Diffuse, Sample, Project: Plug-And-Play Controllable Graph Generation

Kartik Sharma [1] Srijan Kumar [1] Rakshit S Trivedi [2]

Abstract

Diffusion models lend transformative capabilities to the graph generation task, yet controlling the properties of the generated graphs remains challenging. Recent approaches augment support for controlling soft, differentiable properties but they fail to handle user-specified hard constraints that are non-differentiable. This often results in vague control, unsuitable for applications like drug discovery that demand satisfaction of precise constraints, *e.g.*, the maximum number of bonds. To address this, we formalize the problem of controlled graph generation and introduce PRODIGY (PROjected DIffusion for controlled Graph Generation), an innovative plug-and-play approach enabling the generation of graphs with precise control, from any pre-trained diffusion model. PRODIGY employs a novel operator to project the samples at each diffusion step onto the specified constrained space. For a large class of practical constraints and a variety of graphs, our extensive experiments demonstrate that PRODIGY empowers state-of-the-art continuous and discrete diffusion models to produce graphs meeting specific, hard constraints. Our approach achieves up to 100% constraint satisfaction for non-attributed and molecular graphs, under a variety of constraints, marking a significant step forward in precise, interpretable graph generation. Code is provided on the project webpage: https://prodigy-diffusion.github.io.

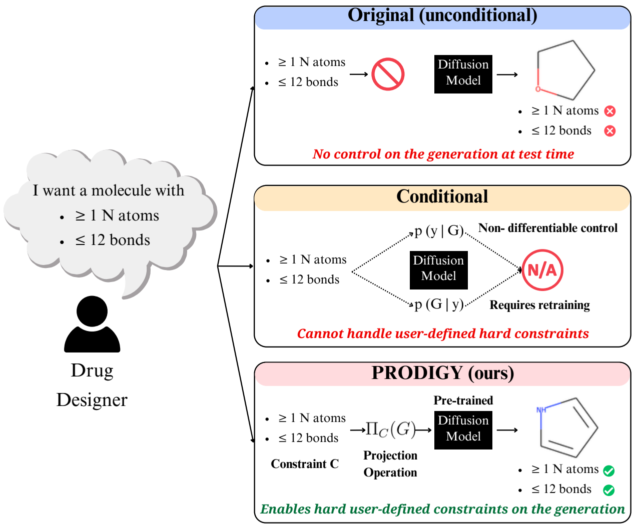


Figure 1. Comparison of the existing methods (unconditional and conditional) with PRODIGY for the problem of plug-and-play controllable graph generation. While unconditional models (Jo et al., 2022; Vignac et al., 2022) do not allow test-time control on generation, conditional models cannot handle hard constraints as they either require retraining (Xu et al., 2023) or assume differentiability (Vignac et al., 2022). PRODIGY enables controllable graph generation from any pre-trained diffusion model.

1. Introduction

Deep generative models serve as an effective approach to learn the underlying distribution of graph-structured data (You et al., 2018; Jo et al., 2022; Martinkus et al., 2022; Liu et al., 2019). Recently, diffusion-based models (Niu et al., 2020; Vignac et al., 2022; Jo et al., 2022; 2023) have shown impressive performance in generating graphs in an efficient manner and achieving distributional realism that outperforms most of its contemporary autoregressive and adversarial learning frameworks. The ultimate objective of the graph generation research field is to enable the simulation of large-scale networks that can help make tangible progress in domains such as network optimization (Xie et al., 2019), social network analysis (Grover et al., 2019), and drug design (Yang et al., 2022).

While demonstrating excellent performance in terms of matching the data generating distribution on benchmark graphs datasets, current diffusion-based approaches suffer from a key limitation that keeps them away from use in practice: inability to support **meaningful, precise control** of properties of the generated graphs. This limits their application in domains such as drug discovery and mate-

[1]Georgia Institute of Technology, Atlanta, GA, USA [2]Massachusetts Institute of Technology, Cambridge, MA, USA. Correspondence to: Kartik Sharma <ksartik@gatech.edu>, Rakshit Trivedi <rstrivedi@csail.mit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

rial science. To achieve this ability, the generative models should be designed such that they can generate graphs under user-specified, hard constraints. The key challenge with this approach is that such constraints are **non-differentiable**. A naive solution to support such constraints would rely on **curating additional labeled datasets** that provide constrained ground-truth graphs for each possible constraint and/or **retraining** the entire generative model.

Existing methods (Hoogeboom et al., 2022; Xu et al., 2023; Vignac et al., 2022) partly adopt this solution strategy in the form of conditional generation of graphs by approximating the conditional probability distribution with the property. However, these approaches require the property (or its approximation) to be differentiable and it influences the sampling process in an obscure and uninterpretable manner. This limitation severely restricts the applicability of these methods in graph-related applications where there are several structural properties that need to be precisely controlled when generating from the pre-trained models.

In this work, we fill this gap by investigating the problem of controllable graph generation to generate graphs that satisfy certain user-defined hard constraints on their structure and derived properties. This paper introduces PRODIGY (PROjected DIffusion for generating constrained Graphs), a **plug-and-play** controllable graph generation method inspired by theoretical works on Projected Langevin sampling (Bubeck et al., 2018). Specifically, we propose a novel sampling process that, given any pre-trained diffusion model, augments the denoising step with a (weighted) projection step onto the given constrained space. To exemplify the recipe of using PRODIGY for generating desired graphs, we present a novel framework to devise various graph constraints and find efficient projection operators for them.

Figure 1 summarizes the motivation of our approach. Through extensive experiments on 5 non-attributed and 2 molecular graph datasets, we showcase the superior performance of PRODIGY in controlling the properties of generated graphs, while preserving the distribution learned by the underlying pre-trained diffusion model. PRODIGY achieves up to 100% constraint satisfaction for constraints on a variety of properties such as edge count, atom counts, etc. on these datasets across 4 different diffusion models. This performance is also observed in 3D molecule generation, thereby demonstrating the ability of our approach to effectively handle complex structures. We further qualitatively demonstrate the versatile applicability of PRODIGY through efficiency, sensitivity, and visual analysis.

2. Background & Related Work

Suppose $\mathbf{G} = (\mathbf{X}, \mathbf{A})$ denotes an undirected graph with attribute matrix $\mathbf{X} \in \mathbb{R}^{n\times F}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, where $n = |\mathbf{V}|$ is the number of nodes. A 3D structure, on the other hand, can be defined as a point cloud $\mathbf{G} = (\mathbf{X}, \mathbf{S})$ with $\mathbf{S} \in \mathbb{R}^{n\times 3}$ denoting the positions of each node in the 3-dimensional space. Let $\mathcal{G}$ denote the set of all possible graphs for the structural cases and the set of all point clouds for the 3D case. All vectors and matrices are represented using bold lowercase and uppercase characters. We also use $\mathbf{1}$ and $\mathbf{0}$ to denote an all-ones and an all-zeros vector with the appropriate size for the usage, e.g., in $\mathbf{A1}$, $\mathbf{1}$ denotes a n-dimensional vector.

Diffusion Models For Graphs. Denoising diffusion models have demonstrated significant success in generating graphs for various purposes (Niu et al., 2020; Jo et al., 2022; Yan et al., 2023; Jo et al., 2023; Vignac et al., 2022; Chen et al., 2023). They are based on the idea of learning to denoise multiple noise-diffused versions of a given data to better approximate an unknown target distribution. This allows us to effectively generate novel graphs from the underlying distribution by simply removing the noise from a fixed prior distribution p_T. In particular, given a graph $\mathbf{G}_0 \sim p_0$ (p_0 is not known), they transform it to a completely random graph $\mathbf{G}_T \sim p_T = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ or $U[a, b]$ by following the "forward" step for T times as

$$\mathbf{G}_{t+1} \leftarrow \text{Forward}(\mathbf{G}_t, t, \boldsymbol{\varepsilon}(t)), \tag{1}$$

where $\varepsilon(t)$ denotes a pre-defined noise at time t, *e.g.*, $\varepsilon(t) = \mathbf{w}(t) \in \mathcal{G}$ would be a standard Wiener process on the graphs. The noisy graphs $\{\mathbf{G}_t\}$ are maintained to be discrete at all time steps in the discrete diffusion models (Vignac et al., 2022), while they are relaxed to the continuous space in the continuous models (Niu et al., 2020; Jo et al., 2022).

To generate samples from the unknown data distribution p_0, the forward process is reversed so that samples from the prior distribution can be converted to the target distribution. This is done by learning a parameterized neural network $\mathbf{s}_\theta(\mathbf{G}, t) \approx \nabla_{\mathbf{G}} \log p_t(\mathbf{G})$ in the continuous models and $\mathbf{s}_\theta(\mathbf{G}, t) \approx p_t(\mathbf{G})$ in the discrete models. While a score-matching objective (Song et al., 2020) is used to train the former, one can simply use a cross-entropy loss to train the $\mathbf{s}_\theta$ for the discrete case. Then, they follow the "reverse" step for T time steps to generate a graph sample that likely belongs to p_0, starting from a graph $\mathbf{G}_T \sim p_T$.

$$\mathbf{G}_{t-1} \leftarrow \text{Reverse}(\mathbf{G}_t, \mathbf{s}_\theta(\mathbf{G}_t, t), \bar{\varepsilon}(t), t), \tag{2}$$

where $\bar{\varepsilon}(t)$ denotes a pre-defined reverse-time noise at time t, *e.g.*, $\varepsilon(t) = \bar{\mathbf{w}}(t) \in \mathcal{G}$ would be a standard reverse-time Wiener process on the graphs.

Appendix A provide a more detailed discussion on both continuous and discrete diffusion models.

Conditional Generation of Graphs. Recent advancements in diffusion models for graphs augment support for conditioning the output of the generated graphs. They operate by approximating a conditional probability distribution $p(\mathbf{G}|c) := p(\mathbf{G}|c(\mathbf{G}, y))$. Typically, the condition $c(\mathbf{G}, y) = \mathbb{1}\{y_c(\mathbf{G}) = y\}$, i.e., $c(\mathbf{G}, y) = 1$ if $y_c(\mathbf{G}) = y$ and 0 otherwise. Note that this does not guarantee that $c(\mathbf{G}, y)$ will hold for the sampled output $\mathbf{G}$ since it forms a cyclical Markov chain. To the best of our knowledge, there have been sporadic attempts to support control with diffusion models of graph generation, and all existing works in this space fall under this category of soft control. Conditional denoising models (Hoogeboom et al., 2022; Xu et al., 2023) learn a conditional score function $s_\theta(\mathbf{G}, c) \approx \nabla \log p(\mathbf{G}|c)$ and classifier-free guidance (Ho & Salimans, 2022; Ninniri et al., 2023) learns a joint model for both conditional and unconditional sampling for a given control condition. Thus, each condition type demands a unique model and cannot be used in a plug-and-play manner for an unconditional model as it requires retraining the model for a new control. On the other hand, guidance-based diffusion methods (Vignac et al., 2022; Graikos et al., 2022; Lee et al., 2023; Li et al., 2022; Song et al., 2023) infer $p(\mathbf{G}|c)$ from $p(c|\mathbf{G})$ by using the fact that $\nabla \log p(\mathbf{G}|c) \approx s_\theta(\mathbf{G}) + \nabla \log p(c|\mathbf{G})$. This allows for plug-and-play conditional control on pre-trained diffusion models $s_\theta(\mathbf{G})$ as long as we can approximate $\nabla \log p(c|\mathbf{G})$. When $c(\cdot)$ is not known, it is approximated by a classifier $\hat{y}_c$ while when it is known, the property c is assumed to be a differentiable function of $\mathbf{G}$ and y. Classifier-based guidance requires labeled data along with capabilities to train a classifier for every new control.

One can view the above approaches as providing **soft** control over the generated outputs. In real-world settings, it is often crucial for practitioners to have *precise* control over the properties of the generated graph, *e.g.*, constraining the number of Nitrogen atoms. This is an open and challenging problem where it is impossible to directly apply the above-discussed approaches due to their requirements of **retraining**, **condition labels**, and **differentiability**.

Controllable Generation of Graphs. Precise control on the generated output can be formulated in terms of specific well-defined constraints. A user-defined constraint function $c(\cdot)$ is characterized by its feasible set $\mathcal{C}$ such that $c(\mathbf{G}) = 1$ if $\mathbf{G} \in \mathcal{C}$ and 0 otherwise. Thus, $c(\mathbf{G})$ is non-differentiable. One can view this as requiring a **hard** control over the generated outputs. To the best of our knowledge, no prior work exists that can support graph generation under arbitrary hard constraints from pre-trained diffusion models. Recently, controllable generation of images have garnered attention but the approaches there are not directly applicable to domain of graphs. Mirror diffusion models learn specialized diffusion models to generate images under certain convex constraint sets, particularly for watermark generation (Liu et al., 2023). However, this does not provide plug-and-play control for these constraints at test time. On the other hand, Bar-Tal et al. (2023) enabled constraint-based control in image diffusion models, but limit their focus to specific image constraints such as panorama, aspect ratio, and spatial guiding, by solving a specific optimization problem to match the pre-trained sampling process in the constrained space.

Projected Sampling. In the literature, the theoretical ability of projected/mirrored Langevin sampling to enable constrained sampling from underlying distribution has been explored (Bubeck et al., 2018; Hsieh et al., 2018). However, its effectiveness for deep learning-based diffusion models is still unknown as the underlying distribution is approximated from the training data, which would render sampling infeasible in uncertain domains. Furthermore, diffusion models employ additional reverse-SDE dynamics on top of the simple Langevin sampling. Finally, efficient projections for many graph-level constraints need to be derived for application in this framework. In this work, we address all these challenges by studying newly proposed variants of projected sampling in the realm of modern diffusion models under a set of practically-motivated graph constraints.

3. Problem Setup: Plug-and-Play Control

Given a set of training graphs $\mathcal{G}_{tr} \subset \mathcal{G}$ ($\mathcal{G}$ is the set of all possible graphs), the problem of graph generation involves learning the underlying distribution p_0 of $\mathcal{G}_{tr}$ and sampling from the learned distribution $\hat{p}_0$ to generate new graphs $\{\mathbf{G}\}$ such that they mimic the training distribution p_0. In this work, we consider the problem of *controllable graph generation*, where the objective is to control the generative process within a **specified constraint's feasible set**, while preserving the distribution $\hat{p}_0$ learned by the underlying diffusion model. Concretely, we solve the following problem:

Problem 1. ***(Plug-and-Play Controllable Graph Generation)*** *Given a constraint feasible set $\mathcal{C} \subset \mathcal{G}$ and a pre-trained unconditional graph generation model $\mathcal{M}$ trained on some training set $\mathcal{G}_{tr}$, generate new graphs $\{\mathbf{G}\} \sim \hat{p}_0^{\mathcal{C}}$ such that $\hat{p}_0^{\mathcal{C}} \approx \hat{p}_0$ and $\hat{p}_0^{\mathcal{C}}$ has support $\mathcal{C}$.*

Key Assumption: Model $\mathcal{M}$ may not be available for further training nor do we assume access to training set $\mathcal{G}_{tr}$ or the model parameters $\mathbf{\Theta}(\mathcal{M})$ as these are often not released due to proprietary reasons (Ramesh et al., 2021; OpenAI, 2023). Thus, the proposed method is required to be flexible to the choice of the the constraints, training graphs and the underlying model (**plug-and-play approach**). Next, we discuss general class of constraints supported by our approach and outline the particular instances studied in this work.

3.1. Constraints

We consider a wide range of arbitrary constraints with a focus on interpretability and minimal restrictions. Concretely, our approach is able to handle any constraint of the form $\mathcal{C} = \{\mathbf{G} : h_1(\mathbf{G}) \le 0, h_2(\mathbf{G}) \le 0, \cdots, h_k(\mathbf{G}) \le 0\}$, with efficient solutions of simultaneous equality.

As such, a practitioner may be interested in controlling the generation with a variety of constraints on the structure and the derived properties of the graph, depending on the downstream applications. To this end, we motivate our approach by instantiating a set of constraints based on well-studied graph properties in both non-attributed graphs (with applications to network design and efficiency) and molecules (with applications to drug design). Below we discuss the set of constraints[1] that we instantiate to provide the recipe of our approach and further discuss extensions towards more complex properties in Appendix C and E.2.

Non-attributed Graphs. A user may want to control the number of different substructures in the graph (Tabourier et al., 2011; Ying & Wu, 2009) since these represent different aspects of real-world network design (Farahani et al., 2013). We consider three such constraints outlined in Table 1. Here, $\mathbf{E}, \mathbf{A}$ denotes the edge set and the adjacency matrix and $\text{tr}(\cdot)$ finds the trace of the given matrix. The constants $\mathcal{B}, T, \delta_d$ form the constraint parameters that can be varied. In network design problems, Edge Count reflects a budget on the total (unweighted) cost of roads, degree measures network efficiency for the load on a node, and triangle count measures the local clustering.

Table 1. Constraints studied for non-attributed graphs.

Edge Count	Number of edges $\lvert\mathbf{E}\rvert = \frac{1}{2}\mathbf{1}^T\mathbf{A}\mathbf{1}$ $\le \mathcal{B}$ for a given constant $\mathcal{B} \ge 0$
Triangle Count	Number of triangles $\frac{1}{6}\text{tr}(\mathbf{A}^3)$ $\le T$ for a given constant $T \ge 0$
Degree	Degree of each node is bounded by a constant δ_d, *i.e.*, $\mathbf{A}\mathbf{1} \le \delta_d\mathbf{1}$

Molecular graphs. Assume $\mathbf{X}$ denotes the one-hot encoding of each node being a certain atom $\in \{1, 2, \cdots, F\}$. It is often desired that the generated molecule is valid (Vignac et al., 2022; Jo et al., 2022) and has some desired properties. Chemical descriptors (Todeschini & Consonni, 2008) link molecular structure to its properties. At a base level, a molecular structure is comprised of atoms $\mathbf{X}$, their connections $\mathbf{A}$ and 3D positions $\mathbf{S} \in \mathbb{R}^{n\times 3}$. Assuming hidden Hydrogen atoms (Jo et al., 2022), Table 2 outlines three such constraints of interest based on molecular structures. Variables $\mathbf{v}, \mathbf{c}, W, \xi_0, \xi_1$ form the constraint parameters for these and can be varied accordingly.

[1]We note that this list comprises a representative and practical set of constraints but not an exhaustive one – PRODIGY can handle arbitrary constraints and we consider the current instantiations to provide an extensive proof of concept of our approach.

Table 2. Constraints studied for molecular graphs.

Valency	Given valencies $\mathbf{v}$, degree is at most valency, *i.e.*, $\mathbf{A}\mathbf{1} \le \mathbf{X}\mathbf{v}$
Atom Count	Number of atoms of each type is bounded, *i.e.*, $\mathbf{X}^T\mathbf{1} \le \mathbf{c}$, for counts $\mathbf{c}$
Molecular Weight	Total weight is bounded by W, *i.e.*, $\mathbf{1}^T\mathbf{X}\mathbf{m} \le W$, for atomic weights $\mathbf{m}$
Dipole Moment	Norm of the vector sum of the atomic charges $\mathbf{Q}$ is bounded, *i.e.*, $\lVert\mathbf{S}^T\mathbf{X}\mathbf{Q}\rVert_2 \in [\xi_0, \xi_1]$

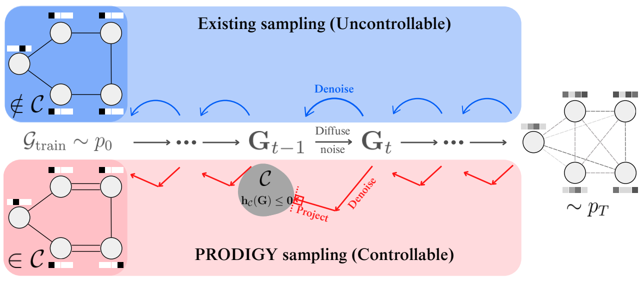

Existing sampling (Uncontrollable)
Denoise
$\notin \mathcal{C}$
$\mathcal{G}_{\text{train}} \sim p_0$
$\mathbf{G}_{t-1}$
Diffuse
noise
$\mathbf{G}_t$
$\sim p_T$
$\mathcal{C}$
$h_{\mathcal{C}}(\mathbf{G}) \le 0$
Project
Denoise
$\in \mathcal{C}$
PRODIGY sampling (Controllable)

Figure 2. Comparison of existing and proposed projected diffusion sampling methods for generating graphs under the given constrained set $\mathcal{C}$ (*e.g.*, number of edges is at most 12).

4. Proposed Method: PRODIGY

We propose PROjected DIffusion for controllable Graph generation (PRODIGY), a plug-and-play sampling approach that enables **any (discrete or continuous)**, **pre-trained** diffusion models for graph generation to generate graphs that satisfy **hard, interpretable** constraints specified by the user. Figure 2 provides an illustrative view of how PRODIGY operates. One simple instantiation of our method can be made by extending theoretical works in Mirrored Langevin Dynamics (Bubeck et al., 2018; Hsieh et al., 2018) to modern diffusion models by doing alternate denoising and projection, *i.e.*,

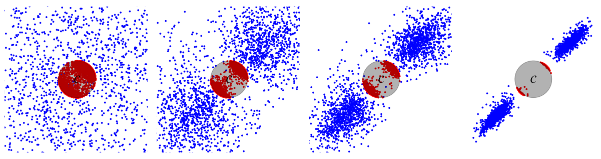
$$\begin{cases} \widetilde{\mathbf{G}}_{t-1} \leftarrow \text{Reverse}(\mathbf{G}_t, \mathbf{s}_\theta(\mathbf{G}_t, t), \bar{\varepsilon}_t, t) \\ \mathbf{G}_{t-1} \leftarrow \Pi_{\mathcal{C}}(\widetilde{\mathbf{G}}_{t-1}), \end{cases} \tag{3}$$

where Reverse the reverse process defined in Equation 2 with a pre-trained score function $\mathbf{s}_\theta$ and the projection operator $\Pi_{\mathcal{C}}$, $\Pi_{\mathcal{C}}(\mathbf{G}) = \arg\min_{\mathbf{z}\in\mathcal{C}} \lVert\mathbf{z} - \mathbf{G}\rVert_2^2$. Figure 3 visualizes the sampling process of PRODIGY (red) on GDSS as compared to the original model (blue). Our approach enables the baseline model to sample within the constrained ℓ_2 ball while preserving the distribution learned by that model.

However, this simple solution is not optimal since a complete projection to an arbitrary constraint feasible set during sampling can destroy the smoothness of the learned reverse process. This is because the denoised sample can be far from the feasible set and the projection step can lead the dynamics into a region of high uncertainty. To account for

Table 3. Projection Operators for different constraints, given as $\Pi_{\mathcal{C}}(\mathbf{G}) = \varphi_{\mathbf{0}}(\mathbf{G})$ if $\mathbf{h}_{\mathcal{C}}(\varphi_{\mathbf{0}}(\mathbf{G})) \leq \mathbf{0}$ otherwise $\varphi_{\boldsymbol{\mu}}(\mathbf{G})$ such that $\mathbf{h}_{\mathcal{C}}(\varphi_{\boldsymbol{\mu}}(\mathbf{G})) = \mathbf{0}$. See Appendix B for proofs and extensions.

2D structure $\mathbf{G} = (\mathbf{X}, \mathbf{A})$. $\mathbf{X} \in [\mathbf{0}, \mathbf{1}]$, $\mathbf{A} \in [\mathbf{0}, \mathbf{1}]$ or $\in [\mathbf{0}, \mathbf{3}]$, $\mathbf{A}^T = \mathbf{A}, \text{Diag}(\mathbf{A}) = \mathbf{0}$			
Constraint ($\mathcal{C}$)	Function ($\mathbf{h}_{\mathcal{C}}$)	$\varphi_{\boldsymbol{\mu}}^X$	$\varphi_{\boldsymbol{\mu}}^A$
Edge Count	$\frac{1}{2}\mathbf{1}^T\mathbf{A}\mathbf{1} - \mathcal{B}$	$\mathbf{X}$	$P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A} - \mu\mathbf{1}\mathbf{1}^T/2 + \mathbf{I}/2)$
Triangle Count	$\frac{1}{6}\text{tr}(\mathbf{A}^3) - T$	$\mathbf{X}$	$P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A} - \mu\mathbf{A}^2/2)$
Degree	$\mathbf{A}\mathbf{1} - \delta_d\mathbf{1}$	$\mathbf{X}$	$P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A} - \frac{1}{2}(\boldsymbol{\mu}\mathbf{1}^T + \mathbf{1}\boldsymbol{\mu}^T) + \text{Diag}(\boldsymbol{\mu}))$
Valency	$\mathbf{A}\mathbf{1} - \mathbf{X}\mathbf{v}$	$P_{[\mathbf{0},\mathbf{1}]}(\mathbf{X})$	$P_{[\mathbf{0},\mathbf{3}]}(\mathbf{A} - \frac{1}{2}(\boldsymbol{\mu}\mathbf{1}^T + \mathbf{1}\boldsymbol{\mu}^T) + \text{Diag}(\boldsymbol{\mu}))$
Atom Count	$\mathbf{X}^T\mathbf{1} - \mathbf{c}$	$P_{[\mathbf{0},\mathbf{1}]}(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T)$	$P_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})$
Molecular Weight	$\mathbf{1}^T\mathbf{X}\mathbf{m} - W$	$P_{[\mathbf{0},\mathbf{1}]}(\mathbf{X} - \mu\mathbf{1}\mathbf{m}^T)$	$P_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})$
3D structure $\mathbf{G} = (\mathbf{X}, \mathbf{S})$. Attributes $\mathbf{X} \in [\mathbf{0}, \mathbf{1}]$, Positions $\mathbf{S} \in \mathbb{R}^{n\times 3}$			
Constraint ($\mathcal{C}$)	Function ($\mathbf{h}_{\mathcal{C}}$)	$\varphi_{\boldsymbol{\mu}}^X$	$\varphi_{\boldsymbol{\mu}}^S$
Dipole Moment	$\xi_0 \leq \lVert\mathbf{S}^T\mathbf{X}\mathbf{Q}\rVert_2 \leq \xi_1$	$\mathbf{X}$	$\mu\mathbf{S}/\lVert\mathbf{S}^T\mathbf{X}\mathbf{Q}\rVert_2$



(a) $t = T/10$ (b) $t = T/2$ (c) $t = 3T/4$ (d) $t = T$

Figure 3. Sampling process of PRODIGY (red) versus existing methods (GDSS (Jo et al., 2022), blue) at different diffusion timesteps (t) to generate points inside an ℓ_2 centered ball at the origin and a radius of 0.1.

this, we propose to take a partial γ_t step from $\widetilde{\mathbf{G}}_{t-1}$ towards the feasible set. In particular, we consider PRODIGY(β) as

$$\begin{cases} \widetilde{\mathbf{G}}_{t-1} \leftarrow \text{Reverse}(\mathbf{G}_t, \mathbf{s}_\theta(\mathbf{G}_t, t), \bar{\varepsilon}_t, t) \\ \mathbf{G}_{t-1} \leftarrow (1-\gamma_t)\,\widetilde{\mathbf{G}}_{t-1} + \gamma_t\,\Pi_{\mathcal{C}}(\widetilde{\mathbf{G}}_{t-1}), \end{cases} \quad (4)$$

where we consider two variants of γ_t. The first variant is based on the distance, *i.e.*, $\gamma_t = \exp(-\beta\, d_{\mathcal{C}}(\widetilde{\mathbf{G}}_{t-1})) := \exp(-\beta\lVert\widetilde{\mathbf{G}}_{t-1} - \Pi_{\mathcal{C}}(\widetilde{\mathbf{G}}_{t-1})\rVert_2)$ for some $\beta > 0$, which means we distort the original sampling process for constraint satisfaction more if its distance to the feasible set is less and vice versa. The second variant is based on the diffusion timestep, *i.e.*, $\gamma_t = (1-\gamma_0)(t/T)^p + \gamma_0$, giving more preference to projection at the later timesteps.

Note that $\mathbf{G}_{t-1}$, thus obtained, has a graph structure with continuous edge weights. For discrete models, we add a rounding step $\mathbf{G}_{t-1} \leftarrow \text{Round}(\mathbf{G}_{t-1})$ to obtain a discrete graph close to the feasible set after following the steps in Equation 4. For the continuous models, we follow existing works and round only at time 0. Rounding involves $\lfloor \mathbf{A}[i,j] \rceil$ for each element i, j to obtain the discrete adjacency matrix and $\arg\max_i \mathbf{X}[i]$ for the attribute of each node i.

How do we get the projection operators, $\Pi_{\mathcal{C}}$? The projection operators $\Pi_{\mathcal{C}}(\mathbf{G})$ efficiently transform a given graph $\mathbf{G}$ to its closest counterpart that satisfies a given constraint from the set of constraints discussed in Section 3.1.

4.1. Projection Operators

Consider a constraint of the form $\mathcal{C} = \{\mathbf{Z} = (\mathbf{Z}_X, \mathbf{Z}_A) \in \mathcal{G} : \mathbf{h}_{\mathcal{C}}(\mathbf{Z}) \leq \mathbf{0}\}$ on the set of graphs. Then, the projection operator is given as:

$$\Pi_{\mathcal{C}}(\mathbf{G}) = \underset{\substack{(\mathbf{Z}_X, \mathbf{Z}_A) \in \mathcal{G} \\ \mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) \leq \mathbf{0}}}{\arg\min} \ \tfrac{1}{2}\lVert\mathbf{Z}_X - \mathbf{X}\rVert_2^2 + \tfrac{1}{2}\lVert\mathbf{Z}_A - \mathbf{A}\rVert_2^2, \quad (5)$$

such that $\mathbf{Z}_X \in [\mathbf{X}_m, \mathbf{X}_M], \mathbf{Z}_A \in [\mathbf{A}_m, \mathbf{A}_M], \mathbf{Z}_A^T = \mathbf{Z}_A, \text{Diag}(\mathbf{Z}_A) = \mathbf{0}$. This can be solved using the Lagrangian method, $\mathcal{L}(\mathbf{Z}_X, \mathbf{Z}_A, \mathbf{h}_{\mathcal{C}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2}\lVert\mathbf{Z}_X - \mathbf{X}\rVert_2^2 + \frac{1}{2}\lVert\mathbf{Z}_A - \mathbf{A}\rVert_2^2 + \boldsymbol{\mu}_0 \cdot \mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) + \boldsymbol{\mu}_1 \cdot (\mathbf{Z}_X - \mathbf{X}_m) + \boldsymbol{\mu}_2 \cdot (\mathbf{X}_M - \mathbf{Z}_X) + \boldsymbol{\mu}_3 \cdot (\mathbf{Z}_A - \mathbf{A}_m) + \boldsymbol{\mu}_4 \cdot (\mathbf{A}_M - \mathbf{Z}_A) + \sum_{i>j}\lambda_{ij}(\mathbf{Z}_A[i,j] - \mathbf{Z}_A[j,i]) + \sum_i \lambda_i \mathbf{Z}_A[i]$. We apply the Karush–Kuhn–Tucker (KKT) conditions (Kuhn & Tucker, 2013) and find closed-form solutions for $\mathbf{Z}_X$ and $\mathbf{Z}_A$. For 3D structures, we consider the positions $\mathbf{S}$ instead of $\mathbf{A}$ with no additional constraints on $\mathbf{Z}_S$.

Table 3 lists the projection operators for different constraint functions. Please refer to Appendix B for the complete derivation of each case along with the computational complexity. We note that for several constraints, $\mathbf{h}_{\mathcal{C}}$ and $\boldsymbol{\mu}$ are scalars. Thus, we solve for μ in $h_{\mathcal{C}}(\varphi_\mu(\mathbf{G})) = 0$ using the bisection method (Boyd et al., 2004). When $\mathbf{h}_{\mathcal{C}}$ (and thus, $\boldsymbol{\mu}$) are vectors (as in the Degree, Valency, and Atom Count constraints), we split $\mathbf{h}_{\mathcal{C}}$ into independent functions $h_{\mathcal{C}}^{(i)}$ and solve for μ_i such that $h_{\mathcal{C}}^{(i)}(\varphi_{\mu_i}(\mathbf{G})) = 0$ using the bisection method. The split is done such that if $h_{\mathcal{C}}^{(i)}(\varphi_{\mu_i}(\mathbf{G})) = 0$ for all $i \in [1, M]$, then for $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_M)$, $\mathbf{h}_{\mathcal{C}}(\varphi_{\boldsymbol{\mu}}(\mathbf{G})) \leq \mathbf{0}$. Thus, the obtained solution would satisfy the constraint. For each constraint, $\varphi_{\boldsymbol{\mu}}$ involves matrix multiplication operations that can be efficiently done by exploiting data batching and parallelization of Pytorch (Paszke et al., 2019). Finding $\boldsymbol{\mu}$ is also efficient as the bisection method converges in a logarithmic number of steps and can exploit data batching strategies, thereby making the entire approach highly efficient.

5. Experiments

To test the efficacy of our method, we ask and investigate the following questions — **(1)** Can PRODIGY effectively constrain the generation of non-attributed graphs from existing diffusion models? **(2)** Can PRODIGY constrain 2D molecular graph generation from existing diffusion models? **(3)** How does PRODIGY compare against conditional denoising models for complex 3D molecular properties? **(4)** How does the PRODIGY sampling process affect the distributional properties learned by the original model? **(5)** How sensitive is our approach to its hyperparameters and the given constraint's parameters? **(6)** Can the generated graphs be visually explained under different constraints? **(7)** How time-efficient is the PRODIGY sampling process?

5.1. Setup

We briefly outline the experimental setup here and defer more details to Appendix D.

Datasets. We consider five non-attributed graph datasets including three real-world graphs: Community-small, Ego-small, Enzymes (Jo et al., 2022), and two synthetic graphs: SBM, Planar (Martinkus et al., 2022). We also use two standard molecular datasets: QM9 (Ramakrishnan et al., 2014), ZINC250k (Irwin et al., 2012).

Constraints. As noted in Section 3.1, we consider the constraints of edge count, triangle count, and degree for non-attributed graphs, while for molecular graphs, we use the constraints of valency, atom count, and molecular weight. Each constraint consists of an extrinsic constraint parameter that we vary to take different values from the test set.

Diffusion models. We consider five state-of-art representative diffusion models for graph generation to show how PRODIGY enables them to perform controlled graph generation: (1) **Continuous model**: EDP-GNN (Niu et al., 2020), GDSS (Jo et al., 2022), DruM (Jo et al., 2023), (2) **Discrete model**[2]: DiGress (Vignac et al., 2022) and (3) **3D graph generation model:** GeoLDM (Xu et al., 2023).

Metrics. We assess the performance of our method towards satisfying the given constraint and also report various distributional metrics. For the former, we consider the proportion of generated graphs that satisfy the constraint, *i.e.*, $\textsc{Val}_{\mathcal{C}}(\mathbf{G}) := \frac{1}{N}\sum_{i\in N}\mathbb{1}[\mathbf{G}_i \in \mathcal{C}]$, where we generate N different graphs $\{\mathbf{G}_i\}$. To evaluate the distributional preservation under our approach, we find how close the PRODIGY-generated graphs are to the test graphs $\mathcal{G}_{\mathcal{T}}$ as compared to the originally generated samples. In particular, we measure the closeness of graph samples using the maximum mean discrepancy (MMD) metric between the two distributions (You et al., 2018). We use the average of the MMD of degree, clustering coefficient, and orbit count between the graph sets. Then, we find the distributional preservation as the difference in MMD, $\Delta\text{MMD} := \text{MMD}(\mathcal{G}_M||\mathcal{G}_{\mathcal{T}}) - \text{MMD}(\mathcal{G}'_M||\mathcal{G}_{\mathcal{T}})$, where $\mathcal{G}_M, \mathcal{G}'_M$ denotes the generated graphs by the original sampling and PRODIGY respectively. Furthermore, for synthetic graphs, we also measure the V.U.N. metric that quantifies the proportion of unique and novel generated graphs that are valid based on the underlying graph's property (Martinkus et al., 2022). For molecules, we use the Fréchet ChemNet Distance (FCD) (Preuer et al., 2018) instead of the average MMD and use Δ FCD as a measure of distributional preservation, in addition to the molecular validity and novelty metrics. Note that to measure the distributional metrics (*i.e.*, Δ MMD and Δ FCD), we only consider constraint-feasible test graphs since we want to approximate the probability distribution with support $\mathcal{C}$. All the metrics are averaged over 3 random seeds with standard deviations of < 0.05 in MMD and < 0.01 for the $\textsc{Val}_{\mathcal{C}}$.

[2]We have omitted EDGE (Chen et al., 2023) due to unavailability of the checkpoints on our datasets.

Table 4. Constrained graph generation of real-world datasets using PRODIGY for different diffusion models and constraints. Hyperparameter values and raw MMDs are provided in Table 5.

Dataset	**Constraint**	**EDP-GNN**		**GDSS**		**DruM**	
		Δ MMD	$\textsc{Val}_{\mathcal{C}}$	Δ MMD	$\textsc{Val}_{\mathcal{C}}$	Δ MMD	$\textsc{Val}_{\mathcal{C}}$
Community small	Edge Count	0.07	0.52	−0.02	1.00	0.11	0.55
	Triangle Count	−0.01	0.83	0.04	0.90	0.01	0.30
	Degree	0.02	0.66	0.01	1.00	0.02	0.25
Ego small	Edge Count	0.22	0.64	0.22	0.63	0.29	0.65
	Triangle Count	0.02	0.98	−0.04	0.83	0.03	0.62
	Degree	0.09	0.73	0.04	0.65	0.15	0.55
Enzymes	Edge Count	−0.01	0.95	−0.32	0.82	0.04	0.77
	Triangle Count	−0.32	1.00	0.03	0.96	0.03	1.00
	Degree	0.00	1.00	0.23	1.00	0.08	0.80

Hyperparameters. We tune the hyperparameters to search for the optimal γ_t in Equation 4 to minimize the trade-off between constraint satisfaction and distributional preservation. In particular, we searched for the variables involved in these two functional forms, particularly, $\beta \in [0.1, 1.0, 10.0, 100.0], \gamma_0 \in [0, 0.1], p \in [0, 1, 5]$.

5.2. Constrained Non-Attributed Graph Generation

We first test the effectiveness of PRODIGY in sampling constrained graphs from non-attributed real-world graph distributions as trained by the three continuous diffusion models: EDP-GNN, GDSS, and DruM. To this end, we consider Edge Count, Triangle Count, and Degree constraints and choose the constraint parameters $(\mathcal{B}, T, \delta_d)$ such that the constraint is satisfied by only 10% of the test graphs. This means that the expected number of generated graphs from a perfect unconditional generative model that satisfies such constraints is only $\mathbf{0.1}$. Table 4 shows that PRODIGY enables the pre-trained models to satisfy these hard con-

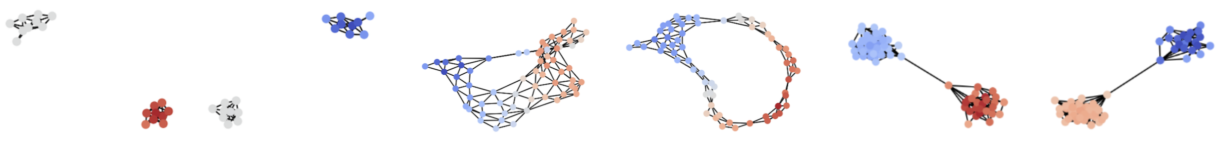

(a) Community-small ($|\mathbf{E}| \leq 21$)
(b) Planar ($|\mathbf{E}| \leq 177$)
(c) SBM ($|\mathbf{E}| \leq 700$)

Figure 4. Visualization of different graph datasets under the Edge Count constraint using DruM as the base diffusion model. We can note that the graphs are much sparser than those in the train dataset due to the constraint enforcement using PRODIGY.

Table 5. Raw average MMD scores and hyperparameters for the results on non-attributed real-world graph datasets. We also include the results for the graphs generated by base diffusion models to note any bias towards the constraint in the learned distribution. "+PRODIGY" indicates the diffusion model sampled using PRODIGY sampling. Note that poly$(a, b) = (1 - b)t^a + b$.

		Community-small			Ego-small			Enzymes		
		γ_t	Avg.↓	VAL_C ↑	γ_t	Avg.↓	VAL_C ↑	γ_t	Avg.↓	VAL_C ↑
Edge Count	EDP-GNN		0.18	0.43		0.23	0.23		0.08	0.56
	+PRODIGY	$\exp(-d_C)$	0.26	0.52	$\exp(-d_C)$	0.02	0.64	0.01	0.08	0.95
	GDSS		0.19	0.30		0.27	0.18		0.21	0.05
	+PRODIGY	$\exp(-d_C)$	0.21	1.00	$\exp(-d_C)$	0.05	0.62	poly(0,1)	0.54	0.82
	DruM		0.42	0.25		0.40	0.10		0.16	0.29
	+PRODIGY	$\exp(-d_C)$	0.31	0.55	$\exp(-d_C)$	0.11	0.65	$\exp(-d_C)$	0.19	0.77
Triangle Count	EDP-GNN		0.18	0.70		0.05	0.66		0.08	0.64
	+PRODIGY	$\exp(-100d_C)$	0.20	0.83	$\exp(-10d_C)$	0.03	0.98	0.01	0.39	1.00
	GDSS		0.19	0.70		0.02	0.80		0.16	0.03
	+PRODIGY	$\exp(-0.1d_C)$	0.14	0.90	$\exp(-d_C)$	0.06	0.82	poly(0.1,5)	0.13	0.96
	DruM		0.42	0.30		0.12	0.48		0.02	1.00
	+PRODIGY	$\exp(-d_C)$	0.40	0.30	$\exp(-10d_C)$	0.08	0.62	$\exp(-d_C)$	0.06	1.00
Degree	EDP-GNN		0.18	0.55		0.12	0.36		0.08	0.52
	+PRODIGY	$\exp(-100d_C)$	0.16	0.66	$\exp(-d_C)$	0.03	0.73	0.01	0.08	1.00
	GDSS		0.19	0.60		0.13	0.32		0.14	0.40
	+PRODIGY	$\exp(-d_C)$	0.17	1.00	$\exp(-d_C)$	0.09	0.65	poly(0,1)	0.36	1.00
	DruM		0.42	0.25		0.23	0.20		0.16	0.21
	+PRODIGY	$\exp(-d_C)$	0.40	0.25	$\exp(-10d_C)$	0.08	0.55	$\exp(-d_C)$	0.07	0.80

straints across different datasets by achieving constraint validity of up to 100% in the generated graphs and over 50% in all but 2 cases. Notably, this is achieved without deviating from the learned distributional properties, as Δ MMD either improves (+ve values in the table for Δ MMD) or remains similar, only decreasing slightly in some cases. We also provide a comparison of the raw metrics in Table 5 between the original and PRODIGY sampling along with the γ_t parameters used for each case.

Next, we study the effect of PRODIGY to constrain the generation in non-attributed synthetic datasets. Since EDP-GNN and GDSS show almost zero V.U.N. numbers on these datasets (Jo et al., 2023), we use more recent models, such as DruM and the discrete DiGress model for this experiment. Using a similar setup as above, Table 6 shows that PRODIGY enables the validity of different constraints (at least 70%) in these datasets while generating graphs that have a V.U.N. of over 50%. We observe a significant enhancement in the Planar dataset as we not only obtain high constraint validity but this happens with minimal change in the MMDs and a V.U.N. of up to 98%. Further, results on DiGress show that PRODIGY can be effectively used to constrain generation in discrete models.

Table 6. Constrained graph generation of synthetic datasets using PRODIGY for different diffusion models and constraints. Hyperparameter values and raw MMDs are provided in Appendix E.1.

Dataset	Constraint	DruM			DiGress		
		Δ MMD	V.U.N.	VAL_C	Δ MMD	V.U.N.	VAL_C
SBM	Edge Count	-0.28	0.38	1.00	0.01	0.60	0.65
	Triangle Count	-0.29	0.58	1.00	0.00	0.62	0.85
	Degree	-0.09	0.43	0.90	0.00	0.65	0.58
Planar	Edge Count	-0.08	0.80	1.00	0.00	0.88	0.75
	Triangle Count	-0.00	0.98	1.00	-0.00	0.75	1.00
	Degree	-0.02	0.92	0.70	0.00	0.70	0.75

We also evaluate the constrained graphs qualitatively under the Edge Count constraint on DruM. As the number of edges is bounded, the graphs are likely to be sparse and smaller. Figure 4 shows this is indeed the case for Community-small, Planar, and SBM. We provide visualizations on other datasets and constraints in Appendix E.3.

5.3. Constrained 2D Molecular Graph Generation

In this section, we test PRODIGY's effectiveness in constraining 2D molecular graphs from existing diffusion mod-

Table 7. Constrained 2D molecule generation using PRODIGY for different diffusion models and constraints. OOTM denotes that model checkpoints for these datasets were not provided in the original paper and we ran into out-of-memory error while training them from scratch. [*] The original models give low novelty on these datasets as well. For more details, see Appendix E.1.

Dataset	**Constraint**	**EDP-GNN**				**GDSS**				**DruM**				**DiGress**			
		Val.%	Novel.%	Δ FCD	$\text{VAL}_{\mathcal{C}}$	Val.%	Novel.%	Δ FCD	$\text{VAL}_{\mathcal{C}}$	Val.%	Novel.%[*]	Δ FCD	$\text{VAL}_{\mathcal{C}}$	Val.%	Novel.%[*]	Δ FCD	$\text{VAL}_{\mathcal{C}}$
QM9	Valency	96.29	76.68	−0.08	0.96	99.83	82.74	−0.35	0.99	100.00	22.6	−2.65	0.99	99.66	26.94	5.90	1.00
	Atom Count	98.06	54.36	2.97	1.00	95.02	67.67	5.68	1.00	100.00	15.9	7.32	1.00	99.84	30.05	0.94	0.65
	Mol. Weight	28.91	70.05	13.28	0.17	99.95	81.14	12.92	0.53	100.00	23.6	−3.71	0.65	99.84	34.90	−0.06	0.33
ZINC250k	Valency	OOTM	OOTM	OOTM	OOTM	99.88	100.00	−15.75	0.99	100.00	100.00	−5.62	0.98	OOTM	OOTM	OOTM	OOTM
	Atom Count	OOTM	OOTM	OOTM	OOTM	96.90	100.00	3.77	0.99	99.97	99.99	22.91	1.00	OOTM	OOTM	OOTM	OOTM
	Mol. Weight	OOTM	OOTM	OOTM	OOTM	97.66	100.00	−1.00	0.63	100.00	99.60	0.19	0.26	OOTM	OOTM	OOTM	OOTM

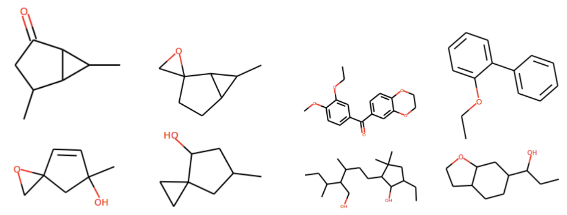

(a) QM9 (b) ZINC250k

Figure 5. 2D molecules generated under the Atom-Count constraint that allows only Carbon and Oxygen atoms from the DruM diffusion model. We pick the molecules based on the maximum Tanimoto similarity (Tanimoto, 1958) with the test set.

Table 8. Comparison of PRODIGY against specially-trained conditional denoising models for 3D molecule generation with constraints on dipole moment μ_{dm}. Atom. and Mol. Stability (%) denotes atomic and molecular stability.

	Atom. Stability (%) ↑	Mol. Stability (%) ↑	MAE μ_{dm} ↓
EDM (μ_{dm}-conditional)	98.7	82.0	1.11 (0.04)
GeoLDM (μ_{dm}-conditional)	98.9	89.4	1.10 (0.04)
GeoLDM (unconditional) **+ PRODIGY**	98.9	89.4	**0.00** (1.15)

els. We consider all four diffusion models and constraints of valency, atom count, and molecular weight. For the valency constraint, we consider the valencies $C_4N_5O_2F_1$ in QM9 and $C_4N_3O_2F_1P_5S_2Cl_1Br_1I_1$ in ZINC250k. For the Atom Count, we constrained the generated molecule to only contain C and O for both QM9 and ZINC250k. Lastly, we constrain the molecular weight of the generated molecules to be within the lower 10-percentile range of the test set.

Table 7 shows PRODIGY's impact on the generation of these molecules under different constraints and diffusion models. PRODIGY provides a high constraint validity in all the cases except for molecular weight in some models. Furthermore, this is achieved with minimal change in the distributional metric of FCD, which we improve in many cases. Figure 5 visualizes the generated molecules under the Atom Count constraint and we find that they only contain C and O. We further also investigated the failure cases in molecular weight by first noting that the hyperplane projection in this constraint tends to a solution with negative values in the $\mathbf{X}$. We find that adding a trivial lower bound helps to achieve a higher validity in all cases than 0.1, as expected by a perfect unconditional model. But the numbers are still lower than what we achieve for the other constraints and we leave further improvement as future directions.

5.4. 3D Molecule Generation for Complex Properties

Here, we evaluate if PRODIGY is effective in generating 3D molecules under constraints on their dipole moment (as formulated in Sections 3.1, 4.1). We consider a 3D latent diffusion model, GeoLDM, that learns a diffusion process in the latent space of node types $\mathbf{L}_X^{\text{GeoLDM}}$ and positions $\mathbf{L}_S^{\text{GeoLDM}}$. Here, instead of projecting in the graph space, we directly project these embedded values to the constrained space instead of extracting the X and S from them respectively. Here, we had assumed that if the constraint is satisfied in the latent space, it will also be satisfied in the original space. Since it is hard to find a dipole moment from just the structure, we first formulate a simple proxy for it and constrain this estimation in the latent space. In particular, we assume that the induced charge for a particular atom type is fixed and does not depend on the surrounding structure. Then, the dipole moment is given as $\hat{\mu}_{dm} = \|\mathbf{S}(i)^T \mathbf{X}(i) \mathbf{Q}^*\|_2$. We consider the charges as learnable parameters that are found by minimizing the ℓ_1 loss between the actual μ_{dm} and $\hat{\mu}_{dm}$ on the training set.

We then use PRODIGY on a 3D unconditional GeoLDM (Xu et al., 2023) for QM9 while constraining the estimated dipole moment. Following existing works on conditional denoising models (Hoogeboom et al., 2022), we chose the constraint parameters $[\xi_0, \xi_1]$ such that the estimated dipole moment lies within one standard deviation from the mean of the 2nd half of the training set. Table 8 shows the MAE of the generated molecules $= (1 - \text{VAL}_{\mathcal{C}})$ along with the bias of the predicted model, i.e. MAE (μ, $\hat{\mu}_{dm}$), in the parentheses. We note that even though our simple approximation of the dipole moment is highly biased, our effective constraint satisfaction during sampling enables unconditional models to give competitive errors to models that are specifically trained for the dipole moment. Thus, we can match the performance of state-of-the-art conditional models without requiring retraining of the diffusion models.

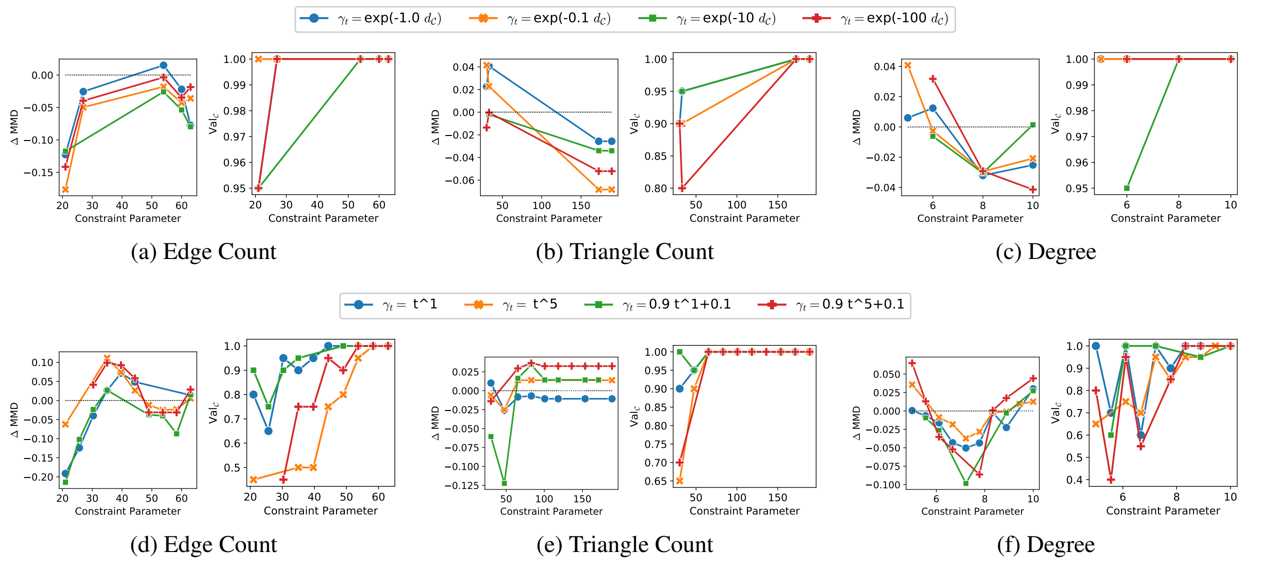


Figure 6. Comparison of PRODIGY at different hyperparameters and constraint parameters in generating Community-small graphs for different constraints. The upper row compares distance-based scheduling of γ_t while the lower row compares time-based scheduling. Note that the lack of a data point is when sampling leads to a trivial solution of zero edges or an empty graph.

5.5. Flexibility of PRODIGY

Our method allows for an arbitrary constraint satisfaction (*i.e.* for any constraint parameter) and an interpretable tuning hyperparameter γ_t. Figure 6 compares PRODIGY with different hyperparameters for a range of constraint parameters belonging to the test set. This shows that our method can support a wide range of constraint parameters with appropriate tuning of the γ_t values. As mentioned in Section 4, γ_t can control the tradeoff between preserving the distributional statistics and the validity of the given constraint. For example, giving more weight to smaller distances can be effective in more constrained settings (*i.e.*, smaller constraint parameter value), while a higher weight may be more effective in less constrained settings. Similarly, a slowly growing γ_t may be more effective in preserving the statistics but comes at the cost of low constraint validity.

5.6. Running Time

We show that the PRODIGY sampling time is similar to the original sampling time for different constraints and datasets for GDSS while noting similar trends in other diffusion models. In particular, Table 9 reports the sampling time taken per diffusion timestep. This shows that the projection step doesn't change the scale of the diffusion sampling and the time taken is mostly minimal as compared to the denoising step. This is expected from the linear/logarithmic time complexity of these projection operators that were proved in Table 3 and Appendix B.

Table 9. Time taken (in seconds) per diffusion timestep. * denotes the time taken by the original (unconstrained) GDSS sampling.

	Original*	Edge Count	Triangle Count	Degree
Community-small	0.47	0.58	0.51	0.57
Ego-small	0.04	0.13	0.07	0.13
Enzymes	0.07	0.41	0.11	0.22

6. Discussion and Conclusion

We proposed PRODIGY, the first plug-and-play approach to controllable graph generation with diffusion models. Our work enables precise control of the graph generation process under arbitrary well-specified and hard constraints, thereby making it applicable to a wide range of real-world applications including network design and drug discovery. PRODIGY supports **hard, non-differentiable constraints**, does **not** require model retraining, and does **not** depend on additional labeled curated datasets. We hope that this opens future research avenues for enabling interpretable control in the generative models across different domains. Future directions include extending our method to enable control of more complex non-linear properties of the graphs, *e.g.* GNN-based molecular property prediction, and we discuss some first steps in more detail in Appendix C. Finally, we also believe there is a lot of scope for future works to provide controlled generation for precise control in latent diffusion models. Since the constraints will be provided in the graph space, it is not clear how should one constrain the corresponding latent variables to ensure their satisfaction.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

This work is supported in part by NSF grants CNS-2154118, IIS-2027689, ITE-2137724, ITE-2230692, CNS-2239879, Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112290102 (subcontract No. PO70745), and funding from Microsoft, Google, and The Home Depot. We also appreciate the valuable feedback of the anonymous reviewers which has helped us to improve the quality of our exposition.

References

Anderson, B. D. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

Bar-Tal, O., Yariv, L., Lipman, Y., and Dekel, T. Multi-diffusion: Fusing diffusion paths for controlled image generation. *arXiv preprint arXiv:2302.08113*, 2, 2023.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Bubeck, S., Eldan, R., and Lehec, J. Sampling from a log-concave distribution with projected langevin monte carlo. *Discrete & Computational Geometry*, 59(4):757–783, 2018.

Chen, X., He, J., Han, X., and Liu, L.-P. Efficient and degree-guided graph generation via discrete diffusion modeling. *arXiv preprint arXiv:2305.04111*, 2023.

Farahani, R. Z., Miandoabchi, E., Szeto, W. Y., and Rashidi, H. A review of urban transportation network design problems. *European journal of operational research*, 229 (2):281–302, 2013.

Graikos, A., Malkin, N., Jojic, N., and Samaras, D. Diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 35:14715–14728, 2022.

Grover, A., Zweig, A., and Ermon, S. Graphite: Iterative generative modeling of graphs. In *International conference on machine learning*, pp. 2434–2444. PMLR, 2019.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022.

Hsieh, Y.-P., Kavis, A., Rolland, P., and Cevher, V. Mirrored langevin dynamics. *Advances in Neural Information Processing Systems*, 31, 2018.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

Jo, J., Lee, S., and Hwang, S. J. Score-based generative modeling of graphs via the system of stochastic differential equations. *arXiv preprint arXiv:2202.02514*, 2022.

Jo, J., Kim, D., and Hwang, S. J. Graph generation with destination-driven diffusion mixture. *arXiv preprint arXiv:2302.03596*, 2023.

Kuhn, H. W. and Tucker, A. W. Nonlinear programming. In *Traces and emergence of nonlinear programming*, pp. 247–258. Springer, 2013.

Lee, S., Jo, J., and Hwang, S. J. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, pp. 18872–18892. PMLR, 2023.

Li, X., Thickstun, J., Gulrajani, I., Liang, P. S., and Hashimoto, T. B. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343, 2022.

Liu, G.-H., Chen, T., Theodorou, E. A., and Tao, M. Mirror diffusion models for constrained and watermarked generation. *arXiv preprint arXiv:2310.01236*, 2023.

Liu, J., Kumar, A., Ba, J., Kiros, J., and Swersky, K. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Ninniri, M., Podda, M., and Bacciu, D. Classifier-free graph diffusion for molecular property targeting. *arXiv preprint arXiv:2312.17397*, 2023.

Niu, C., Song, Y., Song, J., Zhao, S., Grover, A., and Ermon, S. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pp. 4474–4484. PMLR, 2020.

OpenAI. Gpt-4 technical report, 2023.

Parikh, N., Boyd, S., et al. Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239, 2014.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Preuer, K., Renz, P., Unterthiner, T., Hochreiter, S., and Klambauer, G. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018.

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

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

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

Song, J., Zhang, Q., Yin, H., Mardani, M., Liu, M.-Y., Kautz, J., Chen, Y., and Vahdat, A. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, pp. 32483–32498. PMLR, 2023.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Tabourier, L., Roth, C., and Cointet, J.-P. Generating constrained random graphs using multiple edge switches. *Journal of Experimental Algorithmics (JEA)*, 16:1–1, 2011.

Tanimoto, T. T. Elementary mathematical theory of classification and prediction. 1958.

Todeschini, R. and Consonni, V. *Handbook of molecular descriptors*. John Wiley & Sons, 2008.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Xie, S., Kirillov, A., Girshick, R., and He, K. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1284–1293, 2019.

Xu, M., Powers, A. S., Dror, R. O., Ermon, S., and Leskovec, J. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pp. 38592–38610. PMLR, 2023.

Yan, Q., Liang, Z., Song, Y., Liao, R., and Wang, L. Swingnn: Rethinking permutation invariance in diffusion models for graph generation. *arXiv preprint arXiv:2307.01646*, 2023.

Yang, N., Wu, H., Yan, J., Pan, X., Yuan, Y., and Song, L. Molecule generation for drug design: a graph learning perspective. *arXiv preprint arXiv:2202.09212*, 2022.

Ying, X. and Wu, X. Graph generation with prescribed feature constraints. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pp. 966–977. SIAM, 2009.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. Graphrnn: Generating realistic graphs with deep autoregressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Appendix

A. Diffusion Models

A.1. Continuous Diffusion

Continuous-time diffusion models have demonstrated significant success in generating graphs for various purposes (Niu et al., 2020; Jo et al., 2022; 2023). These models are based on the idea of smoothly diffusing an unknown target distribution towards a fixed noise distribution (typically Gaussian) so that one can reverse it back to sample true data from noise. Given a graph $\mathbf{G}(0) \sim p_0$, the method follows a 'forward SDE' to gradually convert the graph into Gaussian noise, *i.e.*, $\mathbf{G}(T) \sim p_T = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, for a fixed $\boldsymbol{\mu}, \boldsymbol{\Sigma}$.

$$\mathrm{d}\mathbf{G} = \mathbf{f}(\mathbf{G}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}, \tag{6}$$

where $\mathbf{f} : \mathcal{G} \times t \to \mathcal{G}$ is the *drift coefficient*, $g : \mathbb{R} \to \mathbb{R}$ is the *diffusion coefficient*, and $\mathbf{w}(t) \in \mathcal{G}$ is a standard Wiener process. In order to generate samples from the unknown data distribution p_0, the forward process is reversed so that samples from the prior distribution can be converted to the target distribution. (Anderson, 1982) shows that the reverse process can be given as:

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

where $\bar{\mathbf{w}}$ is a reverse-time standard Wiener process and p_t denotes the marginal distribution of $\mathbf{G}_t$ at time-step t. $\nabla_{\mathbf{G}} \log p_t(\mathbf{G})$ is called the score function at time t.

Since the time-conditional score function is not available for an unknown distribution p_0, one estimates it with a parameterized neural network $\mathbf{s}_\theta(\mathbf{G}, t) \approx \nabla_{\mathbf{G}} \log p_t(\mathbf{G})$ by minimizing a score-matching objective across multiple time steps and training samples. EDP-GNN (Niu et al., 2020) ignores the diffusion process of $\mathbf{X}$ and samples directly from the prior distribution of $\mathbf{X}$. GDSS (Jo et al., 2022) considers a system of SDEs to efficiently estimate score functions for $\mathbf{X}$ and $\mathbf{A}$ separately. DruM (Jo et al., 2023) models the graph topology by conditioning the process on the destination data distribution using a mixture of Ornstein-Uhlenbeck processes. These models have also been proposed to predict structures in the 3-dimensional space by generating the positions $\mathbf{S}$ and types of each node in the 3D space (Hoogeboom et al., 2022; Xu et al., 2023).

A.2. Discrete Diffusion

Discrete-diffusion models have been recently proposed to model discrete structures such as graphs directly by diffusing with a discrete noise (Austin et al., 2021; Vignac et al., 2022). Each node and edge of a graph is assumed to have a fixed number of states and the model learns a categorical probability distribution for each node and edge. For example, node states can denote different atom types in a molecule, while edge states can denote whether a connection is a single, double, or triple bond. In particular, we consider the graph $\mathbf{G} = (\mathbf{X}, \mathbf{E})$ to be a multivariate random variable that has a categorical probability distribution in each variable.

The 'forward step' here corresponds to adding a pre-defined multivariate categorical noise to a given graph $\mathbf{G}_0$ followed by sampling from the updated probability distribution. This maintains that the noisy graph obtained at each step is a discrete graph structure. We repeat for T steps until we obtain a uniform distribution over these categories. The 'reverse step' then corresponds to retrieving the original graph after multiple rounds of denoising, starting from a random graph that is uniformly distributed over the categories. DiGress (Vignac et al., 2022) assumes an independent distribution of each node and edge at each step and learns it using a Graph Transformer. Then, it uses this distribution to find the posterior distribution given the graph in the previous time step.

B. Projection Operators

In this section, we discuss projection operators for the constraints mentioned in Table 3. We first solve for $\varphi^A_{\boldsymbol{\mu}} = \mathbf{Z}^*_A$ and $\varphi^X_{\boldsymbol{\mu}} = \mathbf{Z}^*_X$. Then, we propose a way to solve $\mathbf{h}_{\mathcal{C}}(\varphi_{\boldsymbol{\mu}}(\mathbf{G})) = \mathbf{0}$. Further, we replace $\boldsymbol{\mu}_0$ with $\boldsymbol{\mu}$ in the Lagrangian without loss of generality.

KKT conditions. The optimal solution $\mathbf{Z}^*$ for the problem must satisfy the following conditions:

1. *Stationarity.* $\nabla_{\mathbf{Z}}\mathcal{L}|_{\mathbf{Z}^*} = \mathbf{0} \implies \mathbf{Z}^*_X - \mathbf{X} + \boldsymbol{\mu}_0 \nabla_{\mathbf{Z}_X} \mathbf{h}_{\mathcal{C}}(\mathbf{Z}^*_X, \mathbf{Z}^*_A) + \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 = \mathbf{0}$ such that $\mathbf{Z}^*_X \in [\mathbf{X}_m, \mathbf{X}_M]$ and $\mathbf{Z}^*_A - \mathbf{A} + \boldsymbol{\mu}_0 \nabla_{\mathbf{Z}_A} \mathbf{h}_{\mathcal{C}}(\mathbf{Z}^*_X, \mathbf{Z}^*_A) + \boldsymbol{\mu}_3 - \boldsymbol{\mu}_4 + \boldsymbol{\Lambda} = \mathbf{0}$ such that $\mathbf{Z}^*_A \in [\mathbf{A}_m, \mathbf{A}_M]$ and $\Lambda[i, j] = \lambda_{ij}$ if $i > j$, λ_i if

$i = j$, and $-\lambda_{ij}$ otherwise. It is hard to solve this system of equations simultaneously as $\nabla \mathbf{h}_{\mathcal{C}}$ can be non-linear so we assume either $\nabla_{\mathbf{Z}_A} \mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X^*, \mathbf{Z}_A^*) = \mathbf{0}$ or $\nabla_{\mathbf{Z}_X} \mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X^*, \mathbf{Z}_A^*) = \mathbf{0}$ depending on the form of $\mathbf{h}_{\mathcal{C}}(\mathbf{X}, \mathbf{A})$.

2. *Primal and Dual feasibility.* $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3, \boldsymbol{\mu}_4 \geq \mathbf{0}$, $\mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X^*, \mathbf{Z}_A^*) \leq \mathbf{0}$, $\mathbf{Z}_X^* \in [\mathbf{X}_m, \mathbf{X}_M]$, $\mathbf{Z}_A^* \in [\mathbf{A}_m, \mathbf{A}_M]$, $(\mathbf{Z}_A^*)^T = \mathbf{Z}_A^*$, $\text{Diag}(\mathbf{Z}_A^*) = \mathbf{0}$.

3. *Complementary Slackness (CS).* $\boldsymbol{\mu}_0 \mathbf{h}_{\mathcal{C}}(\mathbf{Z}^*) = \mathbf{0}$, $\boldsymbol{\mu}_1(\mathbf{Z}_X^* - \mathbf{X}_m) = \mathbf{0}$, $\boldsymbol{\mu}_2(\mathbf{X}_M - \mathbf{Z}_X^*) = \mathbf{0}$, $\boldsymbol{\mu}_3(\mathbf{Z}_A^* - \mathbf{A}_m) = \mathbf{0}$, $\boldsymbol{\mu}_4(\mathbf{A}_M - \mathbf{Z}_A^*) = \mathbf{0}$.

First, we note that $\boldsymbol{\mu}_0 \mathbf{h}_{\mathcal{C}}(\mathbf{Z}^*) = \mathbf{0}$, $\boldsymbol{\mu}_0 \geq \mathbf{0}$, and $\mathbf{h}_{\mathcal{C}}(\mathbf{Z}^*) \leq \mathbf{0}$ imply that if $\mathbf{h}_{\mathcal{C}}(\mathbf{Z}^*(\boldsymbol{\mu}_0 = \mathbf{0})) \leq \mathbf{0}$ then $\boldsymbol{\mu}_0 = \mathbf{0}$ otherwise we find $\boldsymbol{\mu}_0 \geq \mathbf{0}$ such that $\mathbf{h}_{\mathcal{C}}(\mathbf{Z}^*(\boldsymbol{\mu}_0)) = \mathbf{0}$.

We also note that $\boldsymbol{\mu}_{1,2}$ can be replaced by a clamp operation $P_{[\cdot,\cdot]}$ that clamps $\mathbf{Z}_X^*$ within $[\mathbf{X}_m, \mathbf{X}_M]$. This is because if a certain entry of $\mathbf{Z}_X^*$ is within the range, then, the corresponding $\mu = 0$ (due to CS), and if not, we add/subtract a $\mu \geq 0$ such that $\mathbf{Z}_X^* = \mathbf{X}_m$ or $\mathbf{X}_M$ (CS). Similarly, we also note that $\boldsymbol{\mu}_{3,4}$ can be replaced by a clamp operation that clamps $\mathbf{Z}_A^*$ within $[\mathbf{A}_m, \mathbf{A}_M]$.

Thus, we can find $\mathbf{Z}_X^*$ and $\mathbf{Z}_A^*$ as $\Pi_{\mathcal{C}}(\mathbf{G}) = \varphi_{\mathbf{0}}(\mathbf{G})$ if $\mathbf{h}_{\mathcal{C}}(\varphi_{\mathbf{0}}(\mathbf{G})) \leq 0$, otherwise $\varphi_{\boldsymbol{\mu}}(\mathbf{G})$ such that $\mathbf{h}_{\mathcal{C}}(\varphi_{\boldsymbol{\mu}}(\mathbf{G})) = 0$. Here, $\varphi_{\boldsymbol{\mu}} = (\varphi_{\boldsymbol{\mu}}^X, \varphi_{\boldsymbol{\mu}}^A)$ can be found for the following two cases:

1. $\nabla_{\mathbf{Z}_A} \mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X^*, \mathbf{Z}_A^*) = \mathbf{0}$: We get $\mathbf{Z}_A^* = P_{[\mathbf{A}_m, \mathbf{A}_M]}(\mathbf{A} - \boldsymbol{\Lambda})$ such that $(\mathbf{Z}_A^*)^T = \mathbf{Z}_A^*$, $\text{Diag}(\mathbf{Z}_A^*) = \mathbf{0}$. We assume that the input $\mathbf{A}$ is undirected and has no self-loops, then, $\boldsymbol{\Lambda} = \mathbf{0}$ would be feasible. Thus, we get $\mathbf{Z}_A^* = \varphi_{\boldsymbol{\mu}}^A(\mathbf{G}) = P_{[\mathbf{A}_m, \mathbf{A}_M]}(\mathbf{A})$. We can find $\varphi_{\boldsymbol{\mu}}^X$ by solving for $\mathbf{Z}_X^*$ in the equation $\mathbf{Z}_X^* + \boldsymbol{\mu}_0 \nabla_{\mathbf{Z}_X} \mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X^*, \mathbf{Z}_A^*) = \mathbf{X}$ and then, clamping it within $[\mathbf{X}_m, \mathbf{X}_M]$.

2. $\nabla_{\mathbf{Z}_X} \mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X^*, \mathbf{Z}_A^*) = \mathbf{0}$: We get $\mathbf{Z}_X^* = \varphi_{\boldsymbol{\mu}}^X(\mathbf{G}) = P_{[\mathbf{X}_m, \mathbf{X}_M]}(\mathbf{X})$. We can find $\varphi_{\boldsymbol{\mu}}^A$ by solving for $\mathbf{Z}_A^*$ in the equation $\mathbf{Z}_A^* + \boldsymbol{\mu}_0 \nabla_{\mathbf{Z}_A} \mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X^*, \mathbf{Z}_A^*) + \boldsymbol{\Lambda} = \mathbf{A}$ and then, clamping it within $[\mathbf{A}_m, \mathbf{A}_M]$, while satisfying $(\mathbf{Z}_A^*)^T = \mathbf{Z}_A^*$ and $\text{Diag}(\mathbf{Z}_A^*) = \mathbf{0}$.

B.1. Edge Count ($|\mathbf{E}| \leq \mathcal{B}$)

Find $\varphi_{\boldsymbol{\mu}}$. We have $\mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) = h_{\mathcal{C}}(\mathbf{Z}_A) = \frac{1}{2}\mathbf{1}^T\mathbf{Z}_A\mathbf{1} - \mathcal{B}$, $\mathbf{Z}_A \in [\mathbf{0}, \mathbf{1}], \text{Diag}(\mathbf{Z}_A) = \mathbf{0}, \mathbf{Z}_A^T = \mathbf{Z}_A$. Then, we can note that $\nabla_{\mathbf{Z}_X} h_{\mathcal{C}} = \mathbf{0}$. Thus, we solve for $\mathbf{Z}_A^*$ in $\mathbf{Z}_A^* + \mu \nabla_{\mathbf{Z}_A} h_{\mathcal{C}}(\mathbf{Z}_A^*) + \boldsymbol{\Lambda} = \mathbf{A}$. Since $\nabla_{\mathbf{Z}_A} h_{\mathcal{C}} = \frac{1}{2}\mathbf{1}\mathbf{1}^T$, we get $\mathbf{Z}_A^* = \mathbf{A} - \frac{\mu}{2}\mathbf{1}\mathbf{1}^T - \boldsymbol{\Lambda}$. Satisfying $\text{Diag}(\mathbf{Z}_A^*) = 0, (\mathbf{Z}_A^*)^T = \mathbf{Z}_A^*$ (given these conditions hold for $\mathbf{A}$) implies $\Lambda_{ii} = -1/2$ and $\Lambda_{ij} = \Lambda_{ji} = 0$. In other words, $\boldsymbol{\Lambda} = \mathbf{I}/2$. Thus, $\mathbf{Z}_A^* = \mathbf{A} - \mu/2\mathbf{1}\mathbf{1}^T + \mu/2\mathbf{I}$ followed by clamping between $[\mathbf{0}, \mathbf{1}]$.

Find $\boldsymbol{\mu}$. To find μ, we can do a bisection method between $\max\{0, 2(\min(\mathbf{A}) - 1)\}$ and $2\max(\mathbf{A})$. This is because $\frac{1}{2}\mathbf{1}^T P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A} - (\min(\mathbf{A}) - 1)\mathbf{1}\mathbf{1}^T + (\min(\mathbf{A}) - 1))\mathbf{1} = \binom{|\mathcal{V}|}{2} \geq \mathcal{B}$ and $\frac{1}{2}\mathbf{1}^T P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A} - \max(\mathbf{A})\mathbf{1}\mathbf{1}^T + \max(\mathbf{A})\mathbf{I})\mathbf{1} = \frac{1}{2}\mathbf{1}^T\mathbf{0}\mathbf{1} = 0 \leq \mathcal{B}$.

Complexity. The bisection method finishes in $O(\log(\max(\mathbf{A}) - \max\{0, (\min(\mathbf{A}) - 1)\})/\xi) = O(\log(\frac{1}{\xi}))$ for a tolerance level ξ, since $\mathbf{A} \in [\mathbf{0}, \mathbf{1}]$. Finding $\mathbf{Z}_A^*$ involves only matrix operations (addition) that have been highly optimized in Pytorch with the worst-case time complexity of $O(n^2)$. Thus, we get the time complexity of the projection operator as $O(n^2 \log(\frac{1}{\xi}))$.

B.2. Triangle Count ($|\triangle| = \frac{1}{6}\mathbf{tr}(\mathbf{A}^3) \leq T$)

Find $\varphi_{\boldsymbol{\mu}}$. We have $\mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) = h_{\mathcal{C}}(\mathbf{Z}_A) = \frac{1}{6}\text{tr}(\mathbf{Z}_A^3) - T$, $\mathbf{Z}_A \in [\mathbf{0}, \mathbf{1}], \text{Diag}(\mathbf{Z}_A) = \mathbf{0}, \mathbf{Z}_A^T = \mathbf{Z}_A$. Then, we can note that $\nabla_{\mathbf{Z}_X} h_{\mathcal{C}} = \mathbf{0}$. Thus, we solve for $\mathbf{Z}_A^*$ in $\mathbf{Z}_A^* + \mu \nabla_{\mathbf{Z}_A} h_{\mathcal{C}}(\mathbf{Z}_A^*) + \boldsymbol{\Lambda} = \mathbf{A}$. Since $\nabla_{\mathbf{Z}_A} h_{\mathcal{C}} = \frac{1}{2}\mathbf{Z}_A^2$, we get $\mathbf{Z}_A^* + \frac{\mu}{2}(\mathbf{Z}_A^*)^2 + \boldsymbol{\Lambda} = \mathbf{A}$. Satisfying $\text{Diag}(\mathbf{Z}_A^*) = \mathbf{0}, (\mathbf{Z}_A^*)^T = \mathbf{Z}_A^*$ (given these hold for $\mathbf{A}$) implies $\boldsymbol{\Lambda} = \mathbf{0}$. Thus, $\mathbf{Z}_A^* + \frac{\mu}{2}(\mathbf{Z}_A^*)^2 = \mathbf{A}$. Let us assume $\mathbf{Z}_A^* \approx \mathbf{A}^2$, *i.e.*, the squared values do not change a lot after projection. Then, we get $\mathbf{Z}_A^* \approx \mathbf{A} - \frac{\mu}{2}\mathbf{A}^2$.

Find $\boldsymbol{\mu}$. We will find for μ using the bisection method here as well. But it is non-trivial to obtain two points for which $\frac{1}{6}\text{tr}(P_{[\mathbf{0},\mathbf{1}]}((\mathbf{A} - \frac{\mu}{2}\mathbf{A}^2)^3)) - T$ have opposite signs. Thus, we assume that one such point is $\mu = 0$ and search for the first point > 0 with an opposite sign using a linear search from μ with a fixed step size s. Then, we apply the bisection method between 0 and the new point μ_1 found using the linear search.

Complexity. Linear search computes $\mathbf{Z}_A^*$ for $(\mu_1 - 0)/s$ times to compare with value at $\mu = 0$. The bisection method finishes in $O(\log(\mu_1/\xi))$ time. Again, finding $\mathbf{Z}_A^*$ involves only matrix operations (addition) that have been highly optimized in

Pytorch with the worst-case time complexity of $O(n^3)$. Thus, we get the time complexity of the projection operator as $O(n^3(\mu_1/s + \log(\mu_1/\xi)))$.

B.3. Degree ($d_{\max} = \mathbf{A1} \leq \delta_d \mathbf{1}$)

***Find* $\varphi_{\boldsymbol{\mu}}$.** We have $\mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) = \mathbf{Z}_A\mathbf{1} - \delta_d\mathbf{1}, \mathbf{Z}_A \in [\mathbf{0}, \mathbf{1}], \text{Diag}(\mathbf{Z}_A) = \mathbf{0}, \mathbf{Z}_A^T = \mathbf{Z}_A$. Then, we can note that $\nabla_{\mathbf{Z}_X}\mathbf{h}_{\mathcal{C}} = \mathbf{0}$ and we solve for $\mathbf{Z}_A^*$ in $\mathbf{Z}_A^* + \boldsymbol{\mu}\cdot\nabla_{\mathbf{Z}_A}\mathbf{h}_{\mathcal{C}}(\mathbf{Z}_A^*) + \boldsymbol{\Lambda} = \mathbf{A}$. In other words, for each row i, we get $\mathbf{Z}_A^*[i,:] = \mathbf{A}[i,:] - \mu_i\mathbf{1} - \boldsymbol{\Lambda}[i,:]$ since $\nabla_{\mathbf{Z}_A} h_{\mathcal{C}}^{(i)} = \mathbf{1}$. Due to symmetricity, we obtain $\mathbf{A}[i,j] - \mu_i - \boldsymbol{\Lambda}[i,j] = \mathbf{A}[j,i] - \mu_j - \boldsymbol{\Lambda}[j,i]$ for all i, j, which gives us $\mu_i + \boldsymbol{\Lambda}[i,j] = \mu_j + \boldsymbol{\Lambda}[j,i]$. We can thus let $\boldsymbol{\Lambda}[i,j] = \frac{1}{2}(\mu_j - \mu_i)$ for all $i \neq j$. For the diagonal entries, we want $\boldsymbol{\Lambda}[i,i] = -\mu_i$ so that $\mathbf{Z}_A^*$ has no non-zero diagonal entries. Thus, we get $\mathbf{Z}_A^* = \mathbf{A} - \frac{1}{2}(\boldsymbol{\mu}\mathbf{1}^T + \mathbf{1}\boldsymbol{\mu}^T) + \text{Diag}(\boldsymbol{\mu})$ followed by clamping between $[\mathbf{0}, \mathbf{1}]$.

***Find* $\boldsymbol{\mu}$.** Since $\mathbf{h}_{\mathcal{C}}$ is a vector function, we cannot find its root using the bisection method. Instead, we divide $\mathbf{h}_{\mathcal{C}}(\varphi_{\boldsymbol{\mu}}) = \mathbf{0}$ into multiple equations $\tilde{h}_{\mathcal{C}}^{(i)}(\varphi_{\mu_i}) = 0$ that we can solve independently such that the root $\tilde{\boldsymbol{\mu}}$ obtained by concatenating these $\tilde{\mu_i}$s satisfies $\mathbf{h}_{\mathcal{C}}(\varphi_{\tilde{\boldsymbol{\mu}}}) \leq \mathbf{0}$.

In particular, we can just solve each row μ_i's equation separately and add it later to satisfy the symmetricity. Thus, we have to solve for $\tilde{\mu}_i \geq 0$ such that $\mathbf{1}^T P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A}[i,:] - \tilde{\mu}_i\mathbf{1}) = \delta_d$. Thus, we solve for $\tilde{\mu}_i$ for all i and use it to find $\tilde{\boldsymbol{\mu}}$ using the bisection method between $\max\{0, 2(\min(\mathbf{A}[i,:]) - 1)\}$ and $2\max(\mathbf{A}[i,:])$ (due to the same logic as for Edge Count constraint). Note that if $\mathbf{1}^T P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A}[i,:] - \tilde{\mu}_i\mathbf{1}) = \delta_d$, then $\mathbf{1}^T P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A}[i,:] - (\tilde{\mu}_i + \epsilon)\mathbf{1}) \leq \delta_d$, for all $\epsilon \geq 0$, because $(\tilde{\mu}_i + \epsilon) \geq \tilde{\mu}_i$ and it is a decreasing function. We have ϵ_i to be $\tilde{\mu}_j$ for different columns j. Thus, $P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A} - \frac{2}{2}(\tilde{\boldsymbol{\mu}}\mathbf{1}^T + \mathbf{1}\tilde{\boldsymbol{\mu}}^T) + 2\text{Diag}(\tilde{\boldsymbol{\mu}}))\mathbf{1} \leq \delta_d\mathbf{1}$.

Complexity. We solve n different equations using the bisection method in time $O(\log(\frac{1}{\xi}))$ as $\mathbf{A} \in [\mathbf{0}, \mathbf{1}]$. Note that this can be done in a parallel manner by using the Pytorch functionalities. Again, finding $\mathbf{Z}_A^*$ involves only matrix addition that has been highly optimized in Pytorch with the worst-case time complexity of $O(n^2)$. Thus, we get the time complexity of the projection operator as $O(n^2 \cdot n\log(1/\xi)) = O(n^3\log(1/\xi))$.

B.4. Valency ($\mathbf{A1} \leq \mathbf{Xv}$)

Here, we fix $\mathbf{X}$ and let $\mathbf{Xv} = \mathbf{u}$ denote the weighted valency of each node in the graph. Then, the constraint becomes similar to the Max Degree constraint and we follow the same steps to find $\mathbf{Z}_A^* = \mathbf{A} - \frac{1}{2}(\boldsymbol{\mu}\mathbf{1}^T + \mathbf{1}\boldsymbol{\mu}^T) + \text{Diag}(\boldsymbol{\mu})$ except now, we clamp within $[\mathbf{0}, \mathbf{3}]$ since it's a molecular graph and clamp $\mathbf{X}$ within $[\mathbf{0}, \mathbf{1}]$ as well.

B.5. Atom Count ($\mathbf{X}^T\mathbf{1} - \mathbf{c}$)

***Find* $\varphi_{\boldsymbol{\mu}}$.** We have $\mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) = \mathbf{Z}_X^T\mathbf{1} \leq \mathbf{c}$, $\mathbf{Z}_X \in [\mathbf{0}, \mathbf{1}]$. Then, we can note that $\nabla_{\mathbf{Z}_A}\mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) = \mathbf{0}$ and for each column or atom type in $\mathbf{X}$, we get $\mathbf{Z}_X^*[:,j] = \mathbf{X}[:,j] - \mu_j\mathbf{1}^T$ since $\nabla_{\mathbf{Z}_X}\mathbf{h}_{\mathcal{C}} = \mathbf{1}$. Thus, we get $\mathbf{Z}_X^* = P_{[\mathbf{0},\mathbf{1}]}(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T)$.

***Find* $\boldsymbol{\mu}$.** Since $\mathbf{h}$ is a vector-valued function, we cannot obtain its root w.r.t. $\boldsymbol{\mu}$ directly using the bisection method. However, we make another observation that allows us to do that. In particular, $\mathbf{h}_{\mathcal{C}}(\varphi_{\boldsymbol{\mu}}) = \mathbf{0}$ can be divided into F independent equations such that $h_{\mathcal{C}}^j$ satisfies the jth column $(\mathbf{Z}_X^*[:,j] - \mu_j\mathbf{1}^T)\mathbf{1} = c_j$. This can be solved independently for each j using the bisection method between $[\max\{0, \min_i(X_{ij}) - 1\}, \max_i(X_{ij})]$ as $\sum_i P_{[0,1]}(X_{ij} - \max_i(X_{ij})) = 0 \leq c_j$ and $\sum_i P_{[0,1]}(X_{ij} - \min_i(X_{ij}) + 1) = |\mathcal{V}| \geq c_j$.

Complexity. We solve F different equations using bisection method with $\log(\frac{1}{\xi})$ steps each, as $\mathbf{X} \in [\mathbf{0}, \mathbf{1}]$. Further, $\varphi_{\boldsymbol{\mu}}^X$ only involves a matrix addition that is almost constant in Pytorch with worst-case complexity of $O(n^2)$. The total complexity thus, becomes $O(n^2F\log(\frac{1}{\xi}))$.

B.6. Molecular Weight ($\mathbf{1}^T\mathbf{Xm} \leq W$)

***Find* $\varphi_{\boldsymbol{\mu}}$.** We have $\mathbf{h}_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) = h_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) = \mathbf{1}^T\mathbf{Z}_X\mathbf{m} \leq W$, $\mathbf{Z}_X \in [\mathbf{0}, \mathbf{1}]$. Then, $\nabla_{\mathbf{Z}_A}h_{\mathcal{C}} = \mathbf{0}$ and $\nabla_{\mathbf{Z}_X}h_{\mathcal{C}}(\mathbf{Z}_X, \mathbf{Z}_A) = \mathbf{1m}^T$, which gives us $\mathbf{Z}_X^* = \mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T$ followed by clamping within $[\mathbf{0}, \mathbf{1}]$.

***Find* $\boldsymbol{\mu}$.** It is non-trivial to find two end-points between which we can conduct the bisection method for $\mathbf{1}^T P_{[\mathbf{0},\mathbf{1}]}(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T)\mathbf{m} = W$. Thus, we assume that one such point is $\mu = 0$ and search for the first point > 0 with an opposite sign using a linear search from μ with a fixed step size s. Then, we apply the bisection method between 0 and the new point μ_1 found using the linear search.

Complexity. Linear search finds φ^X_μ for μ_1/s different values of μ. This is followed by a bisection method that finishes in $O(\log(\mu_1/\xi))$ steps. Computing φ^X_μ involves just matrix addition that has been highly optimized in Pytorch with worst-case complexity of $O(n^2)$. Thus, the total time-complexity of the projection operator can be given as $O(n^2(\mu_1/s + \log(\mu_1/\xi)))$.

B.7. Dipole Moment ($\|\mathbf{S}^T\mathbf{X}\mathbf{Q}\|_2 \in [\xi_0, \xi_1]$)

A 3D molecular structure can be described as $(\mathbf{X}, \mathbf{S})$, where $\mathbf{S} \in \mathbb{R}^{n\times 3}$ denotes the positions of each atom in the 3-dimensional space from the center-of-mass origin. In addition to this structure, one also requires other quantitative measures such as atomic charges to calculate molecular properties. Let the charges in a molecule be given by a vector $\mathbf{Q} \in \mathbb{R}^F$, then the dipole moment vector can be written as $\boldsymbol{\mu}_{dm} = \mathbf{S}^T\mathbf{X}\mathbf{Q} \in \mathbb{R}^3$. We consider a constraint on its norm as $\|\boldsymbol{\mu}_{dm}\|_2 = \|\mathbf{S}^T\mathbf{X}\mathbf{Q}\|_2 \in [\xi_0, \xi_1]$. We assume no projection with respect to $\mathbf{X}$ and project $\mathbf{S}$ using the projection vector of the ℓ_2-norm (Parikh et al., 2014) simply as $\mathbf{Z}^*_S = \mathbf{S}$ if $\|\mathbf{S}^T\mathbf{X}\mathbf{Q}\|_2 \in [\xi_0, \xi_1]$ otherwise $\mathbf{S}/\|\mathbf{S}^T\mathbf{X}\mathbf{Q}\|_2 \cdot \xi_0$ if $\|\mathbf{S}^T\mathbf{X}\mathbf{Q}\|_2 < \xi_0$ and $\mathbf{S}/\|\mathbf{S}^T\mathbf{X}\mathbf{Q}\|_2 \cdot \xi_1$ if $\|\mathbf{S}^T\mathbf{X}\mathbf{Q}\|_2 > \xi_1$.

However, note that the charges on each atom are unknown and depend upon the given molecular structure. As an approximation, we learn the charges for each atom $\mathbf{Q}$ from the dataset by minimizing the ℓ_1 loss $\sum_{i\in\mathcal{D}} \left|\mu_{dm}(i) - \|\mathbf{S}(i)^T\mathbf{X}(i)\mathbf{Q}\|_2\right|$ over the trainable parameters $\mathbf{Q} \in \mathbb{R}^F$.

C. Extensions

In this section, we discuss several extensions and the corresponding recipes to support more complex constraints and properties including, box constraints and linear and non-linear properties.

C.1. Non-differentiable graph property

While we have focused on the common graph properties that are mostly non-differentiable, we note that our framework can theoretically support non-differentiable graph properties as long as the number of non-differentiabilities is finite. Suppose we want to constrain a non-differentiable property $h_{ND}(\mathbf{G}) \le 0$ of a graph $\mathbf{G}$. In our framework, we consider a constraint space of the form $\mathcal{C} = \{\mathbf{G} : h_1(\mathbf{G}) \le 0, h_2(\mathbf{G}) \le 0, \cdots, h_k(\mathbf{G}) \le 0\}$, where each h_i is assumed to be differentiable to have a closed-form projection operator. Thus, to constrain $h_{ND}(\cdot)$, we can simply divide the constraint on the non-differentiable function h_{ND} at its finite non-differentiable points and consider $h^{(1)}_{ND}(\cdot) \le 0, h^{(2)}_{ND}(\cdot) \le 0, \cdots, h^{(l)}_{ND}(\cdot) \le 0$, where each $h^{(i)}_{ND}$ is differentiable.

C.2. Box Constraint

We note that our formulation allows us to solve a box constraint from the projection operator for the upper bound constraint. In particular, a box constraint can be defined as $\mathcal{C} = \{\mathbf{G} : \delta_{low} \le b(\mathbf{G}) \le \delta_{upp}\}$. This is equivalent to considering $\mathbf{h}_\mathcal{C} : [h^1_\mathcal{C}, h^2_\mathcal{C}]$, such that $h^1_\mathcal{C}(\mathbf{G}) = \delta_{low} - b(\mathbf{G})$ and $h^2_\mathcal{C}(\mathbf{G}) = b(\mathbf{G}) - \delta_{upp}$. Given that $\delta_{low} \le \delta_{upp}$, we can note that both $h^1_\mathcal{C}(\mathbf{G}) > 0$ and $h^2_\mathcal{C}(\mathbf{G}) > 0$ cannot hold. Thus, we get

$$\Pi_\mathcal{C}(G) = \begin{cases} \varphi_0(\mathbf{G}) & ; h^1_\mathcal{C}(\varphi_0(\mathbf{G})) \le 0, h^2_\mathcal{C}(\varphi_0(\mathbf{G})) \le 0 \\ \varphi_\mu(\mathbf{G}) & ; h^1_\mathcal{C}(\varphi_0(\mathbf{G})) \le 0, h^2_\mathcal{C}(\varphi_0(\mathbf{G})) > 0, h^2_\mathcal{C}(\varphi_\mu(\mathbf{G})) = 0 \\ \varphi_{-\mu}(\mathbf{G}) & ; h^2_\mathcal{C}(\varphi_0(\mathbf{G})) \le 0, h^1_\mathcal{C}(\varphi_0(\mathbf{G})) > 0, h^1_\mathcal{C}(\varphi_{-\mu}(\mathbf{G})) = 0 \end{cases} \tag{8}$$

C.3. Linear Property Approximators ($\mathbf{1}^T\hat{\mathbf{A}}^k\mathbf{X}\Theta \le p$)

Find $\boldsymbol{\varphi_\mu}$***.*** We have $\mathbf{h}_\mathcal{C}(\mathbf{Z}_X, \mathbf{Z}_A) = h_\mathcal{C}(\mathbf{Z}_X, \mathbf{Z}_A) = \mathbf{1}^T\hat{\mathbf{Z}_A}^k\mathbf{Z}_X\mathbf{\Theta} - p$, $\mathbf{Z}_X \in [\mathbf{0}, \mathbf{1}]$. We fix $\mathbf{A}$ and thus, assume $\mathbf{Z}_A = P_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})$. Let $\hat{P}_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})$ denote the normalized adjacency matrix corresponding to $P_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})$. Then, $\nabla_{\mathbf{Z}_A} h_\mathcal{C} = \mathbf{0}$ and $\nabla_{\mathbf{Z}_X} h_\mathcal{C}(\mathbf{Z}_X, \mathbf{Z}_A) = (\hat{P}_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})^k)^T\mathbf{1}\mathbf{\Theta}^T$, which gives us $\mathbf{Z}^*_X = \mathbf{X} - \mu(\hat{P}_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})^k)^T\mathbf{1}\mathbf{\Theta}^T$ followed by clamping within $[\mathbf{0}, \mathbf{1}]$.

Find $\boldsymbol{\mu}$***.*** It is non-trivial to find two end-points between which we can conduct the bisection method for which there is equality on $h_\mathcal{C}$. Thus, we assume that one such point is $\mu = 0$ and search for the first point > 0 with an opposite sign using a linear search from μ with a fixed step size s. Then, we apply the bisection method between 0 and the new point μ_1 found using the linear search.

Complexity. Linear search finds φ_μ^X for μ_1/s different values of μ. This is followed by a bisection method that finishes in $O(\log(\mu_1/\xi))$ steps. Computing φ_μ^X involves just matrix multiplication that has been highly optimized in Pytorch with worst-case complexity of $O(n^3)$. Thus, the total time-complexity of the projection operator can be given as $O(n^3(\mu_1/s + \log(\mu_1/\xi)))$.

C.4. Non-Linear Property Approximators

Many graph properties are estimated using neural networks with non-linear activation functions. Constraining these properties implies constraining the output of these networks. Let us consider a typical single-layer neural network prediction, whose output can be written as $\mathbf{w}_2^T\text{ReLU}(\mathbf{W}_1^T\mathbf{u} + \mathbf{b}_1) + b_2$. The corresponding constraint would then look like $h_{\mathcal{C}}(\mathbf{u}) = \mathbf{w}_2^T\text{ReLU}(\mathbf{W}_1^T\mathbf{u} + \mathbf{b}_1) + b_2 - \epsilon \leq 0$ for some trained parameters $\mathbf{W}_1, \mathbf{w}_2, \mathbf{b}_1, b_2$. We want to find z_u such that $\|\mathbf{z}_u - \mathbf{u}\|_2^2$ is minimized such that this constraint is satisfied. The Lagrangian is given as $\mathcal{L}(\mathbf{z}_u, \lambda) = \frac{1}{2}\|\mathbf{z}_u - \mathbf{u}\|_2^2 + \lambda(\mathbf{w}_2^T\text{ReLU}(\mathbf{W}_1^T\mathbf{u} + \mathbf{b}_1) + b_2 - \epsilon)$ and applying the KKT conditions give us $(\mathbf{z}_u^* - \mathbf{u}) + \lambda\mathbf{w}_2^T\mathbb{1}\{\mathbf{W}_1^T\mathbf{z}_u^* + \mathbf{b}_1 \geq 0\} \odot \mathbf{W}_1^T\mathbf{1} = \mathbf{0}, \lambda \geq 0, \mathbf{w}_2^T\text{ReLU}(\mathbf{W}_1^T\mathbf{z}_u^* + \mathbf{b}_1) + b_2 - \epsilon \leq 0, \lambda(\mathbf{w}_2^T\text{ReLU}(\mathbf{W}_1^T\mathbf{z}_u^* + \mathbf{b}_1) + b_2 - \epsilon) = 0$. If $\mathbf{w}_2^T\text{ReLU}(\mathbf{W}_1^T\mathbf{u} + \mathbf{b}_1) + b_2 - \epsilon \leq 0$, then $\mathbf{z}^* = \mathbf{u}$ (since $\lambda = 0$ otherwise we find a $\lambda \geq 0$ and $\mathbf{z}_u^*$ such that $\mathbf{w}_2^T\text{ReLU}(\mathbf{W}_1^T\mathbf{z}_u^* + \mathbf{b}_1) + b_2 - \epsilon = 0$ and $(\mathbf{z}_u^* - \mathbf{u}) + \lambda\mathbf{w}_2^T\mathbb{1}\{\mathbf{W}_1^T\mathbf{z}_u^* + \mathbf{b}_1 \geq 0\} \odot \mathbf{W}_1^T\mathbf{1} = \mathbf{0}$. Solving such a system of equations is hard since the first equation gives us $\mathbf{w}_2^T\text{ReLU}(\mathbf{W}_1^T\mathbf{z}_u^* + \mathbf{b}_1) = \epsilon - b_2$, which can have infinitely many solutions for $\text{ReLU}(\mathbf{W}_1^T\mathbf{z}_u^* + \mathbf{b}_1)$ and consequently $\mathbb{1}\{\mathbf{W}_1^T\mathbf{z}_u^* + \mathbf{b}_1 \geq 0\}$ and $\mathbf{z}_u^*$, which can not be directly substituted in the second equation. Therefore, we do not consider non-linear approximators of graph properties and leave it for future works to find efficient projections for these general functions.

D. Additional Experiment Details

D.1. Datasets

We consider the following 4 generic graph datasets:

1. **Ego-small** contains 200 small ego graphs from larger Citeseer network (Sen et al., 2008).
2. **Community-small** consists of 100 randomly generated community graphs.
3. **Enzymes** has 587 protein graphs of the enzymes from the BRENDA database (Schomburg et al., 2004).
4. **SBM** is a synthetic dataset of stochastic block model graphs with 20-40 nodes per community and 2-5 communities.
5. **Planar** is a synthetic dataset of planar graphs of 64 nodes.

We also consider these 2 molecular graph datasets:

1. **QM9** consists of 133k small molecules with 1-9 atoms as Carbon (C), Nitrogen (N), Oxygen (O), and Fluorine (F).
2. **ZINC250k** consists of 250k molecules with 6-38 atoms as Carbon (C), Nitrogen (N), Oxygen (O), Fluorine (F), Phosphorus (P), Chlorine (Cl), Bromine (Br), and Iodine (I).

E. Additional Results

E.1. Raw MMD values and hyperparameters

Table 10 shows the raw average MMD and VUN (wherever applicable) numbers along with the best γ_t that gives the best performance in different constraints and models on synthetic datasets. Table 11 shows the raw scores obtained for 2D molecular graphs with different diffusion models under different constraints. One can note that different γ_t might be suitable in different scenarios. In these tables, we also include the results for the generated graphs by base diffusion models using the original sampling. Note that the original model's samples are agnostic of the given constraint and any constraint validity is merely an indication of the bias learned by the model towards the constrained set. The expected constraint validity of these constraints for a perfectly matching distribution should be only around 0.1.

Table 10. Raw average MMD scores and hyperparameters for the results on non-attributed synthetic graph datasets. We also include the results for the graphs generated by base diffusion models to note any bias towards the constraint in the learned distribution. "+PRODIGY" indicates the diffusion model sampled using PRODIGY sampling. Note that $\text{poly}(a, b) = (1-b)t^a + b$.

		SBM				**Planar**			
		γ_t	Avg. MMD↓	V.U.N.↑	$\text{VAL}_{\mathcal{C}}$ ↑	γ_t	Avg. MMD↓	V.U.N.↑	$\text{VAL}_{\mathcal{C}}$ ↑
Edge Count	DruM		0.04	0.78	0.50		0.0	0.95	0.12
	+PRODIGY	poly(0,5)	0.32	0.38	1.00	$\exp(-1000d_{\mathcal{C}})$	0.09	0.80	1.00
	DiGress		0.04	0.65	0.48		0.01	0.68	0.55
	+PRODIGY	poly(0,5)	0.03	0.60	0.65	$\exp(-10d_{\mathcal{C}})$	0.01	0.75	1.00
Triangle Count	DruM		0.01	0.78	0.78		0.00	0.95	1.00
	+PRODIGY	poly(0,5)	0.31	0.57	1.00	$\exp(-d_{\mathcal{C}})$	0.00	0.98	1.00
	DiGress		0.04	0.65	0.88		0.01	0.68	1.00
	+PRODIGY	poly(0,1)	0.03	0.62	0.85	$\exp(-d_{\mathcal{C}})$	0.02	0.75	1.00
Degree	DruM		0.21	0.78	0.15		0.00	0.95	0.22
	+PRODIGY	poly(0,5)	0.31	0.42	0.90	$\exp(-d_{\mathcal{C}})$	0.02	0.92	0.70
	DiGress		0.04	0.65	0.45		0.01	0.68	0.55
	+PRODIGY	poly(0,5)	0.03	0.65	0.57	$\exp(-10d_{\mathcal{C}})$	0.01	0.70	0.75

E.2. How does PRODIGY compare to state-of-art guidance-based generation approaches?

To answer this question, we compare our results with that of DiGress (Vignac et al., 2022), which considers a soft constraint on the molecular properties that our approach also supports (but with the ability to apply hard interpretable constraints).

Molecular Property. Here, we use PRODIGY to generate molecules with a specified molecular property. We follow the framework of DiGress (Vignac et al., 2022) and constrain the dipole moment (μ) and the highest occupied molecular orbit (HOMO) of the generated molecules to be close to a certain set of values. These properties cannot be written easily in terms of the molecular graph $(\mathbf{X}, \mathbf{A})$, as required by our framework.

Table 12. MAE of dipole moment constrained generation.

	μ	HOMO
DiGress (Unconditional)	$1.71 \pm .04$	$0.93 \pm .01$
DiGress+Guidance	$0.81 \pm .04$	$0.56 \pm .01$
GDSS (Unconditional)	$2.09 \pm .01$	$0.30 \pm .02$
GDSS+PRODIGY	$1.09 \pm .02$	$0.29 \pm .10$

Hence, we train a simple graph convolutional network (Wu et al., 2019), as described in Section 3.1, to act as a proxy for the molecular property. We then constrain the predicted property to lie within a range of the given value. Following the conditional generation of Digress, we consider minimizing the mean absolute error (MAE) between the generated molecules and the first hundred molecules of QM9. We thus use the median of these values to constrain the predicted property. Table 12 shows the performance of our sampling technique (on top of a pre-trained GDSS) model against the DiGress baseline. We can see that even after using a simple linear model for property estimation, we can generate molecules with competitive μ and HOMO as DiGress that employs a Graph Transformer as the proxy.

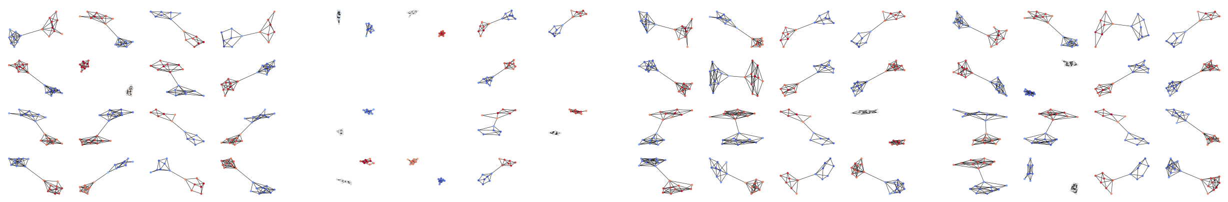
E.3. Visualizations

DruM. Here, we provide additional visualizations of PRODIGY using the DruM diffusion model. Figures 7, 8, 9, 10, and 11 provide the visualization of different constraints in Community-small, Ego-small, Enzymes, SBM, and Planar. We can observe that the generated graphs can satisfy the given constraints while being close to the original distribution.

GeoLDM (3D molecules). Figure 12 shows some sample molecules that were generated with predicted dipole moment within the specified range. This shows that we can generate molecules with a large variety of atom types as we see Oxygen and Nitrogen across different ranges.

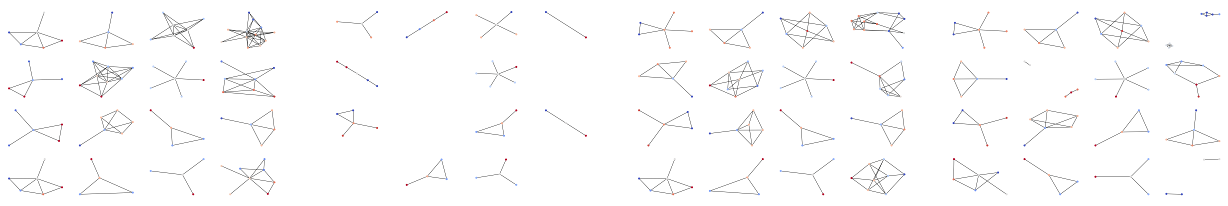
Table 11. Raw scores and hyperparameters for the results on molecular graph datasets. We also include the results for the graphs generated by base diffusion models to note any bias towards the constraint in the learned distribution. "+PRODIGY" indicates the diffusion model sampled using PRODIGY sampling. Note that poly$(a, b) = (1 - b)t^a + b$.

		QM9					**ZINC250k**				
		γ_t	Val. (%) ↑	Novel. (%) ↑	FCD ↓	VAL$_\mathcal{C}$ ↑	γ_t	Val. (%) ↑	Novel. (%) ↑	FCD ↓	VAL$_\mathcal{C}$ ↑
Valency	EDP-GNN		96.95	76.74	6.15	0.97	OOTM	OOTM	OOTM	OOTM	OOTM
	+PRODIGY	0.01	96.29	76.68	6.23	0.96	OOTM	OOTM	OOTM	OOTM	OOTM
	GDSS		95.72	81.04	2.47	0.88		97.01	100.00	14.04	0.94
	+PRODIGY	poly(0,5)	99.83	82.74	2.82	0.99	poly(0,5)	99.88	100.00	29.79	0.99
	DruM		100.00	25.17	0.11	0.99		100.00	99.97	3.34	0.84
	+PRODIGY	$\exp(-100d_\mathcal{C})$	100.00	23.80	2.76	0.99	$\exp(-100d_\mathcal{C})$	100.00	100.00	8.96	0.98
	DiGress		99.76	35.82	6.23	1.00	OOTM	OOTM	OOTM	OOTM	OOTM
	+PRODIGY	poly(0,1)	99.76	26.94	5.90	1.00	OOTM	OOTM	OOTM	OOTM	OOTM
Atom Count	EDP-GNN		96.95	76.74	8.63	0.37	OOTM	OOTM	OOTM	OOTM	OOTM
	+PRODIGY	0.01	98.06	54.36	5.66	1.00	OOTM	OOTM	OOTM	OOTM	OOTM
	GDSS		95.72	81.04	7.28	0.33		97.01	100.00	16.01	0.13
	+PRODIGY	poly(0,5)	95.02	67.67	1.60	1.00	poly(0,5)	96.90	100.00	12.24	0.99
	DruM		100.00	25.17	7.54	0.37		100.00	99.97	27.52	0.01
	+PRODIGY	$\exp(-100d_\mathcal{C})$	100.00	15.90	0.22	0.99	poly(0,5)	100.00	99.99	4.61	1.00
	DiGress		99.76	35.82	5.54	0.54	OOTM	OOTM	OOTM	OOTM	OOTM
	+PRODIGY	poly(0.1,1)	99.76	30.05	4.60	0.65	OOTM	OOTM	OOTM	OOTM	OOTM
Molecular Weight	EDP-GNN		96.95	76.74	16.86	0.00	OOTM	OOTM	OOTM	OOTM	OOTM
	+PRODIGY	1.00	28.91	70.05	3.80	0.17	OOTM	OOTM	OOTM	OOTM	OOTM
	GDSS		95.72	81.04	17.08	0.00		97.01	100.00	11.15	0.62
	+PRODIGY	poly(0.1,1)	99.92	81.14	4.16	0.53	poly(0,5)	97.66	100.00	12.15	0.63
	DruM		100.00	25.17	1.04	0.23		100.00	99.97	7.44	0.18
	+PRODIGY	poly(0,1)	100.00	23.60	4.76	0.65	poly(0,1)	100.00	99.96	7.25	0.26
	DiGress		99.76	35.82	6.14	0.34	OOTM	OOTM	OOTM	OOTM	OOTM
	+PRODIGY	poly(0.1,1)	99.76	34.90	6.20	0.33	OOTM	OOTM	OOTM	OOTM	OOTM



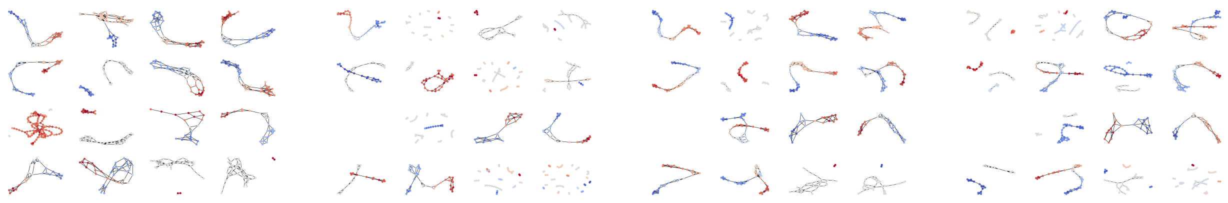
(a) No constraint (b) Edge-Count ≤ 21 (c) Triangle-Count ≤ 30 (d) Max-Degree ≤ 6

Figure 7. Some generated graphs from DruM using PRODIGY for different constraints on Community-small.



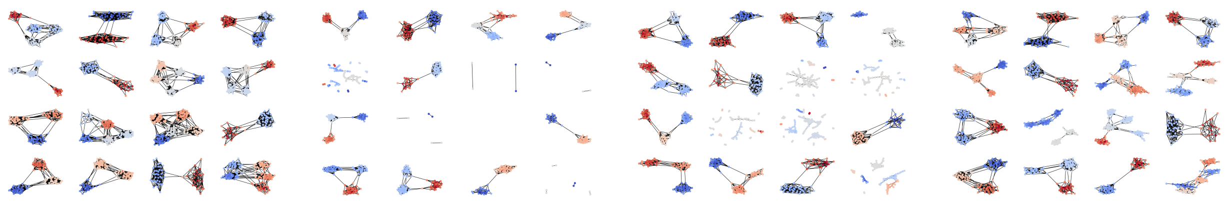
(a) No constraint (b) Edge-Count ≤ 3 (c) Triangle-Count ≤ 3 (d) Max-Degree ≤ 3

Figure 8. Some generated graphs from DruM using PRODIGY for different constraints on Ego-small.



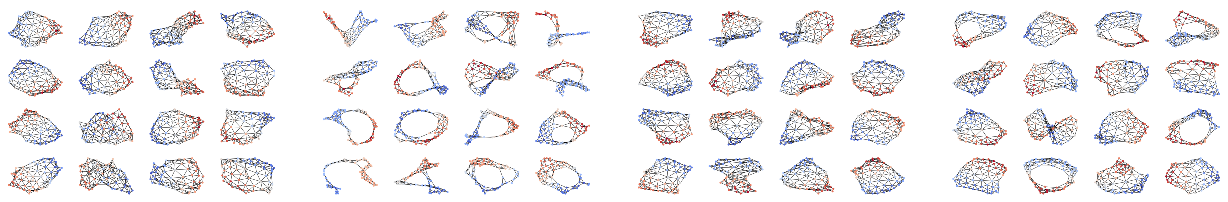
(a) No constraint (b) Edge-Count ≤ 49.7 (c) Triangle-Count ≤ 45 (d) Max-Degree ≤ 5

Figure 9. Some generated graphs from DruM using PRODIGY for different constraints on Enzymes.



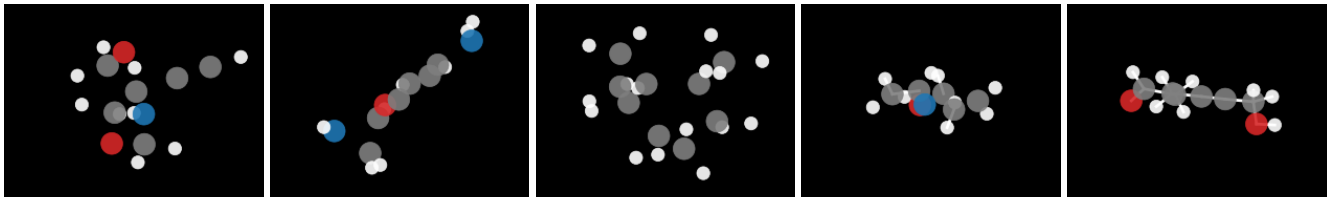
(a) No constraint (b) Edge-Count ≤ 700.12 (c) Triangle-Count ≤ 682.2 (d) Max-Degree ≤ 14

Figure 10. Some generated graphs from DruM using PRODIGY for different constraints on SBM.

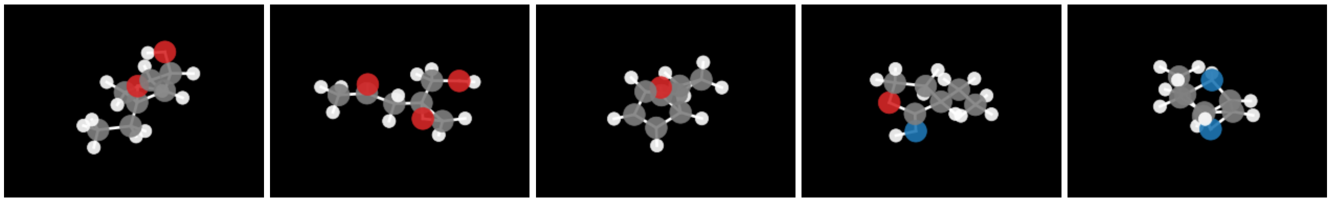


(a) No constraint (b) Edge-Count ≤ 177 (c) Triangle-Count ≤ 342 (d) Max-Degree ≤ 8

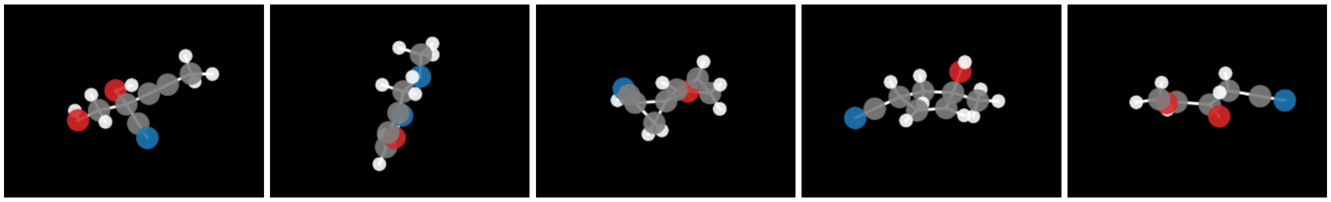
Figure 11. Some generated graphs from DruM using PRODIGY for different constraints on Planar.



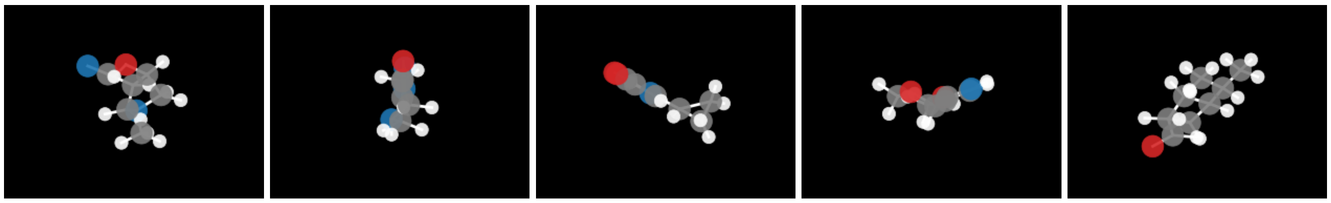
(a) $\mu_{dm} \leq 0.96$



(b) $0.96 \leq \mu_{dm} \leq 1.68$



(c) $1.92 \leq \mu_{dm} \leq 2.40$



(d) $\mu_{dm} \geq 2.40$

Figure 12. 3D GeoLDM+PRODIGY generations for Dipole Moment constraint for a range of values.