Plug-and-Play Controllable Graph Generation with Diffusion Models

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

Abstract

Diffusion models for graph generation present transformative capabilities in generating high-quality graphs. However, controlling the properties of the generated graphs remains a challenging task for the existing methods as they mainly focus on uncontrolled graph generation from the data. To address this limitation, we propose PRODIGY (PROjected DIffusion for generating constrained Graphs), a novel approach for controllable graph generation that works with any pre-trained diffusion model. This formalizes the problem of controlled graph generation and identifies a class of constraints (e.g., edge count, valency, etc.) applicable to practical graph generation tasks. At the center of our approach is a plug-and-play sampling process, based on projection-based optimization to ensure that each generated graph satisfies the specified constraints. Experiments demonstrate the effectiveness of PRODIGY in generating high-quality and diverse graphs that satisfy the specified constraints while staying close to the training distribution.

1. Introduction

Deep generative models have been used in the literature to learn the underlying distribution of graph-structured data (You et al., 2018; Jo et al., 2022; Niu et al., 2020; Vignac et al., 2022; Liu et al., 2019; Shi et al., 2020). Recently, diffusion-based models (Niu et al., 2020; Vignac et al., 2022; Jo et al., 2022) 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 is to enable large-scale simulation of realistic networks for domains such as network optimization (Xie et al., 2019), social network analysis (You et al., 2018; Grover et al., 2019), and drug design (Yang et al., 2022).

However, even with their amazing performance on benchmark datasets, a major limitation of diffusion-based approaches stems from their inability to support meaningful controllable generation. Existing methods sporadically support controllable generation (often termed conditional generation), by using a label to influence reverse diffusion with a soft constraint (Song et al., 2020; Vignac et al., 2022). This approach makes the conditional generation process obscure and hampers interpretation and control over the generated graph samples. In real-world applications like drug discovery, precise control over the generated outputs for specific features (e.g., the number of atoms or motifs in a molecule) is crucial for a generative algorithm.

In this work, we fill this gap by investigating the problem of controllable graph generation to generate graphs that belong to a prior distribution while satisfying certain user-defined hard constraints on the structure and/or attributes. This enables interpretable control over the output of the generation model. Specifically, we propose PRODIGY (PROjected DIffusion for generating constrained Graphs), a plug-and-play controllable generation method for graphs that can be used to sample constrained graphs. In particular, we design a Projected diffusion-based sampling process that can be plugged over any pre-trained diffusion model. For a given constraint, we project the solution to the constrained space after each step of the reverse process. This guarantees that the solution obtained after following the reverse diffusion process would satisfy the constraint.

2. Problem Setup: Plug-and-Play Controllable Graph Generation

Existing works study the problem of (unconstrained) graph generation, where the objective is to generate new graphs $\{\mathbf{G}\}$ given a set of training graphs $\mathcal{G}_{tr} \subset \mathcal{G}$ such that the new graphs are sampled from the same underlying distribution p_0 as $\mathcal{G}_{tr}$. This is achieved using generative models that estimate a probability distribution $\widehat{p}$ over the full set of graphs $\mathcal{G}$ such that it is close to the original distribution, i.e., $\widehat{p} \approx p_0$ (You et al., 2018; Niu et al., 2020; Jo et al., 2022).

We consider the problem of controllable graph generation,

[*]Equal contribution [1]Georgia Institute of Technology, Atlanta, USA [2]Massachusetts Institute of Technology, Boston, USA. Correspondence to: Kartik Sharma <ksartik@gatech.edu>.

Accepted to ICML workshop on Structured Probabilistic Inference Generative Modeling, Honolulu, Hawaii, USA.

where the objective is to control the generation within a given constrained set. In addition, the constraints are provided during the sampling stage such that the generative model cannot be retrained – this requirement will allow reusing existing models as-is.

Problem 1. ***(Plug-and-Play Controllable Graph Generation)*** *Given a constrained set $\mathcal{C} \subseteq \mathcal{G}$ and a pre-trained graph generation model $\mathcal{M}$ trained on some training set $\mathcal{G}_{tr} \sim p_0$, generate new graphs $\{\mathbf{G}\}$ such that $\mathbf{G} \sim p_0$ and $\mathbf{G} \in \mathcal{C}$.*

We assume a black-box access to the generative model $\mathcal{M}$ such that only the probability distribution $\widehat{p}$ can be accessed during sampling. In other words, there is no access to the training set $\mathcal{G}_{tr}$ or the model parameters $\mathbf{\Theta}(\mathcal{M})$. This is a practical assumption as these models and datasets are often not released due to proprietary reasons (Ramesh et al., 2021; OpenAI, 2023). Thus, the proposed constrained generation solution must be flexible to the choice of the generative model and the constraints. In this work, we consider $\mathcal{M}$ to be any diffusion-based generative model as these are shown to outperform other methods for graph generation (Niu et al., 2020; Jo et al., 2022).

2.1. Constraints

Graph generation should be controlled for different constraints on the structure and derived properties. It is often desired that the structure of the generated graph can be controlled by the user. For example, a user may want to control the number of edges, triangles, or the maximum degree in the graph structure (Tabourier et al., 2011; Ying & Wu, 2009). In particular, we consider the following three constraints on the graph structure of non-attributed graphs:

1. **Edge Count**: The number of edges in the generated graph is at most a constant $\mathcal{B}$, i.e., $|\mathbf{E}| = \frac{1}{2}\mathbf{1}^T\mathbf{A}\mathbf{1} \leq \mathcal{B}$ (undirected graph).

2. **Triangle Count**: The number of triangles in the generated graph is at most a constant T, i.e., $\frac{1}{6}\text{tr}(\mathbf{A}) \leq T$.

3. **Degree**: The maximum degree in the generated graph is at most a constant δ_d. Thus, we restrict the degree of each node to be δ_d, i.e., $\mathbf{A}\mathbf{1} \leq \delta_d\mathbf{1}$.

For molecular graphs, we have $\mathbf{X} \in \mathbb{R}^{n\times F}$ denoting 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 and has the given properties (molecular weight, dipole moment, HOMO, etc.) (Liu et al., 2018; Vignac et al., 2022; Jo et al., 2022; Langevin et al., 2020; Shi et al., 2020). In addition, restricting the number of atoms of each type in the generated molecule is important for the molecule to be valid. Here, we consider the following four constraints for molecular graphs:

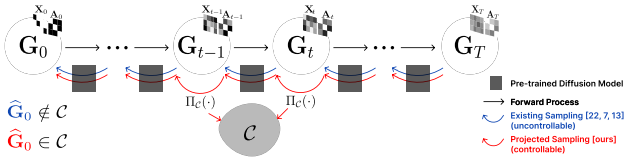


Figure 1: Constrained graph generation.

1. **Valency**: Assuming hidden Hydrogen atoms will be added at the end (Jo et al., 2022; Vignac et al., 2022), we want the number of connections to a certain atom to be at most its valency, i.e., $\mathbf{A}\mathbf{1} \leq \mathbf{X}\mathbf{v}$, where $\mathbf{v} \in \mathbb{Z}^F_{>0}$ denotes the valency of each atom type.

2. **Atom Count**: The number of atoms of each type i in the generated molecule is at most c_i, i.e., $\mathbf{X}^T\mathbf{1} - \mathbf{c}$, for a fixed vector $\mathbf{c} \in \mathbb{Z}^F_{\geq 0}$.

3. **Molecular Weight**: Molecular weight of the generated molecule is at most a constant W, i.e., $\mathbf{1}^T\mathbf{X}\mathbf{m} \leq W$, where $\mathbf{m} \in \mathbb{R}^F_{>0}$ denotes the atomic weights of each atom type.

4. **Molecular Property**: A general molecular property cannot be formulated in simple terms of the graph structure. Thus, we predict a property of a molecular graph by training a linear SGCN model (Wu et al., 2019) on a regression task. Then, we consider a constraint that the predicted property is at most a given constant p, i.e., $\mathbf{1}^T\hat{\mathbf{A}}^k\mathbf{X}\mathbf{\Theta} \leq p$, where $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ is the normalized adjacency matrix such that $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is a diagonal matrix storing the degrees of nodes in $\tilde{\mathbf{A}}$. $\mathbf{\Theta}$ denotes the trained weights of the SGCN model.

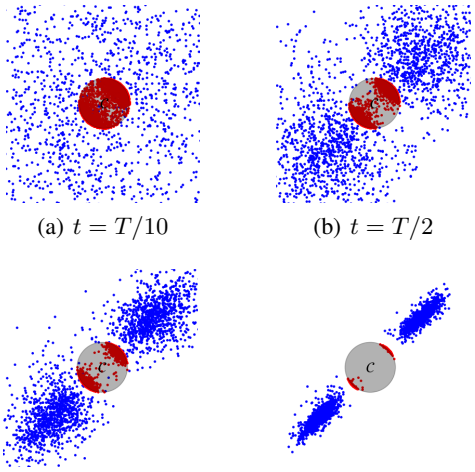
3. Proposed Method: Projected Diffusion for Constrained Graphs

We propose PROjected DIffusion for constrained Graphs (PRODIGY), a plug-and-play sampling method for controllable graph generation for continuous-time diffusion models. Figure 1 illustrates our idea and how it will enable controlled graph generation.

Based on works in Mirrored Langevin Dynamics (Bubeck et al., 2018; Hsieh et al., 2018), we extend the idea of Projected Gradient Descent of alternate dynamics and projection to sample from a constrained set. In particular, we consider the following sampling process

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

where Reverse is some arbitrary discretization of the reverse process of the continuous-time diffusion model. Figure 2 illustrates the sampling process of our method as compared to



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

Figure 2: Sampling process of PRODIGY (red) versus existing methods (blue) at different timesteps (t) for a constrained generation within an ℓ_2 ball (gray) centered at the origin and a radius of 0.1. PRODIGY generates points at the periphery of the constrained set $\mathcal{C}$, closest to the data density. The original distribution is a Gaussian mixture model of two equally likely normal distributions with means $(1, 1)$ and $(-1, -1)$ and symmetric covariances of 0.01 and 0.009.

the existing sampling strategies. PRODIGY is able to sample within the constrained ℓ_2 ball while existing strategies fail to do that.

However, we also note that projection to an arbitrary constrained set can destroy the smoothness of the reverse process. This is because $\mathbf{G}_t$ and $\mathbf{G}_{t-1}$ can be very far apart since the original sampling is designed to reverse the original unconditional distribution and may be very different from the constrained set. To account for this, we also propose a smoother variant of Equation 1 as $\mathbf{G}_{t-1} \leftarrow (1-\gamma_t)\,\widetilde{\mathbf{G}}_{t-1} + \gamma_t\,\Pi_{\mathcal{C}}(\widetilde{\mathbf{G}}_{t-1})$.

One can note that a higher γ_t implies a higher chance of constraint satisfaction but also a higher distortion to the original sampling process. Thus, we should select a lower γ_t when the focus is to approximate the underlying distribution p_0 and a higher γ_t when we want to just satisfy the constraint. We handle this tradeoff by considering a polynomial schedule on this parameter w.r.t. the diffusion timestep as $\gamma_t = \text{poly}(t) = (1-\gamma_0)\left(\frac{T-t}{T}\right)^r + \gamma_0$, for some $\gamma_0, r \geq 0$.

3.1. Projection operators

Consider a constraint of the form $\mathcal{C} = \{\mathbf{S} = (\mathbf{S}_X, \mathbf{S}_A) \in \mathcal{G} : \mathbf{h}_{\mathcal{C}}(\mathbf{S}) \leq \mathbf{0}\}$, i.e., that forms a sublevel set. In addition to this, we also require the graphs to be undirected, have zero self-loops, and be within a certain range of values.

Table 1: Projection Operators for the given constraints. We have $\Pi_{\mathcal{C}}(\mathbf{G}) = \varphi_{\mathbf{0}}(\mathbf{G})$ if $\mathbf{h}_{\mathcal{C}}(\varphi_{\mathbf{0}}(\mathbf{G})) \leq \mathbf{0}$, otherwise $\varphi_{\boldsymbol{\mu}}(\mathbf{G})$ such that $\mathbf{h}_{\mathcal{C}}(\varphi_{\boldsymbol{\mu}}(\mathbf{G})) = \mathbf{0}$.

Constraint	Constraint Function ($\mathbf{h}_{\mathcal{C}}$)	Projection ($\varphi_{\boldsymbol{\mu}} = (\varphi^X_{\boldsymbol{\mu}}, \varphi^A_{\boldsymbol{\mu}})$)	
		$\varphi^X_{\boldsymbol{\mu}}$	$\varphi^A_{\boldsymbol{\mu}}$
Edge Count	$\frac{1}{2}\mathbf{1}^T\mathbf{A}\mathbf{1} - \mathcal{B}$	$\mathbf{X}$	$P_{[0,1]}(\mathbf{A} - \mu\mathbf{1}\mathbf{1}^T/2 + \mathbf{I}/2)$
Triangle Count	$\frac{1}{6}\text{tr}(\mathbf{A}^3) - T$	$\mathbf{X}$	$P_{[0,1]}(\mathbf{A} - \mu\mathbf{A}^2/2)$
Degree	$\mathbf{A}\mathbf{1} - \delta_d\mathbf{1}$	$\mathbf{X}$	$P_{[0,1]}(\mathbf{A} - \frac{1}{2}(\boldsymbol{\mu}\mathbf{1}^T + \mathbf{1}\boldsymbol{\mu}^T) + \text{Diag}(\boldsymbol{\mu}))$
Valency	$\mathbf{A}\mathbf{1} - \mathbf{X}\mathbf{v}$	$P_{[0,1]}(\mathbf{X})$	$P_{[0,3]}(\mathbf{A} - \frac{1}{2}(\boldsymbol{\mu}\mathbf{1}^T + \mathbf{1}\boldsymbol{\mu}^T) + \text{Diag}(\boldsymbol{\mu}))$
Atom Count	$\mathbf{X}^T\mathbf{1} - \mathbf{c}$	$P_{[0,1]}(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T)$	$P_{[0,3]}(\mathbf{A})$
Molecular Weight	$\mathbf{1}^T\mathbf{X}\mathbf{m} - W$	$P_{[0,1]}(\mathbf{X} - \mu\mathbf{1}\mathbf{m}^T)$	$P_{[0,3]}(\mathbf{A})$
Molecular Property	$\mathbf{1}^T\hat{\mathbf{A}}^k\mathbf{X}\boldsymbol{\Theta} - p$	$P_{[0,1]}(\mathbf{X} - \mu(\hat{P}_{[0,3]}(\mathbf{A})^k)^T\mathbf{1}\boldsymbol{\Theta}^T)$	$P_{[0,3]}(\mathbf{A})$

Then, the projection operator is given as:

$$\Pi_{\mathcal{C}}(\mathbf{G}) = \underset{\substack{(\mathbf{S}_X,\mathbf{S}_A)\in\mathcal{G}\\ \mathbf{h}_{\mathcal{C}}(\mathbf{S}_X,\mathbf{S}_A)\leq\mathbf{0}}}{\arg\min} \tfrac{1}{2}\|\mathbf{S}_X - \mathbf{X}\|_2^2 + \tfrac{1}{2}\|\mathbf{S}_A - \mathbf{A}\|_2^2, \quad (2)$$

such that $\mathbf{S}_X \in [\mathbf{X}_m, \mathbf{X}_M], \mathbf{S}_A \in [\mathbf{A}_m, \mathbf{A}_M], \mathbf{S}_A^T = \mathbf{S}_A, \text{Diag}(\mathbf{S}_A) = \mathbf{0}$. This can be solved using the Lagrangian method, which gives us $\mathcal{L}(\mathbf{S}_X, \mathbf{S}_A, \mathbf{h}_{\mathcal{C}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{S}_X - \mathbf{X}\|_2^2 + \frac{1}{2}\|\mathbf{S}_A - \mathbf{A}\|_2^2 + \boldsymbol{\mu}_0 \cdot \mathbf{h}_{\mathcal{C}}(\mathbf{S}_X, \mathbf{S}_A) + \boldsymbol{\mu}_1 \cdot (\mathbf{S}_X - \mathbf{X}_m) + \boldsymbol{\mu}_2 \cdot (\mathbf{X}_M - \mathbf{S}_X) + \boldsymbol{\mu}_3 \cdot (\mathbf{S}_A - \mathbf{A}_m) + \boldsymbol{\mu}_4 \cdot (\mathbf{A}_M - \mathbf{S}_A) + \sum_{i>j} \lambda_{ij}(\mathbf{S}_A[i,j] - \mathbf{S}_A[j,i]) + \sum_i \lambda_i \mathbf{S}_A[i]$. Karush–Kuhn–Tucker (KKT) conditions (Kuhn & Tucker, 2013) imply that the optimal solution $\mathbf{S}^*$ should follow

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

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

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

Table 1 lists the projection operators for different constraint functions. We defer the proofs to Appendix B. For each constraint ($\mathbf{h}_{\mathcal{C}}$), we have $\Pi_{\mathcal{C}}(\mathbf{G}) = \varphi_{\mathbf{0}}(\mathbf{G})$ if $\mathbf{h}_{\mathcal{C}}(\varphi_{\mathbf{0}}(\mathbf{G})) \leq \mathbf{0}$, otherwise $\varphi_{\boldsymbol{\mu}}(\mathbf{G})$ such that $\mathbf{h}_{\mathcal{C}}(\varphi_{\boldsymbol{\mu}}(\mathbf{G})) = \mathbf{0}$. For most constraints, we can note that $\mathbf{h}_{\mathcal{C}}$ 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 $h^{(i)}_{\mathcal{C}}$ and solve for μ_i such that $h^{(i)}_{\mathcal{C}}(\varphi_{\mu_i}(\mathbf{G})) = 0$ using the bisection method. The split is done such that if $h^{(i)}_{\mathcal{C}}(\varphi_{\mu_i}(\mathbf{G})) = 0$ 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.

4. Experimental Setup

Datasets. We consider four generic datasets that include Community-small, Ego-small, Grid (You et al., 2018), and

Enzymes (Jo et al., 2022). In addition, we also consider two molecular datasets, QM9 (Ramakrishnan et al., 2014) and ZINC250k (Irwin et al., 2012). For a fair comparison, we follow the standard experimental setup of existing works (You et al., 2018; Jo et al., 2022). Please see Appendix C for more details.

Base models. (1) EDP-GNN (Niu et al., 2020) follows Langevin dynamics to sample only adjacency matrices, (2) GDSS (Jo et al., 2022) samples both node attributes and adjacency matrices at the same time from a diffusion process.

Metrics. We measure the performance of our method for both the generation quality and their effectiveness in satisfying the given constraint. For the latter, we simply find 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 $\{\mathbf{G}_i\}$. The quality of unconditionally generated graphs is evaluated by comparing the distributions of certain graph statistics between generated and test graphs by using the maximum mean discrepancy (MMD) metric (Jo et al., 2022; You et al., 2018). However, since our objective is to specifically generate constrained graphs, comparing against the whole test set is not correct as the difference in the statistics could be due to the constraint itself. To account for this, we filter the test set to only consider those graphs that satisfy the constraint and find the MMD between the generated graphs and the *constrained* test set.

5. Results

We show the empirical performance of PRODIGY to control graph generation under different constraints. Below, we provide results for only the most constrained setting with the best parameter of γ_t and defer results for other settings (including maximal constraint) to Appendix C.

Generic Graphs. We compare both the quality and the constraint validity of the generated generic graphs under the *minimal* constrained setting, where we choose the constraint parameter equal to its minimum value in the test set for the given constraint. For example, we consider the minimum number of edges in the graph for the Edge Count constraint. Table 2 shows the effect of plugging PRODIGY for sampling the two base models under 3 different constraints. We can note that the constraint validity with PRODIGY sampling is almost always close to 1, *i.e.*, almost all the generated graphs satisfy the constraint. PRODIGY increases the constraint validity in GDSS by at least 20% and at most 100% across 4 datasets and 3 different constraints. We can also note that the quality of the generated graphs is not compromised due to PRODIGY sampling. In particular, we find that the MMDs between the generated graphs and the constraint-filtered graphs under PRODIGY sampling are similar to the original sampling. Except for Grid, the average MMD is increased by only 0.07 from the original GDSS due to PRODIGY. Please refer to Appendix C for more analysis and visualizations.

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 3 compares the original GDSS and different PRODIGY sampling methods on a range of parameters for the Edge Count constraint ($|\mathbf{E}| \leq \mathcal{B} \in [|\mathbf{E}|_{\text{min}}, |\mathbf{E}|_{\text{max}}]$) on the Community dataset. This shows that our method ($\gamma_t > 0$) can satisfy constraints for any arbitrary parameter setting. We can note a trade-off that while increasing γ_t leads to higher constraint validity ($\textsc{Val}_{\mathcal{C}}$), it negatively affects the quality of generated graphs (an increase in the MMD scores). We also find that choosing a higher power for polynomial scheduling reduces the constraint validity since γ_t is low for most of the diffusion process.

For EDP-GNN, we just use a fixed $\gamma_t = 1$ as it only consists of a Langevin corrector and an identity predictor in the sampling stage (Niu et al., 2020; Jo et al., 2022; Song et al., 2020). Since there is no predictor step, each diffusion timestep is independent and is not iterative. Thus, projecting at each step has no effect on the dynamics. For GDSS, the parameters that we used for the minimal constraint generation are provided in Table 7 in Appendix C.

Molecular Graphs. Table 3 shows the effect of plugging PRODIGY in enabling the generation of molecules under different constraints. For the valency constraint, we consider the valencies $C^{4+}N^{5+}O^{2-}F^{-}$ in QM9 and $C^{4+}N^{3+}O^{2-}F^{-}P^{5+}S^{2+}Cl^{-}Br^{-}I^{-}$ in ZINC250k. For the Atom Count, we constrained the generated molecule to only contain C and O atoms for both datasets. We can note that PRODIGY is able to give a near-perfect constraint validity across the two datasets, while not compromising on the quality of the generated molecules. In particular, for the Atom Count constraint, we are able to generate molecules with only C and O with low enough NSPDK and FCD scores (*i.e.*, often lower than the original sampling). We can also note that PRODIGY allows us to increase the validity of the graph generation of GDSS using the Valency constraint. Results for the molecular weight and molecular property constraint are provided in Appendix C.

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. We train a simple graph convolutional network (Wu et al., 2019), as described in Section 2.1, 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

Table 2: Effect of PRODIGY on constrained generic graph generation. We choose the constraint parameter as the minimum from the test dataset. MMD results are evaluated between generated graphs and constraint-filtered test graphs.

		Community-small					Ego-small					Enzymes					Grid				
		Deg.↓	Clus.↓	Orb.↓	Avg.↓	$\text{VAL}_\mathcal{C}$ ↑	Deg.↓	Clus.↓	Orb.↓	Avg.↓	$\text{VAL}_\mathcal{C}$ ↑	Deg.↓	Clus.↓	Orb.↓	Avg.↓	$\text{VAL}_\mathcal{C}$ ↑	Deg.↓	Clus.↓	Orb.↓	Avg.↓	$\text{VAL}_\mathcal{C}$ ↑
Edge Count	EDP-GNN	0.362	0.366	0.125	0.285	**0.23**	0.199	0.469	0.036	0.235	0.23	0.117	0.12	0.004	0.080	0.56	1.005	0.033	0.455	0.498	0.75
	+PRODIGY	0.083	0.379	0.006	0.156	0.12	0.055	0.006	0.000	0.020	**0.62**	0.247	0.008	0.000	0.085	**0.95**	1.854	0.000	0.905	0.92	**1.00**
	GDSS	0.448	0.481	0.077	0.335	0.15	0.187	0.599	0.017	0.268	0.18	0.149	0.411	0.081	0.214	0.05	0.120	0.011	0.047	0.059	0.05
	+PRODIGY	0.539	1.096	0.015	0.550	**0.90**	0.104	0.054	0.001	0.053	**0.65**	0.616	0.966	0.026	0.536	**0.82**	1.249	0.002	0.604	0.618	**0.95**
Triangle Count	EDP-GNN	0.266	0.220	0.068	0.185	0.70	0.170	0.469	0.024	0.221	0.39	0.099	0.120	0.029	0.083	0.64	1.062	0.033	0.513	0.536	0.38
	+PRODIGY	0.179	0.595	0.267	0.347	**0.96**	1.34	0.000	0.018	0.453	**1.00**	1.127	0.000	0.047	0.391	**1.00**	1.996	0.000	0.978	0.991	1.00
	GDSS	0.319	0.187	0.049	0.185	0.70	0.160	0.599	0.005	0.255	0.32	0.236	0.222	0.016	0.158	0.03	0.154	0.011	0.050	0.072	0.00
	+PRODIGY	0.293	0.183	0.048	0.175	**0.90**	1.34	0.00	0.018	0.453	**1.00**	0.056	0.298	0.028	0.127	**0.96**	1.996	0.000	0.978	0.991	**1.00**
Degree	EDP-GNN	0.288	0.202	0.079	0.190	0.38	0.156	0.173	0.037	0.122	0.36	0.117	0.12	0.004	0.080	0.52	1.062	0.033	0.513	0.536	0.50
	+PRODIGY	0.117	0.726	0.252	0.365	**0.44**	0.042	0.022	0.000	0.022	**0.63**	0.242	0.000	0.000	0.081	**1.00**	1.717	0.000	0.958	0.892	**1.00**
	GDSS	0.350	0.203	0.051	0.201	0.40	0.131	0.238	0.018	0.129	0.32	0.158	0.217	0.037	0.137	0.40	0.154	0.011	0.050	0.072	0.00
	+PRODIGY	0.075	0.431	0.097	0.201	**1.00**	0.116	0.169	0.001	0.095	**0.68**	0.265	0.802	0.018	0.362	**1.00**	1.755	0.000	0.972	0.909	**1.00**

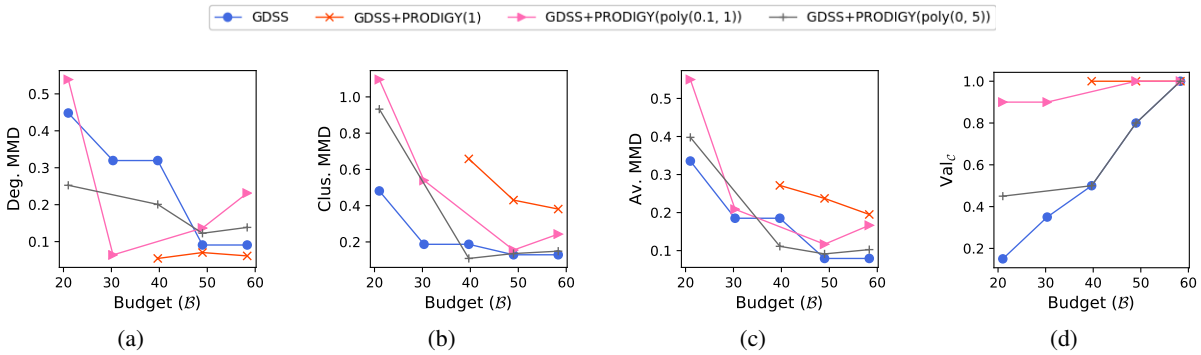


Figure 3: Comparison of methods in generating Community graphs with an arbitrary $\mathcal{B}$ number of edges. Lower MMD scores and higher constraint validity $\text{VAL}_\mathcal{C}$ are desired. We compare different values of γ_t parameter in our method. Note that the lack of a data point is when sampling leads to a trivial solution of zero edges or an empty graph.

Table 3: Effect of PRODIGY on the constrained molecular generation. The minimum constraint parameter is chosen for the evaluation. The base model EDP-GNN with default parameters (as in (Jo et al., 2022)) runs Out of Memory (denoted by OOM) for ZINC250k dataset.

		QM9 molecule graph dataset					ZINC250k molecule graph dataset				
		Val. (%) ↑	Novel. (%) ↑	NSPDK ↓	FCD ↓	$\text{VAL}_\mathcal{C}$ ↑	Val. (%) ↑	Novel. (%) ↑	NSPDK ↓	FCD ↓	$\text{VAL}_\mathcal{C}$ ↑
Valency	EDP-GNN	96.95	76.74	0.05	6.15	**0.97**	OOM	OOM	OOM	OOM	OOM
	+PRODIGY	96.29	76.68	0.07	6.23	0.96	OOM	OOM	OOM	OOM	OOM
	GDSS	95.72	81.04	0.00	2.47	0.88	97.01	100.00	0.02	14.04	0.94
	+PRODIGY	99.83	82.74	0.00	2.82	**0.99**	99.88	100.00	0.09	29.79	**0.99**
Atom Count	EDP-GNN	96.95	76.74	0.014	8.63	0.37	OOM	OOM	OOM	OOM	OOM
	+PRODIGY	98.06	54.36	0.018	5.66	1.00	OOM	OOM	OOM	OOM	OOM
	GDSS	95.72	81.04	0.012	7.282	0.33	97.08	100.00	0.027	16.006	0.13
	+PRODIGY	95.02	67.67	0.005	1.605	**1.00**	96.90	100.00	0.043	12.245	**0.99**

of Digress, we consider minimizing the mean absolute error (MAE) between the generated molecules and the first hundred molecules of QM9 by constraining the predicted property within a small bound from the median of these values (since the median minimizes the MAE). Table 4 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. For more details, please refer Appendix C.

Table 4: MAE in the molecular properties of the controllably generated molecules and the first 100 molecules in the QM9 dataset. μ is the dipole moment, and HOMO is the highest occupied molecular orbit.

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

6. Conclusion

Our work has shown that graph generation of diffusion models can be controlled by specific well-defined constraints. We hope that this opens future research avenues in studying the tradeoff between good quality and controllability of the generated data across domains. Future works can focus on further improving the quality of the generated graphs while satisfying pre-specified user-defined constraints. One can also study how to handle non-linear constraints on the inputs so that more complex molecular property prediction architectures can be used for the constraints.

References

Bar-Tal, O., Yariv, L., Lipman, Y., and Dekel, T. Multidiffusion: 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.

Costa, F. and De Grave, K. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 255–262. Omnipress; Madison, WI, USA, 2010.

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

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.

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

Langevin, M., Minoux, H., Levesque, M., and Bianciotto, M. Scaffold-constrained molecular generation. *Journal of Chemical Information and Modeling*, 60(12):5637–5646, 2020.

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.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

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

Liu, M., Yan, K., Oztekin, B., and Ji, S. GraphEBM: Molecular graph generation with energy-based models. In *Energy Based Models Workshop - ICLR 2021*, 2021. URL https://openreview.net/forum?id=Gc51PtL_zYw.

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31, 2018.

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.

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.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pp. 412–422. Springer, 2018.

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.

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.

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 auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Zang, C. and Wang, F. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 617–626, 2020.

Appendix

A. Related Work

Graph Generation. In the literature, different graph generation models have been proposed using autoregressive RNNs (You et al., 2018; Li et al., 2018), VAEs (Simonovsky & Komodakis, 2018), normalizing flows (Shi et al., 2020; Zang & Wang, 2020), EBMs (Liu et al., 2021), and diffusion models (Niu et al., 2020; Jo et al., 2022; Vignac et al., 2022). Among these, diffusion models have outperformed the other existing models by a significant margin. However, all these works focus on unconditional generation, *i.e.*, the task of generating realistic graphs given a set of training graphs. Our work differs as we study the first constrained generation for graph structures where an explicit constraint function is made available during the sampling time.

Controlled Generation. Different sampling strategies can be employed in a plug-and-play manner on top of a diffusion model to generate graphs with a specific property. In particular, one can do conditional generation (Song et al., 2020) by simply updating the score term with $\nabla \log p_t(\mathbf{G})$ with $\nabla_{\mathbf{G}} \log p_t(\mathbf{G}|\mathbf{y}) = \nabla_{\mathbf{G}} \log p_t(\mathbf{G}) + \nabla_{\mathbf{G}} \log p_t(\mathbf{y}|\mathbf{G})$, where $\nabla_{\mathbf{G}} \log p_t(\mathbf{G})$ is approximated with a parametrized neural network for unconditional generation and $p_t(\mathbf{y}|\mathbf{G})$ is obtained by training a separate classifier for the condition $\mathbf{y}$. However, when $\mathbf{y}$ is a known constraint function of $\mathbf{G}$, a differentiable probability distribution over $\mathbf{G}$ may not be possible. Recently, other sampling strategies have been proposed to control the generation for specific image-based controls (Bar-Tal et al., 2023), and text-based controls (Li et al., 2022). Specific graph-level controls remain an understudied area in the generation literature.

Projected Dynamics. Existing works have studied theoretical bounds on the number of steps required to sample using a Projected Langevin Markov chain (Bubeck et al., 2018; Hsieh et al., 2018) but it is not known whether they would be effective to generate constrained graphs from real-world distributions. Thus, we conduct systematic experiments in this direction. In particular, we show superior controllability of graph generation with our sampling by fixing the number of edges, triangles, and maximum degree.

B. Projection Operators

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

pending on the form of $\mathbf{h}_{\mathcal{C}}(\mathbf{X}, \mathbf{A})$.

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

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

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

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

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

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

Projection Operators for constraints. In this section, we discuss projection operators for the constraints mentioned in Table 1. We first solve for $\varphi_{\boldsymbol{\mu}}^A = \mathbf{S}_A^*$ and $\varphi_{\boldsymbol{\mu}}^X = \mathbf{S}_X^*$. 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.

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

Find $\varphi_{\boldsymbol{\mu}}$**.** We have $\mathbf{h}_{\mathcal{C}}(\mathbf{S}_X, \mathbf{S}_A) = h_{\mathcal{C}}(\mathbf{S}_A) = \frac{1}{2}\mathbf{1}^T \mathbf{S}_A \mathbf{1} - \mathcal{B}$, $\mathbf{S}_A \in [\mathbf{0}, \mathbf{1}]$, $\text{Diag}(\mathbf{S}_A) = \mathbf{0}$, $\mathbf{S}_A^T = \mathbf{S}_A$. Then, we can note that $\nabla_{\mathbf{S}_X} h_{\mathcal{C}} = \mathbf{0}$. Thus, we solve for $\mathbf{S}_A^*$ in $\mathbf{S}_A^* + \mu \nabla_{\mathbf{S}_A} h_{\mathcal{C}}(\mathbf{S}_A^*) + \boldsymbol{\Lambda} = \mathbf{A}$. Since $\nabla_{\mathbf{S}_A} h_{\mathcal{C}} = \frac{1}{2}\mathbf{1}\mathbf{1}^T$, we get $\mathbf{S}_A^* = \mathbf{A} - \frac{\mu}{2}\mathbf{1}\mathbf{1}^T - \boldsymbol{\Lambda}$. Satisfying $\text{Diag}(\mathbf{S}_A^*) = \mathbf{0}$, $(\mathbf{S}_A^*)^T = \mathbf{S}_A^*$ (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, $\mathbf{S}_A^* = \mathbf{A} - \mu/2\mathbf{1}\mathbf{1}^T + \mu/2\mathbf{I}$ 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 $\mathbf{S}_A^*$ 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 $\mathbf{h}_{\mathcal{C}}(\mathbf{S}_X, \mathbf{S}_A) = h_{\mathcal{C}}(\mathbf{S}_A) = \frac{1}{6}\text{tr}(\mathbf{S}_A^3) - T$, $\mathbf{S}_A \in [\mathbf{0}, \mathbf{1}]$, $\text{Diag}(\mathbf{S}_A) = \mathbf{0}$, $\mathbf{S}_A^T = \mathbf{S}_A$. Then, we can note that $\nabla_{\mathbf{S}_X} h_{\mathcal{C}} = \mathbf{0}$. Thus, we solve for $\mathbf{S}_A^*$ in $\mathbf{S}_A^* + \mu \nabla_{\mathbf{S}_A} h_{\mathcal{C}}(\mathbf{S}_A^*) + \boldsymbol{\Lambda} = \mathbf{A}$. Since $\nabla_{\mathbf{S}_A} h_{\mathcal{C}} = \frac{1}{2}\mathbf{A}^2$, we get $\mathbf{S}_A^* = \mathbf{A} - \frac{\mu}{2}\mathbf{A}^2 - \boldsymbol{\Lambda}$. Satisfying $\text{Diag}(\mathbf{S}_A^*) = \mathbf{0}$, $(\mathbf{S}_A^*)^T = \mathbf{S}_A^*$ (given these hold for $\mathbf{A}$) implies $\boldsymbol{\Lambda} = \mathbf{0}$. Thus, $\mathbf{S}_A^* = \mathbf{A} - \mu\mathbf{A}^2/2$ followed by clamping between $[\mathbf{0}, \mathbf{1}]$.

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 $\frac{1}{6}\text{tr}(P_{[\mathbf{0},\mathbf{1}]}(\mathbf{A} - \mu\mathbf{A}^2/2)^3) - T$ 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 $\mathbf{S}_A^*$ 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 $\mathbf{S}_A^*$ 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. Max Degree ($d_{\max} = \mathbf{A}\mathbf{1} \leq \delta_d \mathbf{1}$)

Find $\varphi_{\boldsymbol{\mu}}$**.** We have $\mathbf{h}_{\mathcal{C}}(\mathbf{S}_X, \mathbf{S}_A) = \mathbf{S}_A\mathbf{1} - \delta_d\mathbf{1}$, $\mathbf{S}_A \in [\mathbf{0}, \mathbf{1}]$, $\text{Diag}(\mathbf{S}_A) = \mathbf{0}$, $\mathbf{S}_A^T = \mathbf{S}_A$. Then, we can note that $\nabla_{\mathbf{S}_X} \mathbf{h}_{\mathcal{C}} = \mathbf{0}$ and we solve for $\mathbf{S}_A^*$ in $\mathbf{S}_A^* + \boldsymbol{\mu} \cdot \nabla_{\mathbf{S}_A} \mathbf{h}_{\mathcal{C}}(\mathbf{S}_A^*) + \boldsymbol{\Lambda} = \mathbf{A}$. In other words, for each row i, we get $\mathbf{S}_A^*[i, :] = \mathbf{A}[i, :] - \mu_i \mathbf{1} - \boldsymbol{\Lambda}[i, :]$ since $\nabla_{\mathbf{S}_A} h_{\mathcal{C}}^{(i)} = \mathbf{1}$. 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 $\mathbf{S}_A^*$ has no non-zero diagonal entries. Thus, we get

$\mathbf{S}_A^* = \mathbf{A} - \frac{1}{2}(\boldsymbol{\mu}\mathbf{1}^T + \mathbf{1}\boldsymbol{\mu}^T) + \text{Diag}(\boldsymbol{\mu})$ 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 $\mathbf{S}_A^*$ 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 $\mathbf{S}_A^* = \mathbf{A} - \frac{1}{2}(\boldsymbol{\mu}\mathbf{1}^T + \mathbf{1}\boldsymbol{\mu}^T) + \text{Diag}(\boldsymbol{\mu})$ 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 $\mathbf{h}_\mathcal{C}(\mathbf{S}_X, \mathbf{S}_A) = \mathbf{S}_X^T\mathbf{1} \leq \mathbf{c}$, $\mathbf{S}_X \in [\mathbf{0}, \mathbf{1}]$. Then, we can note that $\nabla_{\mathbf{S}_A}\mathbf{h}_\mathcal{C}(\mathbf{S}_X, \mathbf{S}_A) = \mathbf{0}$ and for each column or atom type in $\mathbf{X}$, we get $\mathbf{S}_X^*[:,j] = \mathbf{X}[:,j] - \mu_j\mathbf{1}^T$ since $\nabla_{\mathbf{S}_X}\mathbf{h}_\mathcal{C} = \mathbf{1}$. Thus, we get $\mathbf{S}_X^* = P_{[\mathbf{0},\mathbf{1}]}(\mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T)$.

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 $(\mathbf{S}_X^*[:,j] - \mu_j\mathbf{1}^T)\mathbf{1} = c_j$. 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 $\mathbf{h}_\mathcal{C}(\mathbf{S}_X, \mathbf{S}_A) = h_\mathcal{C}(\mathbf{S}_X, \mathbf{S}_A) = \mathbf{1}^T\mathbf{S}_X\mathbf{m} \leq W$, $\mathbf{S}_X \in [\mathbf{0}, \mathbf{1}]$. Then, $\nabla_{\mathbf{S}_A}h_\mathcal{C} = \mathbf{0}$ and $\nabla_{\mathbf{S}_X}h_\mathcal{C}(\mathbf{S}_X, \mathbf{S}_A) = \mathbf{1}\mathbf{m}^T$, which gives us $\mathbf{S}_X^* = \mathbf{X} - \mathbf{1}\boldsymbol{\mu}^T$ 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_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 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. Molecular Property ($\mathbf{1}^T\hat{\mathbf{A}}^k\mathbf{X}\Theta \leq p$)

Find $\varphi_{\boldsymbol{\mu}}$. We have $\mathbf{h}_\mathcal{C}(\mathbf{S}_X, \mathbf{S}_A) = h_\mathcal{C}(\mathbf{S}_X, \mathbf{S}_A) = \mathbf{1}^T\hat{\mathbf{A}}^k\mathbf{X}\boldsymbol{\Theta} - p$, $\mathbf{S}_X \in [\mathbf{0}, \mathbf{1}]$. We fix $\mathbf{A}$ and thus, assume $\mathbf{S}_A = P_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})$. Let $\hat{P}_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})$ denote the normalized adjacency matrix corresponding to $P_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})$. Then, $\nabla_{\mathbf{S}_A}h_\mathcal{C} = \mathbf{0}$ and $\nabla_{\mathbf{S}_X}h_\mathcal{C}(\mathbf{S}_X, \mathbf{S}_A) = (\hat{P}_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})^k)^T\mathbf{1}\boldsymbol{\Theta}^T$, which gives us $\mathbf{S}_X^* = \mathbf{X} - \mu(\hat{P}_{[\mathbf{0},\mathbf{3}]}(\mathbf{A})^k)^T\mathbf{1}\boldsymbol{\Theta}^T$ 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 $O(n^3(\mu_1/s + \log(\mu_1/\xi)))$.

B.8. Lower bound constraint

We note that we can also handle a lower-bound constraint by just reversing the order inside $h_{\mathcal{C}}$. For instance, let us consider the Molecular Property constraint, then we can have $h^1_{\mathcal{C}} = p_1 - \mathbf{1}^T\hat{\mathbf{A}}^k\mathbf{X}\mathbf{\Theta}$ and $h^2_{\mathcal{C}} = \mathbf{1}^T\hat{\mathbf{A}}^k\mathbf{X}\mathbf{\Theta} - p_2$. These can be solved together as we now have $\mu_1(p_1 - \mathbf{1}^T\hat{\mathbf{A}}^k\mathbf{X}\mathbf{\Theta}) = 0$ and $\mu_2(\mathbf{1}^T\hat{\mathbf{A}}^k\mathbf{X}\mathbf{\Theta} - p_2) = 0$. Given $p_1 \leq p_2$, for any μ, either $h^1_{\mathcal{C}} > 0$ or $h^2_{\mathcal{C}} > 0$ or both $h^1_{\mathcal{C}} \leq 0$ and $h^2_{\mathcal{C}} \leq 0$. Thus, we get the projection operator as:

$$\Pi_{\mathcal{C}}(G) = \begin{cases} \varphi_0(\mathbf{G}) \ ; p_1 \leq h_{\mathcal{C}}(\varphi_0(\mathbf{G})) \leq p_2 \\ \varphi_\mu(\mathbf{G}) \ ; h_{\mathcal{C}}(\varphi_0(\mathbf{G})) > p_2 \geq p_1; h_{\mathcal{C}}(\varphi_\mu(\mathbf{G})) = p_2 \\ \varphi_\mu(\mathbf{G}) \ ; h_{\mathcal{C}}(\varphi_0(\mathbf{G})) < p_1 \leq p_2; h_{\mathcal{C}}(\varphi_\mu(\mathbf{G})) = p_1 \end{cases} \quad (3)$$

such that both the equations can be solved using the bisection method. We use this approach for Molecular Property constraint by bounding the predicted property to be within an error term ε from the median property value.

C. Additional Experiments

C.1. Method Details

Following existing works on continuous diffusion models on graphs (Jo et al., 2022; Niu et al., 2020), we simply round the continuous graph at the end to obtain a discrete graph. In particular, we do $\lfloor \mathbf{A}_0 \rfloor$ to obtain the discrete adjacency matrix and the categorical attribute of each node i is found as $\arg\max_i \mathbf{X}_0[i]$.

For generic graphs, we compute MMDs for the distributions of degree (degree), clustering coefficient (cluster), and the number of occurrences of orbits with 4 nodes (orbit). For molecules, we use the Fréchet ChemNet Distance (FCD) (Preuer et al., 2018) and Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) MMD (Costa & De Grave, 2010) between the generated and *constrained* test graphs. We also consider the validity, uniqueness, and novelty of the generated molecules to evaluate their quality. Note that validity of a molecule is calculated without doing valency correction while allowing the atoms to have a formal charge, if applicable.

C.2. 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. **Grid** is a dataset of 100 standard 2D grids.

We also consider these 2 molecular graph datasets:

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

C.3. Results

Maximal Constrained Generation We first show that the base performance of the diffusion model is not affected due to PRODIGY sampling when we consider a *maximal* constraint on the generation. A constraint is termed *maximal* when the given constraint is satisfied by the whole test dataset. For example, if we set $\mathcal{B}$ to be the maximum number of edges in the dataset, then, all the test graphs will satisfy the constraint $|\mathbf{E}| \leq \mathcal{B}$. Graph generation under such a constraint should thus be similar to unconstrained generation since the unconstrained generated graphs also satisfy the maximal constraint. Here, we compare the quality of graphs generated by PRODIGY under such a maximal constraint with state-of-the-art (unconstrained) graph generation methods.

Tables 5 and 6 compare the results of adding PRODIGY to the two base models on the graph generation for generic and molecular graphs respectively. For the maximal constraint, we consider the maximum number of edges for the generic graphs and the maximum molecular weight for molecular graphs. We find that the generation performance of sampling PRODIGY under a maximal constraint is competitive with their unconstrained counterparts. In particular, for molecular generation, we can note that PRODIGY sampling remains within a factor of 7% for all the metrics for GDSS. Thus, PRODIGY maintains the state-of-the-art performance of these continuous-time diffusion models (*i.e.*, GDSS (Jo et al., 2022) and EDP-GNN (Niu et al., 2020)) when the constraint is trivially satisfied by the test distribution (and also, the unconstrained methods).

Visualizations Figures 4, 5, 6, 7 compare generations of GDSS with GDSS+PRODIGY sampling on different datasets given a maximal constraint. On the other hand, generations by our method for minimal constraint for each dataset are provided in Figures 8, 9, 10, 11. We can observe that the generated graphs can satisfy the given constraints while being close to the original distribution.

Standard Deviations We run our sampling for 3 different random seeds and find standard deviations of at most 0.05 for all the MMD metrics and up to 0.00 for the constraint va-

Table 5: Comparison of maximal constraint generation with unconstrained generation of generic graph datasets on the MMD of different properties (lower is better). ⋆ We could not reproduce the results for EDP-GNN and GDSS as reported in their papers. The values for the other methods are taken directly from their papers or the implementation of GDSS.

	Community-small				Ego-small				Enzymes				Grid			
	Deg.↓	Clus.↓	Orb.↓	Avg.↓	Deg.↓	Clus.↓	Orb.↓	Avg.↓	Deg.↓	Clus.↓	Orb.↓	Avg.↓	Deg.↓	Clus.↓	Orb.↓	Avg.↓
Deep-GMG (Li et al., 2018)	0.220	0.950	0.400	0.523	0.040	0.100	0.020	0.053	-	-	-	-	-	-	-	-
Graph-RNN (You et al., 2018)	0.080	0.120	0.040	0.080	0.090	0.220	0.003	0.104	0.017	0.062	0.046	0.042	0.064	0.043	0.021	0.043
Graph-VAE (Simonovsky & Komodakis, 2018)	0.350	0.980	0.540	0.623	0.130	0.170	0.050	0.117	1.369	0.629	0.191	0.730	1.619	0.0	0.919	0.846
GNF (Liu et al., 2019)	0.200	0.200	0.110	0.170	0.030	0.100	0.001	0.044	-	-	-	-	-	-	-	-
EDP-GNN (Niu et al., 2020)⋆	0.120	0.071	0.046	0.079	0.020	0.043	0.006	0.023	1.011	0.791	0.239	0.681	1.062	0.033	0.513	0.536
+PRODIGY	0.503	0.116	0.247	0.288	0.018	0.048	0.005	0.024	1.306	1.000	0.261	0.856	1.861	0.000	0.974	0.945
GDSS (Jo et al., 2022)⋆	0.170	0.090	0.079	0.113	0.023	0.010	0.013	0.015	0.034	0.078	0.003	0.038	0.154	0.011	0.050	0.072
+PRODIGY	0.132	0.077	0.044	0.084	0.029	0.030	0.013	0.024	0.265	0.466	0.020	0.250	0.514	0.022	0.437	0.324

Table 6: Comparison of maximal constraint generation with unconstrained generation of molecular graph datasets. Lower NSPDK, FCD and higher Val. (%), Novel. (%) are desired. The values for the other methods are taken directly from their papers or GDSS (Jo et al., 2022). The base model EDP-GNN with default parameters (as in (Jo et al., 2022)) runs Out of Memory (denoted by OOM) for ZINC250k dataset.

	QM9 molecule graph dataset				ZINC250k molecule graph dataset			
	Val. w/o corr. (%)	Novel. (%) ↑	NSPDK ↓	FCD ↓	Val. w/o corr. (%)	Novel. (%) ↑	NSPDK ↓	FCD ↓
GraphAF (Shi et al., 2020)	67	88.83	0.020	5.268	68	100.00	0.044	16.289
MoFlow (Zang & Wang, 2020)	91.36	98.10	0.017	4.467	63.11	100.00	0.046	20.931
GraphEBM (Liu et al., 2021)	8.22	97.01	0.030	6.143	5.29	100.00	0.212	35.471
EDP-GNN (Niu et al., 2020)	96.95	76.74	0.055	6.15	OOM	OOM	OOM	OOM
+PRODIGY	97.03	76.95	0.055	6.15	OOM	OOM	OOM	OOM
GDSS (Jo et al., 2022)	95.72	86.27	0.003	2.900	97.01	100.00	0.019	14.656
+PRODIGY	96.10	81.04	0.003	2.477	96.10	100.00	0.017	15.696

Table 7: PRODIGY parameters for each setting for the GDSS in the generic graph datasets

Constraint	Community-small	Ego-small	Enzymes	Grid
Edge Count	poly$(0.1, 1)$	poly$(0.1, 5)$	poly$(0, 1)$	poly$(0, 5)$
Triangle Count	poly$(0, 1)$	poly$(0.1, 5)$	poly$(0.1, 5)$	poly$(0.1, 5)$
Degree	poly$(0, 1)$	poly$(0.1, 5)$	poly$(0, 1)$	poly$(0.1, 5)$

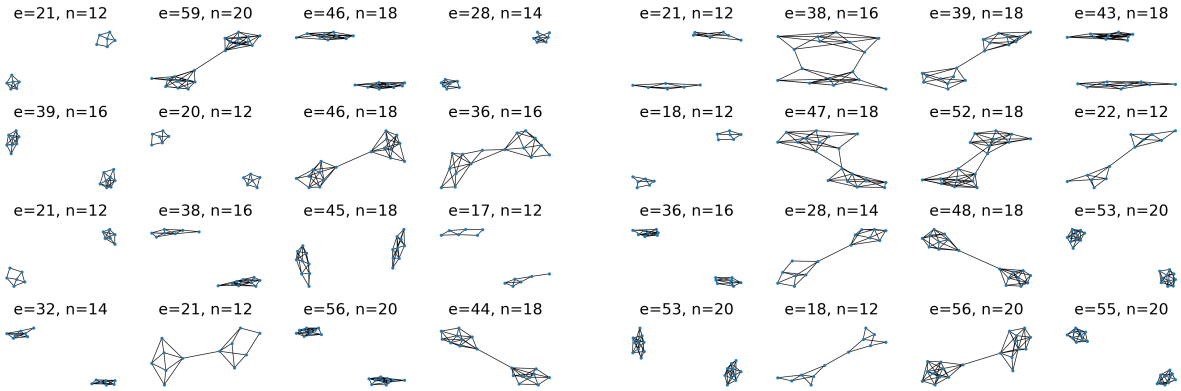
Table 8: Time taken (in seconds) per diffusion timestep. ⋆ denotes the time taken by the original (unconstrained) GDSS sampling.

	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
Grid	0.24	0.52	0.24	0.43

lidity metric (for a given set of parameters for the PRODIGY sampling).

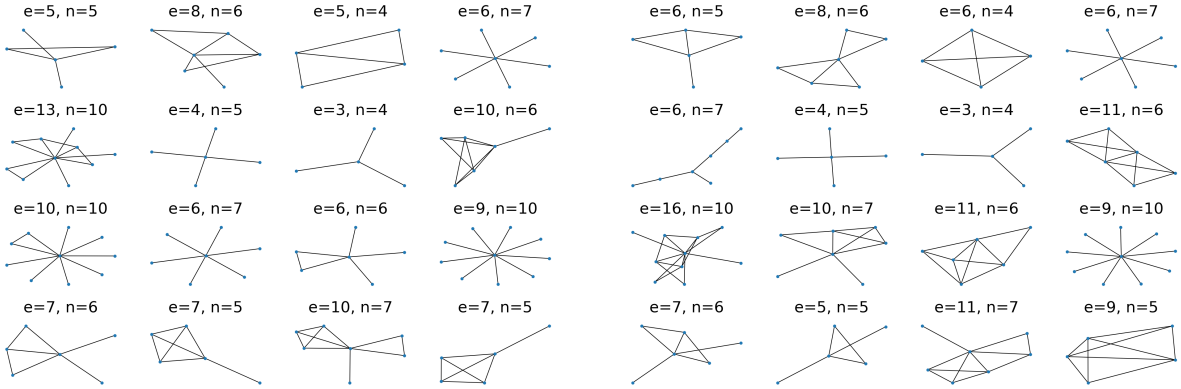
Running Time We report the PRODIGY sampling time for different constraints on different datasets with GDSS. In particular, Table 8 reports the sampling time taken per diffusion timestep.

Molecular Weight The minimal constraint is satisfied by less than $0.1 - 0.5\%$ of the molecules in the test set (*i.e.* just 10-50 molecules out of a total 10000). This makes the quality metrics difficult to calculate. However, we are still able to get 75.36% of the molecules to satisfy the constraint for GDSS.



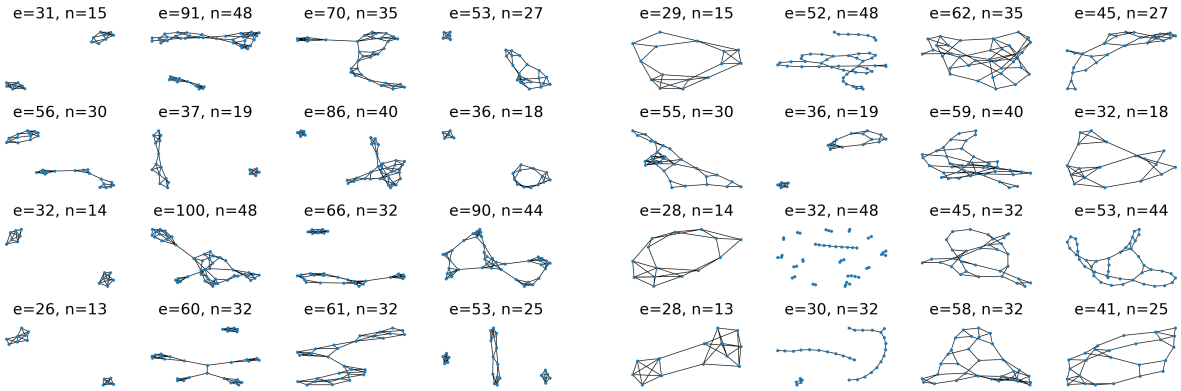
(a) GDSS (b) GDSS+PRODIGY

Figure 4: Comparison of maximal constraint generation on Community-small dataset



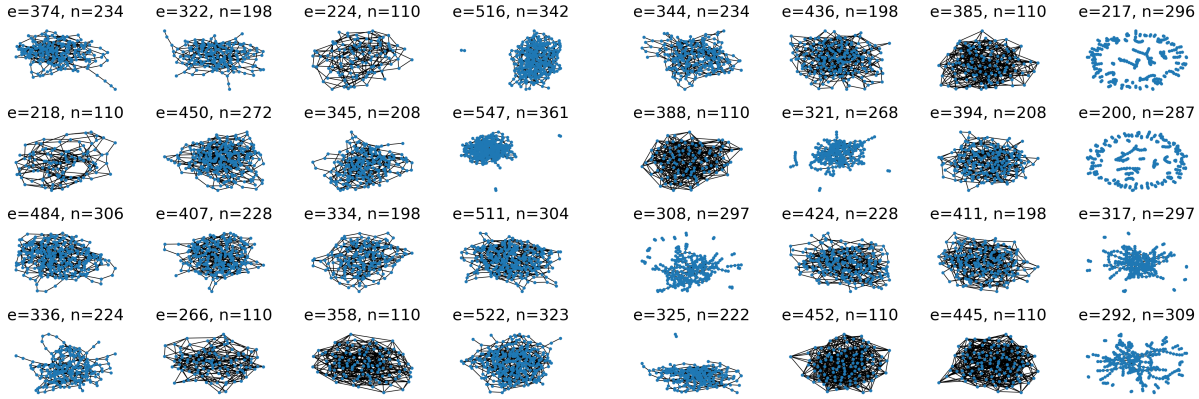
(a) GDSS (b) GDSS+PRODIGY

Figure 5: Comparison of maximal constraint generation on Ego-small dataset



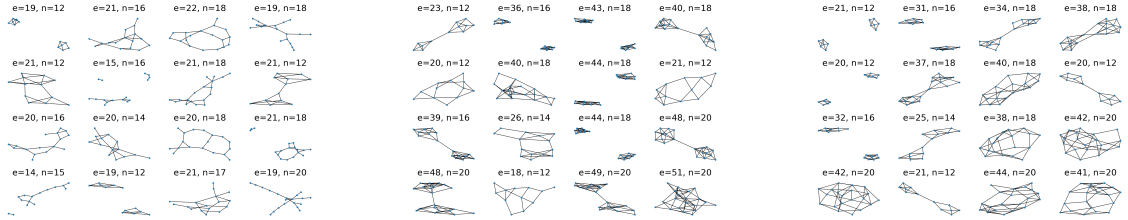
(a) GDSS (b) GDSS+PRODIGY

Figure 6: Comparison of maximal constraint generation on Enzymes dataset



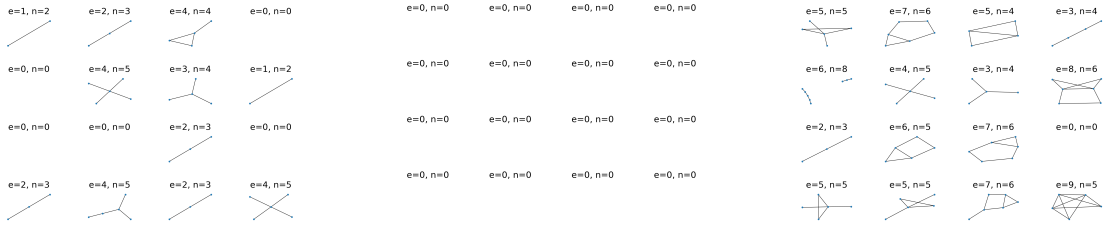
(a) GDSS (b) GDSS+PRODIGY

Figure 7: Comparison of maximal constraint generation on Grid dataset



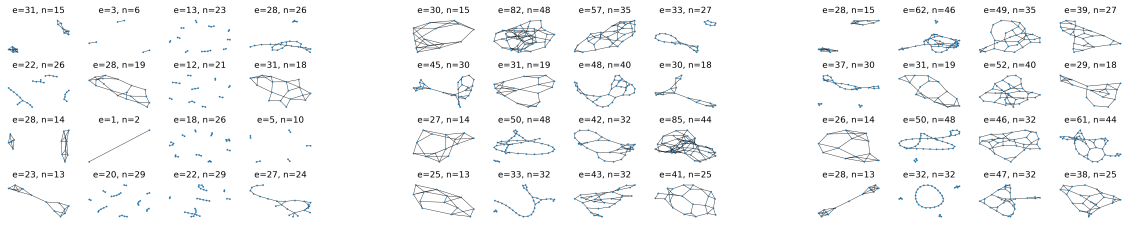
(a) Edge-Count (≤ 21) (b) Triangle-Count (≤ 30) (c) Max-Degree (≤ 5)

Figure 8: GDSS+PRODIGY generations for the *minimal* constrained setting on Community-small



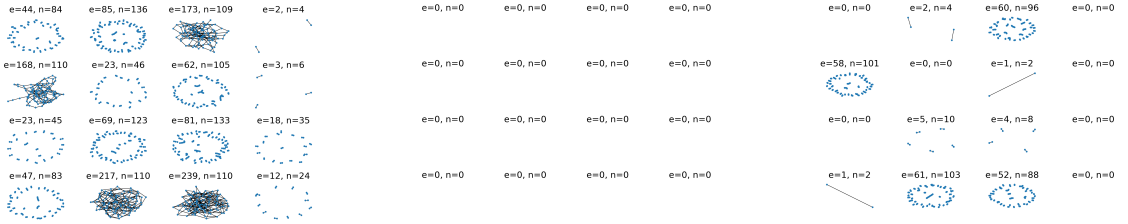
(a) Edge-Count (≤ 3) (b) Triangle-Count (≤ 0) (c) Max-Degree (≤ 5)

Figure 9: GDSS+PRODIGY generations for the *minimal* constrained setting on Ego-small.



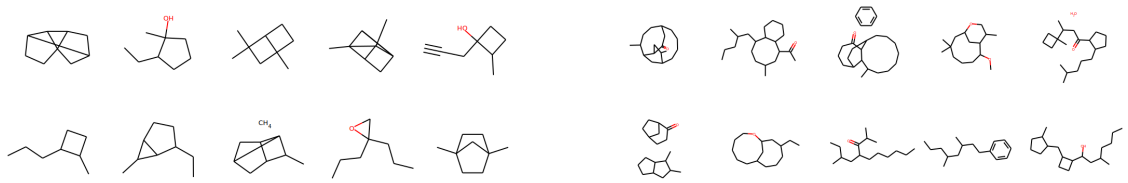
(a) Edge-Count (≤ 29) (b) Triangle-Count (≤ 18.67) (c) Max-Degree (≤ 5.11)

Figure 10: GDSS+PRODIGY generations for the *minimal* constrained setting on Enzymes



(a) Edge-Count (≤ 21) (b) Triangle-Count (≤ 0) (c) Max-Degree (≤ 4)

Figure 11: GDSS+PRODIGY generations for the *minimal* constrained setting on Grid



(a) QM9 (b) ZINC250k

Figure 12: GDSS+PRODIGY generations for the Atom-Count constraint to generate molecules with only Carbon and Oxygen atoms. We pick the 10 novel molecules (*i.e.*, not in the dataset) with the maximum Tanimoto similarity with the test dataset.