational autoencoder for molecular linker design. *arXiv preprint arXiv:2205.07309*, 2022.

[30] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[31] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/f3a4ff4839c56a5f460c88cce3666a2b-Paper.pdf`.

[32] John Ingraham, Max Baranov, Zak Costello, Vincent Frappier, Ahmed Ismail, Shan Tie, Wujie Wang, Vincent Xue, Fritz Obermeyer, Andrew Beam, et al. Illuminating protein space with a programmable generative model. *BioRxiv*, pages 2022–12, 2022.

[33] Woosung Jeon and Dongsup Kim. Autonomous molecule generation using reinforcement learning and docking to develop potential novel inhibitors. *Scientific reports*, 10(1):22104, 2020.

[34] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pages 4839–4848. PMLR, 2020.

[35] Bowen Jing, Gabriele Corso, Renato Berlinghieri, and Tommi Jaakkola. Subspace diffusion generative models. In *Lecture Notes in Computer Science*, pages 274–289. Springer Nature Switzerland, 2022. doi: 10.1007/978-3-031-20050-2_17. URL https://doi.org/10.1007%2F978-3-031-20050-2_17.

[36] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. *Advances in Neural Information Processing Systems*, 35:24240–24253, 2022.

[37] 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*, pages 10362–10383. PMLR, 2022.

[38] Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with destination-driven diffusion mixture. *arXiv preprint arXiv:2302.03596*, 2023.

[39] Kisuk Kang, Ying Shirley Meng, Julien Breger, Clare P Grey, and Gerbrand Ceder. Electrodes with high power and high capacity for rechargeable lithium batteries. *Science*, 311(5763): 977–980, 2006.

[40] Leo Klarner, Tim GJ Rudner, Garrett M Morris, Charlotte M Deane, and Yee Whye Teh. Context-guided diffusion for out-of-distribution molecular and protein design. *arXiv preprint arXiv:2407.11942*, 2024.

[41] 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*, pages 17391–17408. PMLR, 2023.

[42] Xiangzhe Kong, Wenbing Huang, Zhixing Tan, and Yang Liu. Molecule generation by principal subgraph mining and assembling. *Advances in Neural Information Processing Systems*, 35: 2550–2563, 2022.

[43] Najwa Laabid, Severi Rissanen, Markus Heinonen, Arno Solin, and Vikas Garg. Alignment is key for applying diffusion models to retrosynthesis. *arXiv preprint arXiv:2405.17656*, 2024.

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[45] Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[46] Mufei Li, Eleonora Kreačić, Vamsi K Potluru, and Pan Li. Graphmaker: Can diffusion models generate large attributed graphs? *arXiv preprint arXiv:2310.13833*, 2023.

[47] Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, pages 20035–20064. PMLR, 2023.

[48] Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. Graphebm: Molecular graph generation with energy-based models. *arXiv preprint arXiv:2102.00546*, 2021.

[49] Yanchen Luo, Sihang Li, Zhiyuan Liu, Jiancan Wu, Zhengyi Yang, Xiangnan He, Xiang Wang, and Qi Tian. Text-guided diffusion model for 3d molecule generation, 2024. URL https://openreview.net/forum?id=FdUloEgBSE.

[50] Lukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, Tomasz Danel, and Michał Warchoł. Mol-cyclegan: a generative model for molecular optimization. *Journal of Cheminformatics*, 12(1):1–18, 2020.

[51] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

[52] Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv preprint arXiv:2406.01572*, 2024.

[53] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020.

[54] Juhwan Noh, Dae-Woong Jeong, Kiyoung Kim, Sehui Han, Moontae Lee, Honglak Lee, and Yousung Jung. Path-aware and structure-preserving generation of synthetically accessible molecules. In *International Conference on Machine Learning*, pages 16952–16968. PMLR, 2022.

[55] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

[56] Yidong Ouyang, Liyan Xie, and Guang Cheng. Improving adversarial robustness through the contrastive-guided diffusion process. In *International Conference on Machine Learning*, pages 26699–26723. PMLR, 2023.

[57] Bo Qiang, Yuxuan Song, Minkai Xu, Jingjing Gong, Bowen Gao, Hao Zhou, Wei-Ying Ma, and Yanyan Lan. Coarse-to-fine: a hierarchical diffusion model for molecule generation in 3d. In *International Conference on Machine Learning*, pages 28277–28299. PMLR, 2023.

[58] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

[59] Arne Schneuing, Yuanqi Du, Charles Harris, Arian Jamasb, Ilia Igashov, Weitao Du, Tom Blundell, Pietro Lió, Carla Gomes, Max Welling, et al. Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695*, 2022.

[60] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. Brenda, the enzyme database: updates and major new developments. *Nucleic acids research*, 32(suppl_1):D431–D433, 2004.

[61] Daniel Schwalbe-Koda and Rafael Gómez-Bombarelli. Generative models for automatic chemical design. *Machine Learning Meets Quantum Physics*, pages 445–467, 2020.

[62] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[63] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In *International conference on machine learning*, pages 9558–9568. PMLR, 2021.

[64] Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Edward W Lowe. Computational methods in drug discovery. *Pharmacological reviews*, 66(1):334–395, 2014.

[65] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=9_gsMA8MRKQ`.

[66] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pages 32483–32498. PMLR, 2023.

[67] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[68] Yuxuan Song, Jingjing Gong, Minkai Xu, Ziyao Cao, Yanyan Lan, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Equivariant flow matching with hybrid probability transport for 3d molecule generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=hHUZ5V9XFu`.

[69] Brian L Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*, 2022.

[70] Brian L. Trippe, Jason Yim, Doug Tischer, David Baker, Tamara Broderick, Regina Barzilay, and Tommi S. Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6TxBxqNME1Y.

[71] Alex M Tseng, Tommaso Biancalani, Max Shen, and Gabriele Scalia. Hierarchically branched diffusion models for efficient and interpretable multi-class conditional generation. *arXiv preprint arXiv:2212.10777*, 2022.

[72] Alex M Tseng, Nathaniel Diamant, Tommaso Biancalani, and Gabriele Scalia. Graphguide: interpretable and controllable conditional graph generation with discrete bernoulli diffusion. *arXiv preprint arXiv:2302.03790*, 2023.

[73] Yogesh Verma, Samuel Kaski, Markus Heinonen, and Vikas Garg. Modular flows: Differential molecular generation. In *Advances in Neural Information Processing Systems*, volume 35, pages 12409–12421. Curran Associates, Inc., 2022.

[74] Yogesh Verma, Markus Heinonen, and Vikas Garg. Abode: Ab initio antibody design using conjoined odes. In *International Conference on Machine Learning*, pages 35037–35050. PMLR, 2023.

[75] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

[76] Wujie Wang, Minkai Xu, Chen Cai, Benjamin Kurt Miller, Tess Smidt, Yusu Wang, Jian Tang, and Rafael Gómez-Bombarelli. Generative coarse-graining of molecular conformations. *arXiv preprint arXiv:2201.12176*, 2022.

[77] Zichao Wang, Weili Nie, Zhuoran Qiao, Chaowei Xiao, Richard Baraniuk, and Anima Anandkumar. Retrieval-based controllable molecule generation. *arXiv preprint arXiv:2208.11126*, 2022.

[78] Fang Wu and Stan Z Li. Diffmd: a geometric diffusion model for molecular dynamics simulations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5321–5329, 2023.

[79] Kevin Eric Wu, Kevin K Yang, Rianne van den Berg, James Zou, Alex Xijie Lu, and Ava P Amini. Protein structure generation via folding diffusion, 2023. URL https://openreview.net/forum?id=Nkd7AS2USRd.

[80] Lemeng Wu, Chengyue Gong, Xingchao Liu, Mao Ye, and Qiang Liu. Diffusion-based molecule generation with informative prior bridges. *Advances in Neural Information Processing Systems*, 35:36533–36545, 2022.

[81] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PzcvxEMzvQC.

[82] Minkai Xu, Alexander S Powers, Ron O Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pages 38592–38610. PMLR, 2023.

A Proofs

A.1 Derivation of the reverse SDE

For a Stochastic Differential Equation (SDE) of the form,

$$dx = f(x_t, t)\mathrm{d}t + g(x_t, t)\mathrm{d}\mathbf{w} \tag{10}$$

where $f(\cdot)$ and $g(\cdot)$ are diffusion, drift function and $\mathrm{d}\mathbf{w}$ is the weiner noise. The evolution of the distribution of x_t is governed by the Kolmogorov Forward Equation (KFE) as,

$$\partial_t p\left(x_t\right) = -\partial_{x_t}\left[f\left(x_t\right) p\left(x_t\right)\right] + \frac{1}{2}\partial_{x_t}^2\left[g^2\left(x_t\right) p\left(x_t\right)\right] \tag{11}$$

Kolmogrov Forward/Backward Equation (KFE/KBE). Essentially KFE describes the evolution of a probability distribution $p(x_t)$ forward in time. The reverse-time SDE can be derived by solving the Kolmogorov Backward Equation (K.B.E) as derived in Anderson [1]. It can be defined for $t_1 \geq t_0$ as,

$$-\partial_t p\left(x_{t_1} \mid x_{t_0}\right) = f\left(x_{t_0}\right) \partial_{x_{t_0}} p\left(x_{t_1} \mid x_{t_0}\right) + \frac{1}{2} g^2\left(x_{t_0}\right) \partial_{x_{t_0}}^2 p\left(x_{t_1} \mid x_{t_0}\right) \tag{12}$$

where x_{t_0} and x_{t_1} are distributions at the respective time steps. Specifically, it models how the distribution dynamics at a later point t_1 in time changes as we change t_0 at an earlier time.

In our case, we consider the diffusion over structure $\mathbf{y}_s$ and properties $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$. The KFE of the system $\mathbf{y} = \{\mathbf{y}_s, \mathbf{y}_1, \ldots, \mathbf{y}_k\}$ is given by,

$$\partial_t p\left(\mathbf{y}_t\right) = -\partial_{\mathbf{y}_t}\left[f\left(\mathbf{y}_t\right) p\left(\mathbf{y}_t\right)\right] + \frac{1}{2}\partial_{\mathbf{y}_t}^2\left[g^2\left(\mathbf{y}_t\right) p\left(\mathbf{y}_t\right)\right] \tag{13}$$

Independence Factorization. We can factorize $p\left(\mathbf{y}_t\right)$ based on our assumption that the properties $\{\mathbf{y}_{1,t}, \ldots, \mathbf{y}_{k,t}\}$ are independent conditioned on the structure $\mathbf{y}_{s,t}$ as

$$\begin{aligned} p(\mathbf{y}_t) &= p(\mathbf{y}_{s,t}, \mathbf{y}_{1,t}, \ldots, \mathbf{y}_{k,t}) \\ &= p(\mathbf{y}_{s,t}) p(\mathbf{y}_{1,t}, \ldots, \mathbf{y}_{k,t} \mid \mathbf{y}_{s,t}) \\ &= p(\mathbf{y}_{s,t}) \prod_i^k p(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}) \end{aligned} \tag{14}$$

Leveraging this factorization, we can define a system of SDEs with KFEs for each variable, leading us to the SDE system defined in Eq. 1 and Eq. 2.

Reverse SDE: In the reverse case, we aim to denoise the full vector $\mathbf{y} = \{\mathbf{y}_s, \mathbf{y}_1, \ldots, \mathbf{y}_k\}$ where $\mathbf{y}_s$ denotes the diffusion over structure and $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ over the k properties via reverse SDE. Expressing in the form of Eq. 12, we note that for $t_1 \geq t_0$,

$$-\partial_t p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) = f\left(\mathbf{y}_{t_0}\right) \partial_{\mathbf{y}_{t_0}} p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) + \frac{1}{2} g^2\left(\mathbf{y}_{t_0}\right) \partial_{\mathbf{y}_{t_0}}^2 p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) \tag{15}$$

Anderson [1] defines a joint distribution over the time-ordered variables $\mathbf{y}_{t_1}$ and $\mathbf{y}_{t_0}$ to derive the reverse SDE. We utilize their analysis and define a joint distribution

$$\begin{aligned} p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right) &:= p\left(\mathbf{y}_{s,t_1}, \mathbf{y}_{1,t_1}, ..., \mathbf{y}_{k,t_1}, \mathbf{y}_{s,t_0}, \mathbf{y}_{1,t_0}, ..., \mathbf{y}_{k,t_0}\right) \\ &= p\left(\mathbf{y}_{s,t_1}, \mathbf{y}_{1,t_1}, ..., \mathbf{y}_{k,t_1} \mid \mathbf{y}_{s,t_0}, \mathbf{y}_{1,t_0}, ..., \mathbf{y}_{k,t_0}\right) p(\mathbf{y}_{s,t_0}, \mathbf{y}_{1,t_0}, \ldots, \mathbf{y}_{k,t_0}) \end{aligned} \tag{16}$$

We denote $p(\mathbf{y}_{s,t_0}, \mathbf{y}_{1,t_0}, \ldots, \mathbf{y}_{k,t_0})$ by $p(\mathbf{y}_{t_0})$, and note that it can be decomposed similarly as in Eq. 14. Taking the time derivative of Eq. 16, we get

$$\begin{aligned} -\partial_t p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right) = &-\partial_t p\left(\mathbf{y}_{s,t_1}, \mathbf{y}_{1,t_1}, ..., \mathbf{y}_{k,t_1} \mid \mathbf{y}_{s,t_0}, \mathbf{y}_{1,t_0}, ..., \mathbf{y}_{k,t_0}\right) p(\mathbf{y}_{t_0}) \\ &- \partial_t p(\mathbf{y}_{t_0}) p\left(\mathbf{y}_{s,t_1}, \mathbf{y}_{1,t_1}, ..., \mathbf{y}_{k,t_1} \mid \mathbf{y}_{s,t_0}, \mathbf{y}_{1,t_0}, ..., \mathbf{y}_{k,t_0}\right) \end{aligned} \tag{17}$$

Comparison with KFE/KBE. We observe that $\partial_t p\left(\mathbf{y}_{s,t_1}, \mathbf{y}_{1,t_1}, \ldots, \mathbf{y}_{k,t_1} \mid \mathbf{y}_{s,t_0}, \mathbf{y}_{1,t_0}, \ldots, \mathbf{y}_{k,t_0}\right)$ corresponds to the KBE in Eq. 15 and $\partial_t p(\mathbf{y}_{t_0})$ to the KFE in Eq. 13. Denoting

$\{\mathbf{y}_{s,t_1}, \mathbf{y}_{1,t_1}, \ldots, \mathbf{y}_{k t1}\}$ by $\mathbf{y}_{t_1}$, we immediately get

$$\begin{aligned}
&- \partial_t p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) p(\mathbf{y}_{t_0}) - \partial_t p(\mathbf{y}_{t_0}) p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) \\
&= \left(f\left(\mathbf{y}_{t_0}\right) \partial_{\mathbf{y}_{t_0}} p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) + \frac{1}{2} g^2\left(\mathbf{y}_{t_0}\right) \partial^2_{\mathbf{y}_{t_0}} p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right)\right) p(\mathbf{y}_{t_0}) \\
&+ p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) \left(\partial_{\mathbf{y}_{t_0}}\left[f\left(\mathbf{y}_{t_0}\right) p\left(\mathbf{y}_{t_0}\right)\right] - \frac{1}{2} \partial^2_{\mathbf{y}_{t_0}}\left[g^2\left(\mathbf{y}_{t_0}\right) p\left(\mathbf{y}_{t_0}\right)\right]\right)
\end{aligned} \tag{18}$$

The derivatives can be handled, by following standard differentiation rules as,

$$\begin{aligned}
\partial_{\mathbf{y}_{t_0}} p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) &= \partial_{\mathbf{y}_{t_0}}\left[\frac{p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right)}{p\left(\mathbf{y}_{t_0}\right)}\right] \\
&= \frac{\partial_{\mathbf{y}_{t_0}} p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right)}{p\left(\mathbf{y}_{t_0}\right)} - \frac{p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right) \partial_{\mathbf{y}_{t_0}} p\left(\mathbf{y}_{t_0}\right)}{p^2\left(\mathbf{y}_{t_0}\right)}
\end{aligned} \tag{19}$$

Evaluating the derivative of the products in the forward Kolmogorov equation and substituting the derivatives accordingly we obtain,

$$\begin{aligned}
-\partial_t p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right) = \partial_{\mathbf{y}_{t_0}}\left[f\left(\mathbf{y}_{t_0}\right) p\left(\mathbf{y}_{t_0}, \mathbf{y}_{t_1}\right)\right] &+ \frac{1}{2} g^2\left(\mathbf{y}_{t_0}\right) \partial^2_{\mathbf{y}_{t_0}} p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) p(\mathbf{y}_{t_0}) \\
&- \frac{1}{2} p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) \partial^2_{\mathbf{y}_{t_0}}\left[g^2\left(\mathbf{y}_{t_0}\right) p(\mathbf{y}_{t_0})\right]
\end{aligned} \tag{20}$$

Matching the terms of the second-order derivatives with the expansion of the derivative and doing some algebraic manipulations, we obtain

$$\begin{aligned}
-\partial_t p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right) = \partial_{\mathbf{y}_{t_0}}\left[f\left(\mathbf{y}_{t_0}\right) p\left(\mathbf{y}_{t_0}, \mathbf{y}_{t_1}\right)\right] &+ \frac{1}{2} \partial^2_{\mathbf{y}_{t_0}}\left[p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right) g^2\left(\mathbf{y}_{t_0}\right)\right] \\
&- \partial_{\mathbf{y}_{t_0}}\left[p\left(\mathbf{y}_{t_1} \mid \mathbf{y}_{t_0}\right) \partial_{\mathbf{y}_{t_0}}\left[g^2\left(\mathbf{y}_{t_0}\right) p\left(\mathbf{y}_{t_0}\right)\right]\right] ,
\end{aligned} \tag{21}$$

which can be written as

$$-\partial_t p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right) = -\partial_{\mathbf{y}_{t_0}}\left[p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right)\left(-f\left(\mathbf{y}_{t_0}\right) + \frac{1}{p\left(\mathbf{y}_{t_0}\right)} \partial_{\mathbf{y}_{t_0}}\left(g^2\left(\mathbf{y}_{t_0}\right) p\left(\mathbf{y}_{t_0}\right)\right)\right)\right] + \tag{22}$$

$$\frac{1}{2} \partial^2_{\mathbf{y}_{t_0}}\left[p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right) g^2\left(\mathbf{y}_{t_0}\right)\right] \tag{23}$$

Comparison with KFE. The above result is in the form of a Kolmogorov forward equation with the joint probability distribution $p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right)$. The time-ordering is $t_1 > t_0$ and the term $-\partial_t p\left(\mathbf{y}_{t_1}, \mathbf{y}_{t_0}\right)$ describes the change of probability distribution as we move backward in time. We can marginalize over t_1, using the Leibniz rule, to obtain

$$-\partial_t p\left(\mathbf{y}_{t_0}\right) = -\partial_{\mathbf{y}_{t_0}}\left[p\left(\mathbf{y}_{t_0}\right)\left(-f\left(\mathbf{y}_{t_0}\right) + \frac{1}{p\left(\mathbf{y}_{t_0}\right)} \partial_{\mathbf{y}_{t_0}}\left(g^2\left(\mathbf{y}_{t_0}\right) p\left(\mathbf{y}_{t_0}\right)\right)\right)\right] + \frac{1}{2} \partial^2_{\mathbf{y}_{t_0}}\left[p\left(\mathbf{y}_{t_0}\right) g^2\left(\mathbf{y}_{t_0}\right)\right] \tag{24}$$

This finally gives a stochastic differential equation analogous to the Fokker-Planck/forward Kolmogorov equation that can be solved backward in time:

$$d\mathbf{y}_{t_0} = \left(-f(\mathbf{y}_{t_0}, t) + \frac{1}{p\left(\mathbf{y}_{t_0}\right)} \partial_{\mathbf{y}_{t_0}}\left(g^2\left(\mathbf{y}_{t_0}\right) p\left(\mathbf{y}_{t_0}\right)\right)\right) \mathrm{d}t + g\left(\mathbf{y}_{t_0}\right) \mathrm{d}\mathbf{w} \tag{25}$$

We keep $g^2\left(\mathbf{y}_{t_0}\right)$ independent of $\mathbf{y}_{t_0}$. Applying the log-derivative trick, the SDE simplifies to

$$d\mathbf{y}_{t_0} = (f(\mathbf{y}_{t_0}, t) - g^2_{t_0} \nabla_{\mathbf{y}_{t_0}} \log p(\mathbf{y}_{t_0}))\mathrm{d}t + g_{t_0} \mathrm{d}\mathbf{w} \tag{26}$$

A.2 Conditional score factorization

We extend our method to incorporate an external context or conditional information for conditional generation, similar to classifier-based [8] and classifier-free [23] guidance. Following similar notation, the reverse SDE [67], given an external context $\mathbf{y}_C$ can be written as

$$\mathrm{d}\mathbf{y}_t = [\mathbf{f}(\mathbf{y}_t, t) - g_t^2 \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_t, \mathbf{y}_C)]\mathrm{d}t + g_t \mathrm{d}\bar{\mathbf{w}} \tag{27}$$

Here $\mathbf{y}_t = \{\mathbf{y}_{s,t}, \mathbf{y}_{1,t}, \ldots, \mathbf{y}_{k,t}\}$, and $\mathbf{y}_C = \{\mathbf{y}_c \mid c \in C\}$ is an external context or conditioning variable. This external context can be a scalar or vector describing a property value of the primary variable like QED or plogp in the case of molecules or image labels in the case of images. The $\nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_t, \mathbf{y}_C)$ term pertains to the score function which guides the process (see table 2 for comparison with both classifier-based and classifier-free guidance). Under our condition independence assumption, the score function factorizes as

$$p_t(\mathbf{y}_{s,t}, \mathbf{y}_{1,t}, \ldots, \mathbf{y}_{k,t}, \mathbf{y}_C) = \prod_i^k p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}, \mathbf{y}_c) p_t(\mathbf{y}_{s,t}, \mathbf{y}_C) \tag{28}$$

$$\begin{aligned} \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{s,t}, \mathbf{y}_{1,t}, \ldots, \mathbf{y}_{k,t}, \mathbf{y}_C) = \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{s,t}, \mathbf{y}_C) + \sum_{i \notin C}^{k} \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}) \\ + \sum_{c}^{C} \sum_{i}^{k} \delta_{i=c} \nabla_{\mathbf{y}_t} \log p_t(\mathbf{y}_{i,t} \mid \mathbf{y}_{s,t}, \mathbf{y}_c) \end{aligned} \tag{29}$$

B Parameterizations

Here we describe two instances of `Twigs` based on architecture choices: Attention networks, and graph convolution networks (GCNs). `Twigs` with attention is used in 4.1 and 4.2, while `Twigs` with GCNs is used in 4.3 and 4.4.

B.1 Twigs with graph attention

We denote the variable $\mathbf{y}_s$ as a 3D graph $\boldsymbol{G} = (\boldsymbol{A}, \boldsymbol{x}, \boldsymbol{h})$, with node coordinates $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N) \in \mathbb{R}^{N \times 3}$, node features $\boldsymbol{h} = (\boldsymbol{h}^1, \ldots, \boldsymbol{h}^N) \in \mathbb{R}^{N \times d1}$, and edge information $\boldsymbol{A} \in \mathbb{R}^{N \times N \times d2}$. The variable $\mathbf{C} \in \mathbb{R}$ denotes the conditional information, which is obtained by adding the noise level $\log(\alpha_t^2/\sigma_t^2)$, the perturbed property $\mathbf{y}_i \sim \mathcal{N}(0, I) \in \mathbb{R}$, and the fixed property $\mathbf{y}_C \in \mathbb{R}$. The context $\mathbf{C}$ is combined with $\mathbf{y}_s$ by multilayer perceptions (MLP), after projecting $(h, \boldsymbol{A}, x)$ respectively into $\mathbf{H}, \mathbf{E}, \mathbf{P}$:

$$\begin{aligned} &\mathrm{AdaLN} = (1 + \mathrm{MLP}(\mathbf{C})) \cdot \mathrm{LN}(\mathbf{H}) + \mathrm{MLP}(\mathbf{C}) \\ &\mathbf{M}^l = \mathrm{MHA}(\mathrm{AdaLN}(\mathbf{H}, \mathbf{C}), \mathrm{AdaLN}(\mathbf{E}, \mathbf{C}), \mathbf{P}) \end{aligned} \tag{30}$$

where MHA is the multi-head attention, and AdaLN is Adaptive LayerNorm (LN) function. Subsequently, we leverage the Scale function $\mathrm{Scale}(\mathrm{h}, \mathbf{C}) = \mathrm{MLP}(\mathbf{C}) \cdot \mathrm{h}$, and the Feed Forward Network (FFN) to obtain the Diffusion Graph Transformer (DGT) block, as defined in [28], which is described by Eq (31)(32). DGT first computes the intermediate representations for the l-th layer as:

$$\begin{aligned} \mathbf{M}^l &= \mathrm{MHA}(\mathrm{AdaLN}(\mathbf{H}^l, \mathbf{C}), \mathrm{AdaLN}(\mathbf{E}^l, \mathbf{C}), \mathbf{P}^l) \\ \hat{\mathbf{H}} &= \mathrm{Scale}(\mathbf{M}^l, \mathbf{C}) + \mathbf{H}^l \\ \hat{\mathbf{E}} &= \mathrm{Scale}(\mathbf{M}_i^l + \mathbf{M}_j^l, \mathbf{C}) + \mathbf{E}^l \end{aligned} \tag{31}$$

then computes the $l + 1$ layer as:

$$\begin{aligned} &\mathbf{E}^{l+1} = \mathrm{Scale}(\mathrm{FFN}(\mathrm{AdaLN}(\mathrm{Scale}(\hat{\mathbf{E}}, C)), C) + \hat{\mathbf{E}} \\ &\mathbf{H}^{l+1} = \mathrm{Scale}(\mathrm{FFN}(\mathrm{AdaLN}(\hat{\mathbf{H}}, \mathbf{C})), \mathbf{C}) + \hat{\mathbf{H}} \\ &\mathbf{P}_i^{l+1} = \textstyle\sum_{i \neq j} \frac{\mathbf{P}_i^l - \mathbf{P}_j^l}{||\mathbf{P}_i^l - \mathbf{P}_j^l||^2} \tanh(\mathrm{MLP}(\mathbf{E}_{i,j}^{l+1})) \end{aligned} \tag{32}$$

The `Twigs` trunk process s_θ is parameterized as:

$$s_\theta = \mathrm{DGT}(\mathbf{y}_s, \mathbf{y}_i, \mathbf{y}_C) + \textstyle\sum_i \mathrm{PDGT}_i(\mathbf{y}_s, \mathbf{y}_i, \mathbf{y}_C) \tag{33}$$

where PDGT_i resembles the stem process networks s_{ϕ_i}, which is obtained by pooling to a one-dimensional variable by an MLP operation, over the output of the DGT block. To optimize Eq (9), DGT minimizes the denoising score matching objective from [28] for node, edge and position information $(h, \boldsymbol{A}, x)$, while PDGT_i for the perturbed property $\mathbf{y}_i$.

B.2 Twigs with graph convolutions

In the case of 2D graphs with N nodes we consider the variable $\mathbf{y}_s = (\boldsymbol{X}, \boldsymbol{A}) \in \mathbb{R}^{N\times F} \times \mathbb{R}^{N\times N}$, where F is the dimension of the node features, $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ are node features, $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is weighted adjacency matrix. We define the perturbed property $\mathbf{y}_i \in \mathbb{R}$ and the (fixed) property $\mathbf{y}_C \in \mathbb{R}$. The stem process network s_{ϕ_i} is given as:

$$s_{\phi_i} = \mathrm{MLP}_i(\mathrm{GNN}(\mathrm{P}_i, \boldsymbol{A})); \quad \mathrm{P}_i = (\boldsymbol{X} \,\|\, \mathbf{v}_i, \|\mathbf{v}_C) \tag{34}$$

where $\mathbf{v}_i$ and $\mathbf{v}_C$ are vectors obtained by repeating N times the perturbed property $\mathbf{y}_i$ and the fixed property $\mathbf{y}_C$ respectively, and concatenating them into the node features matrix $\boldsymbol{X}$. The `Twigs` trunk process s_θ is obtained by combining the contributions from the properties $\mathbf{y}_i$ derived by the stem processes s_{ϕ_i} and the structure $\mathbf{y}_s$, as follows:

$$s_\theta = s_{\theta^X}(\boldsymbol{X}, \boldsymbol{A}, \mathbf{y}_C) + \textstyle\sum_i s_{\phi_i}(\boldsymbol{X}, \mathbf{y}_i, \mathbf{y}_C) \tag{35}$$

where s_{θ^X} is a conditional node feature score network: $s_{\theta^X} = \mathrm{MLP}(\mathrm{GNN}(\boldsymbol{X} \,\|\, \mathbf{y}_C, \boldsymbol{A}))$. Finally, following [37], $\boldsymbol{A}$ is co-evolved together with the node features, by the adjacency score model $\boldsymbol{s}_\theta^A$

$$\boldsymbol{s}_{\theta^A} = \mathrm{MLP}\left(\left[\{\mathrm{GMH}\,(\boldsymbol{H}_i, \boldsymbol{A}_t^p)\}_{i=0,p=1}^{K,P}\right]\right) \tag{36}$$

where GMH is graph multi-head attention [2], which employs higher-order adjacency matrices $\boldsymbol{A}_t^p$, and K denotes the number of GMH layers. The optimization for the `Twigs` objective function (9), is obtained by minimizing the denoising score matching for $\boldsymbol{A}, \boldsymbol{X}, \mathrm{P}_i$.

The GMH block employs higher-order adjacency matrices $\boldsymbol{A}_t^p$ to represent the long-range dependencies and is provided as: $\boldsymbol{s}_{\theta^A}(\boldsymbol{G}_t) = \mathrm{MLP}\left(\left[\{\mathrm{GMH}\,(\boldsymbol{H}_i, \boldsymbol{A}_t^p)\}_{i=0,p=1}^{K,P}\right]\right)$.

C Additional experimental results

C.1 QM9 dataset

Further details for generation conditioned on quantum properties from Section 4.1.

Molecular quality. Additional results for molecular stability in 2D and Fréchet ChemNet Distance (FCD) for 2D and 3D are given in Table 12.

Table 12: Molecule quality results.

Property	Mol-S-2D ↑	FCD-2D ↓	FCD-3D ↓
C_v	98.88	0.107	0.871
μ	98.93	0.125	0.842
α	98.71	0.106	0.867
$\Delta\epsilon$	98.82	0.105	0.787
ϵ_{HOMO}	98.95	0.111	0.827
ϵ_{LUMO}	98.52	0.117	0.846

C.2 ZINC250K dataset

Conditional generation. The evaluation is performed by measuring the MAE of the pre-trained predictors released from [45], which given a molecule G_t are trained to predict

$$\mathrm{Obj} = \widehat{\mathrm{DS}}(\boldsymbol{G}_t) \times \mathrm{QED}(\boldsymbol{G}_t) \times \widehat{\mathrm{SA}}(\boldsymbol{G}_t) \tag{37}$$

where $\widehat{\mathrm{DS}}$ is the normalized docking score (DS) of the considered target protein, QED is the drug-likeness, and $\widehat{\mathrm{SA}}$ is the normalized synthetic accessibility (SA).

In terms of baselines, we consider the MOOD model [45], which leverages a classifier-based guidance scheme, and we also implement a diffusion guidance version of GDSS [37] based on the classifier-free scheme. Our `Twigs` method is parameterized by the architecture described in B.2, with a single stem process. The models are conditioned on the function in Equation (37).

Table 13: MAE for ZINC250K conditioned on single properties.

	parp1	fa7	5ht1b	braf	jak2
GDSS	5.56	4.76	5.78	5.73	5.98
MOOD ood=0.04	5.42	4.33	5.52	5.37	5.10
MOOD ood=0.01	5.41	4.33	5.52	5.36	5.09
`Twigs`	**5.38**	**4.30**	**5.43**	**5.28**	**5.01**

Results. In Table 13, we report the mean MAE values over multiple runs computed from the generated molecules using the pre-trained classifiers from [45]. We can observe that the `Twigs` consistently achieves a lower error, demonstrating an improved control over generating molecules with the desired target proteins.

Runtime. We have incorporated the runtime for molecule generation at inference time for a large-scale dataset (ZINC250K) as for Section 4.3, in Table 14. A comparison with MOOD [45] indicates that our model incurs a certain overhead, as anticipated. However, it demonstrates improved alignment when generating conditional molecules.

Table 14: Runtime for inference on molecule generation.

model	Seconds per molecule
Twigs	0.378
MOOD	0.267

D Experimental details

D.1 Computational resources

All experiments are performed with GPUs, Nvidia A100 or v100.

D.2 Models details

We follow the data splits from Huang et al. [28] for 4.1, 4.2, the ones from Lee et al. [45] for 4.3, and the data splits from Jo et al. [37] for 4.4. We use Adam optimizers on all experiments.

For Sections 4.1 and 4.2 we follow the same hyperparameters from Huang et al. [28]. For Section 4.3 we follow the hyperparameters from Lee et al. [45], for the MOOD baseline, we explore OOD coefficients between 0.01 and 0.09. For Section 4.4 we follow the hyperparameters from Jo et al. [37].

E Additional Related Works

This section extends the discussion presented in Section 2 by exploring additional related works in the field. In Table 15 we summarise related methods including score-sdes, hierarchical models (not necessarily conditional), and hierarchical conditional models.

Conditional molecular diffusion. Guidance techniques have also been adopted in conditional molecule generation settings: in the context of classifier-free approaches, Hoogeboom et al. [26] proposes an equivariant approach based on DDPM for 3D molecules; Huang et al. [28] explores attention mechanisms within SGM models; and Xu et al. [82] investigates DDPMs in latent space settings.

In terms of classifier-based guidance, Bao et al. [3] incorporate energy guidance into a diffusion model by leveraging a stochastic differential equation; Vignac et al. [75] provide a DDPM coupled with a classifier over quantum molecular properties; and Lee et al. [45] operate over a pre-trained SGM and train an additional predictor for fine-tuning the desired protein target properties.

Table 15: Comparison with related works.

Method	Score-based SDE	Hierarchical modeling	Hierarchical conditional diffusion
EDM [26]	✗	✗	✗
EEGSDE [3]	✓	✗	✗
Digress [75]	✗	✗	✗
HierVAE [34]	✗	✓	✗
GraphGuide [72]	✗	✓	✗
GeoLDM [82]	✗	✗	✗
HierGraph [57]	✗	✓	✗
JODO [28]	✓	✗	✗
`Twigs` (this work)	✓	✓	✓

Guidance methods. Recent works utilize multiple diffusion processes: cascaded diffusion [25], provides a flow for each resolution, and GDSS [37] has a joint system of diffusion processes one for nodes and the other for edge features, but it does not cover mechanisms for conditional generation. Tseng et al. [71] define a hierarchy of branching points within a single diffusion flow.

Other Diffusion methods for Graphs. Other works related to ours focus on hierarchical diffusion processes [7], diffusion applied to protein backbones [69], geometry-based models [59, 82], and autoregressive models [41]. In the realm of stochastic differential equation (SDE)-based approaches, the literature includes bridge methods [38], permutation invariance [27], torsional modeling [36], and docking [6]. Additionally, [63] introduces the ConfGF approach, estimating gradient fields of atomic coordinates, while [80] proposes a method steering the training of diffusion-based generative models using physical and statistical prior information.

Autoencoder-Based graph models. This category includes works employing autoencoders, such as retrieval-based models [77, 12], scaffold modeling [50], link design [29], and coarse-grain modeling [76]. Notably, [54] proposes a reaction-embedded and structure-conditioned variational autoencoder, while [42] defines the concept of principal subgraphs, relevant to informative patterns within molecules.

Conditional Diffusion. In the realm of diffusion generative models, several noteworthy approaches have been developed to enhance their performance and versatility. Du et al. [10] introduce an energy-based parameterization of diffusion models, allowing the integration of novel compositional operators and Metropolis-corrected samplers. Building on this, He et al. [22] contribute a training-free conditional generation framework, leveraging pretrained diffusion models focusing on the manifold hypothesis to refine guided diffusion steps and introduce a shortcut algorithm. Meanwhile, Meng et al. [51] employ a stochastic differential equation (SDE) in synthesizing realistic images, iterating through denoising steps guided by a pretrained diffusion model.

In a different vein, Song et al. [66] propose guiding denoising diffusion models with general differentiable loss functions in a plug-and-play manner, facilitating controllable generation without additional training. Addressing the challenge of inferring high-dimensional data within the context of diffusion models, Graikos et al. [17] present a model consisting of a prior and an auxiliary differentiable constraint. Dinh et al. [9] tackle diversity and adversarial effects in classifier guidance for diffusion generative models by allowing relevant classes' gradients to contribute to shared information construction during noisy early sampling steps. Furthermore, Song et al. [65] put forth a method for estimating conditional scores without additional training. Lastly, Ouyang et al. [56] propose the Contrastive-Guided Diffusion Process (Contrastive-DP), integrating contrastive loss to guide the diffusion model in data generation. These diverse contributions collectively advance the field by addressing various challenges and expanding the capabilities of diffusion generative models.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We demonstrate with theoretical results and a comprehensive set of experiments.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations provided in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Proofs provided in Section A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Section 4 we provide details to reproduce the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

The implementation details are in appendix to run the experiments. The used datasets are public and can be accessed with the reference paper.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Described in D.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report mean and standard deviation in our experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Described in D.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Provided in Section 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Does not apply for our paper.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The resources that we used are cited, the source code we used is released on open licenses.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not introduce new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve Crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve IRB.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.