Unsupervised Discovery of Steerable Factors When Graph Deep Generative Models Are Entangled

Shengchao Liu *shengchao.liu@umontreal.ca*
Quebec AI Institute (Mila)
University de Montréal

Chengpeng Wang *cw83@illinois.edu*
University of Illinois Urbana-Champaign

Jiarui Lu *jiarui.lu@umontreal.ca*
Quebec AI Institute (Mila)
University de Montréal

Weili Nie *wnie@nvidia.com*
Nvidia Research

Hanchen Wang *hw501@cam.ac.uk*
University of Cambridge

Zhuoxinran Li *zhuoxinran.li@mail.utoronto.ca*
University of Toronto

Bolei Zhou *bolei@cs.ucla.edu*
University of California, Los Angeles

Jian Tang *jian.tang@hec.ca*
Quebec AI Institute (Mila)
HEC Montréal

Reviewed on OpenReview: *https://openreview.net/forum?id=wyU3Q4gahM*

Abstract

Deep generative models (DGMs) have been widely developed for graph data. However, much less investigation has been carried out on understanding the latent space of such pretrained graph DGMs. These understandings possess the potential to provide constructive guidelines for crucial tasks, such as graph controllable generation. Thus in this work, we are interested in studying this problem and propose GraphCG, a method for the unsupervised discovery of steerable factors in the latent space of pretrained graph DGMs. We first examine the representation space of three pretrained graph DGMs with six disentanglement metrics, and we observe that the pretrained representation space is entangled. Motivated by this observation, GraphCG learns the steerable factors via maximizing the mutual information between semantic-rich directions, where the controlled graph moving along the same direction will share the same steerable factors. We quantitatively verify that GraphCG outperforms four competitive baselines on two graph DGMs pretrained on two molecule datasets. Additionally, we qualitatively illustrate seven steerable factors learned by GraphCG on five pretrained DGMs over five graph datasets, including two for molecules and three for point clouds.

1 Introduction

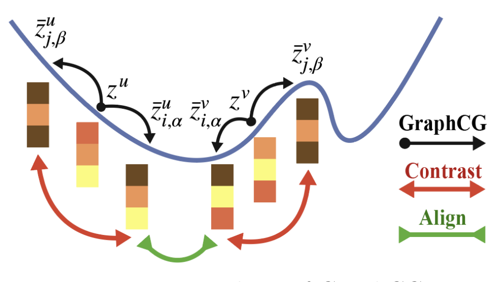
The graph is a general format for many real-world data. For instance, molecules can be treated as graphs (Duvenaud et al., 2015; Gilmer et al., 2017) where the chemical atoms and bonds correspond to the topological nodes and edges respectively. Processing point clouds as graphs is also a popular strategy (Shi & Rajkumar, 2020; Wang et al., 2020), where points are viewed as nodes and edges are built among the nearest neighbors. Many existing works on deep generative models (DGMs) focus on modeling the graph data and improving the synthesis quality. However, understanding the pretrained graph DGMs and their learned representations has been much less explored. This may hinder the development of important applications like *graph controllable generation* (also referred to as *graph editing*) and the discovery of interpretable graph structure.

Concretely, the graph controllable generation task refers to modifying the steerable factors of the graph so as to obtain graphs with desired properties easily (Drews, 2000; Pritch et al., 2009). This is an important task in many applications, but traditional methods (*e.g.*, manual editing) possess inherent limitations under particular circumstances. A typical example is molecule editing: it aims at modifying the substructures of molecules (Mihalić & Trinajstić, 1992) and is related to certain key tactics in drug discovery like functional group change (Ertl et al., 2020) and scaffold hopping (Böhm et al., 2004; Hu et al., 2017). This is a routine task in pharmaceutical companies, yet, relying on domain experts for manual editing can be subjective or biased (Drews, 2000; Gomez, 2018). Different from previous works, this paper starts to explore unsupervised graph editing on pretrained DGMs. It can act as a complementary module to conventional methods and bring many crucial benefits: (1) It enables efficient graph editing in a large-scale setting. (2) It alleviates the requirements for extensive domain knowledge for factor change labeling. (3) It provides a constructive perspective for editing preference, which can reduce biases from the domain experts.

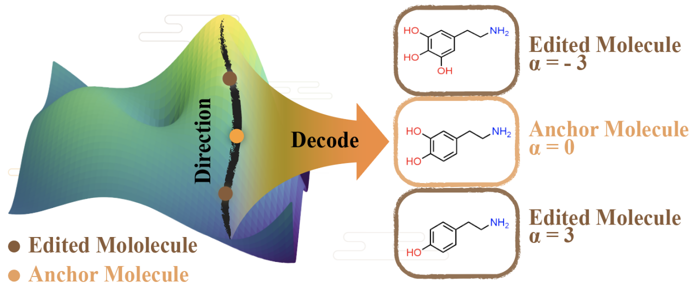
Disentanglement for editing. One core property relevant to the general unsupervised data editing using DGMs is disentanglement. While there does not exist a widely-accepted definition of disentanglement, the key intuition (Locatello et al., 2019) is that a disentangled representation should separate the distinct, informative, and steerable factors of variations in the data. Thus, the controllable generation task would become trivial with the disentangled DGMs as the backbone. Such a disentanglement assumption has been widely used in generative modeling on the image data, *e.g.*, β-VAE (Higgins et al., 2017) learns disentangled representation by forcing the representation to be close to an isotropic unit Gaussian. However, it may introduce extra constraints on the formulations and expressiveness of DGMs (Higgins et al., 2017; Ridgeway & Mozer, 2018; Eastwood & Williams, 2018; Wu et al., 2021).

Entanglement on pretrained graph DGMs. Thus for graph data, one crucial question arises: *Is the latent representation space from pretrained graph DGMs disentangled or not?* In image generation, a series of work (Collins et al., 2020; Shen et al., 2020a; Härkönen et al., 2020; Tewari et al., 2020; Wu et al., 2021) has shown the disentanglement properties on pretrained DGMs. However, such property of pretrained graph DGMs is much less explored. In Section 3, we first study the latent space of three pretrained graph DGMs and empirically illustrate that the learned space is not perfectly disentangled or entangled. In what follows, we adopt the term "entangled" for graph DGMs.

Our approach. This observation then raises the second question: *Given a pretrained yet entangled DGM for graph data, is there a flexible framework enabling the graph controllable generation in an unsupervised manner?* To tackle this problem, we propose a model-agnostic framework coined GraphCG for unsupervised graph controllable generation. GraphCG has two main phases, as illustrated in Figure 1. During the learning phase (Figure 1(a)), GraphCG starts with the assumption that the steerable factors can be learned by maximizing the mutual information (MI) among the semantic directions. We formulate GraphCG using an energy-based model (EBM), which offers a large family of solutions. Then during the inference phase, with the learned semantic directions, we can carry out the editing task by moving along the direction with certain step sizes. As the example illustrated in Figure 1(b), the graph structure (hydroxyl group) changes consistently along the learned editing direction. For evaluation, we qualitatively verify the learned directions of five pretrained graph DGMs. Particularly for the molecular datasets, we propose a novel evaluation metric called sequence monotonic ratio (SMR) to quantitatively measure the structure change over the output sequences.



(a) Learning phase of GraphCG.



(b) Inference phase of GraphCG.

Figure 1: (a) The learning phase. Given two latent codes $\boldsymbol{z}^u$ and $\boldsymbol{z}^v$, we edit the four latent representations along i-th and j-th direction with step size α and β respectively. The goal of GraphCG, is to align the positive pair ($\bar{\boldsymbol{z}}^u_{i,\alpha}$ and $\bar{\boldsymbol{z}}^v_{i,\alpha}$), and contrast them with $\bar{\boldsymbol{z}}^u_{j,\beta}$ and $\bar{\boldsymbol{z}}^v_{j,\beta}$ respectively. (b) The inference phase. We will first sample an anchor molecule and adopt the learned directions in the learning phase for editing. With step size $\alpha \in [-3, 3]$, we can generate a sequence of molecules. Specifically, after decoding, there is a functional group change shown up: the number of hydroxyl groups decreases along the sequence in the decoded molecules.

Our contributions. (1) We conduct an empirical study on the disentanglement property of three pretrained graph DGMs using six metrics, and we observe that the latent space of these pretrained graph DGMs is entangled. (2) We propose a model-agnostic method called GraphCG for the unsupervised graph controllable generation or graph editing. GraphCG aims at learning the steerable factors by maximizing the mutual information among corresponding directions, and its outputs are sequences of edited graphs. (3) We quantitatively verify that GraphCG outperforms four competitive baselines when evaluated on two pretrained graph DGMs over two molecule datasets. (4) We further qualitatively strengthen the effectiveness of GraphCG by illustrating seven semantic factors on five pretrained graph DGMs over five graph datasets, including two for molecular graphs and three for point clouds.

Related work. Recent works leverage the DGMs for various controllable generation tasks (Chen et al., 2018a; Xia et al., 2021), where the inherent assumption is that the learned latent representations encode rich semantics, and thus traversal in the latent space can help steer factors of data (Jahanian et al., 2019; Shen et al., 2020b; Härkönen et al., 2020). Among them, one research direction (Shen et al., 2020b; Nie et al., 2021) is using supervised signals to learn the semantic-rich directions, and most works on editing the graph data focus on the supervised setting (Jin et al., 2020a; Veber et al., 2002; You et al., 2018). However, these approaches can not be applied to many realistic scenarios where extracting the supervised labels is difficult. Another research line (Härkönen et al., 2020; Shen & Zhou, 2021; Ren et al., 2021) considers discovering the latent semantics in an unsupervised manner, but these unsupervised methods are designed to be either model-specific or task-specific, making them not directly generalizable to the graph data. A more comprehensive discussion is in Appendix B.

2 Background and Problem Formulation

Graph and deep generative models (DGMs). Each graph data (including nodes and edges) is denoted as $\boldsymbol{x} \in \mathcal{X}$, where $\mathcal{X}$ is the data space, and DGMs learn the data distribution, *i.e.*, $p(\boldsymbol{x})$. Our proposed graph editing method (GraphCG) is model-agnostic or DGM-agnostic, so we briefly introduce the mainstream DGMs for graph data as below. Variational auto-encoder (VAE) (Kingma & Welling, 2013; Higgins et al., 2017) measures a variational lower bound of $p(\boldsymbol{x})$ by introducing a proposal distribution; flow-based model (Dinh et al., 2014; Rezende & Mohamed, 2015) constructs revertible encoding functions such that the data distribution can be deterministically mapped to a prior distribution. Note that these mainstream DGMs, either explicitly or implicitly, contain an encoder ($f(\cdot)$) and a decoder ($g(\cdot)$) parameterized by neural networks:

$$\boldsymbol{z} = f(\boldsymbol{x}), \qquad \boldsymbol{x}' = g(\boldsymbol{z}), \tag{1}$$

where $\boldsymbol{z} \in \mathcal{Z}$ is the latent representation, $\mathcal{Z}$ is the latent space, and $\boldsymbol{x}'$ is the reconstructed output graph. Since in the literature (Shen & Zhou, 2021; Shen et al., 2020b), people also call latent representations as latent codes or latent vectors, in what follows, we will use these terms interchangeably. Note that the encoding and decoding functions in Equation (1) (f, g) can be stochastic depending on the DGMs we are using.

Steerable factors. The steerable factors are key attributes of DGMs, referring to the semantic information of data that we can explicitly discover from the pretrained DGMs. For instance, existing works (Shen & Zhou, 2021; Ren et al., 2021) have shown that using unsupervised methods on facial image DGM can discover factors such as the size of eyes, smiles, noses, etc. In this work, we focus on the steerable factors of graph data, which are data- and task-specific. Yet, there is one category of factor that is commonly shared among all the graph data: the **structure** factor. Concretely, these steerable factors can be the functional groups in molecular graphs and shapes or sizes in point clouds. The details of these steerable factors are in Appendix C.

Semantic direction and step size. To learn the steerable factors using deep learning tools, we will introduce the semantic directions defined on the latent space of DGM. In such a space $\mathcal{Z}$, we assume there exist D semantically meaningful direction vectors, $\boldsymbol{d}_i$ with $i \in \{0, 1, \dots, D-1\}$.[1] There is also a scalar variable, step size α, which controls the degree to edit the sampled data with desired steerable factors (as will be introduced below), and we follow the prior work (Shen & Zhou, 2021) on taking $\alpha \in [-3, 3]$. Each direction corresponds to one or multiple factors, such that by editing the latent vector $\boldsymbol{z}$ along $\boldsymbol{d}_i$ with step size α, the reconstructed graph will possess the desired structural modifications. The editing with a sequence of step sizes $\alpha \in [-3, 3]$ along the same direction $\boldsymbol{d}_i$ leads to a sequence of edited graphs.

Problem formulation: graph editing or graph controllable generation. Given a *pretrained* DGM (*i.e.*, the encoder and decoder are fixed), our goal is to learn the most semantically rich directions ($\boldsymbol{d}_i$) in the latent space $\mathcal{Z}$. Then for each latent code $\boldsymbol{z}$, with the i-th semantic direction and a step size α, we can get an edited latent vector $\bar{\boldsymbol{z}}_{i,\alpha}$ and edited data $\bar{\boldsymbol{x}}'$ after decoding $\bar{\boldsymbol{z}}_{i,\alpha}$, as:

$$\boldsymbol{z} = f(\boldsymbol{x}), \qquad \bar{\boldsymbol{z}}_{i,\alpha} = h(\boldsymbol{z}, \boldsymbol{d}_i, \alpha), \qquad \bar{\boldsymbol{x}}' = g(\bar{\boldsymbol{z}}_{i,\alpha}), \tag{2}$$

where $\boldsymbol{d}_i$ and $h(\cdot)$ are the edit direction and edit functions that we want to learn. We expect that $\bar{\boldsymbol{z}}_{i,\alpha}$ can inherently possess certain steerable factors, which can be reflected in the graph structure of $\bar{\boldsymbol{x}}'$.

Energy-based model (EBM). EBM is a flexible framework for distribution modeling:

$$p(\boldsymbol{x}) = \frac{\exp(-E(\boldsymbol{x}))}{A} = \frac{\exp(-E(\boldsymbol{x}))}{\int_{\boldsymbol{x}} \exp(-E(\boldsymbol{x}))d\boldsymbol{x}}, \tag{3}$$

where $E(\cdot)$ is the energy function and A is the partition function. In EBM, the bottleneck is the estimation of partition function A. It is often intractable due to the high cardinality of $\mathcal{X}$. Various methods have been proposed to handle this issue, including but not limited to contrastive divergence (Hinton, 2002), noise-contrastive estimation (Gutmann & Hyvärinen, 2010; Che et al., 2020), and score matching (Hyvärinen & Dayan, 2005; Song & Ermon, 2019; Song et al., 2020).

3 Entanglement of Latent Representation for Graph DGMs

In this section, we quantify the degree of disentanglement of the existing DGMs for graph data. Recall that the key intuition (Locatello et al., 2019) behind disentanglement is that a disentangled representation space should separate the distinct, informative, and steerable factors of variations in the data. In other words, each latent dimension of the disentangled representation corresponds to one or multiple factors. Therefore, the change of the disentangled dimension can lead to the consistent change in the corresponding factors of the data. This good property has become a foundational assumption in many existing controllable generation methods (Shen et al., 2020b; Shen & Zhou, 2021; Härkönen et al., 2020).

In the computer vision domain, StyleGAN (Karras et al., 2019) is one of the most recent works on image generation, and several works have proven its nice disentanglement property (Collins et al., 2020; Shen et al., 2020a; Härkönen et al., 2020; Tewari et al., 2020; Wu et al., 2021). In image generation, (Locatello et al., 2019) shows that without inductive bias, the representation learned by VAEs is not perfectly disentangled. Then the next question naturally arises: *Is the latent space of pretrained graph DGMs disentangled or not?* To answer this question, we conduct the following experiment.

[1] In unsupervised editing, the steerable factors on each semantic direction is known by post-training human selection.

Table 1: The six disentanglement metrics on three pretrained DGMs and two graph types. All measures range from 0 to 1, and higher scores mean more disentangled representation.

Graph Type	DGM	Dataset	BetaVAE ↑	FactorVAE ↑	MIG ↑	DCI ↑	Modularity ↑	SAP ↑
Molecular Graph	MoFlow	ZINC250K	0.260	0.175	0.031	0.953	0.620	0.009
	HierVAE	ChEMBL	0.178	0.165	0.022	0.114	0.606	0.026
Point Cloud	PointFlow	Airplane	0.022	0.025	0.029	0.160	0.745	0.022

There have been a series of works exploring the disentanglement of the latent space in DGMs, and here we take six widely-used ones: BetaVAE (Higgins et al., 2017), FactorVAE (Kim & Mnih, 2018), MIG (Chen et al., 2018b), DCI (Eastwood & Williams, 2018), Modularity (Ridgeway & Mozer, 2018), and SAP (Kumar et al., 2018). Each measure has its own bias, and we put a detailed comparison in Appendix C. Meanwhile, they all share the same high-level idea: given the latent representation from a pretrained DGM, they are proposed to measure how predictive it is to certain steerable factors.

To adapt them to our setting, first, we need to extract the steerable factors in graph DGMs, which requires domain knowledge. For instance, in molecular graphs, we can extract some special substructures called fragments or functional groups. These substructures can be treated as steerable factors since they are the key components of the molecules and are closely related to certain molecular properties (Seybold et al., 1987). We use RDKit (Landrum et al., 2013) to extract the 11 most distinguishable fragments as steerable factors for disentanglement measurement. For point clouds, we use PCL tool (Rusu & Cousins, 2011) to extract 75 VFH descriptors (Rusu et al., 2010) as steerable factors, which depicts the geometries and viewpoints accordingly.

Then to measure the disentanglement on graph DGMs, we consider six metrics on three datasets and two data types with three backbone models. All the metric values range from 0 to 1, and the higher the value, the more disentangled the DGM is. According to Table 1, we can observe that most of the disentanglement scores are quite low, except the DCI (Eastwood & Williams, 2018) on MoFlow. Thus, we draw the conclusion that, generally, these graph DGMs are entangled. More details of this experiment (the steerable factors on two data types and six disentanglement metrics) can be found in Appendix C.

4 Our Method

The analysis in Section 3 naturally raises the next research question: *Given an entangled DGM, is there a flexible way to do the graph data editing in an unsupervised manner?* The answer is positive. We propose GraphCG, a flexible model-agnostic framework to learn the semantic directions in an unsupervised manner. It starts with the assumption that the latent representations edited with the same semantic direction and step size should possess similar information (with respect to the factors) to a certain degree, thus by maximizing the mutual information among them, we can learn the most semantic-rich directions. Then we formulate this editing task as a density estimation problem with the energy-based model (EBM). As introduced in Section 2, EBM covers a broad range of solutions, and we further propose GraphCG-NCE by adopting the noise-contrastive estimation (NCE) solution.

4.1 GraphCG with Mutual Information

Motivation: learning semantic directions using MI on entangled DGM. Recall that our ultimate goal is to enable graph editing based on semantic vectors. Existing deep generative models are entangled, thus obtaining such semantic vectors is a nontrivial task. To handle this problem, we propose using mutual information (MI) to learn the semantics. MI measures the non-linear dependency between variables. Here we set the editing condition as containing both the semantic directions and step sizes. We assume that maximizing the MI between different conditions can maximize the shared information within each condition, *i.e.*, graphs moving along the same condition share more semantic information. The pipeline is as follows.

We first sample two latent codes in the latent space, $\boldsymbol{z}^u$ and $\boldsymbol{z}^v$. Such two latent codes will be treated as positive pairs, and their construction will be introduced in Section 4.3. Then we pick up the i-th semantic direction and one step size α to obtain the edited latent codes in the latent space $\mathcal{Z}$ as:

$$\bar{\boldsymbol{z}}^u_{i,\alpha} = h(\boldsymbol{z}^u, \boldsymbol{d}_i, \alpha), \qquad \bar{\boldsymbol{z}}^v_{i,\alpha} = h(\boldsymbol{z}^v, \boldsymbol{d}_i, \alpha). \tag{4}$$

Under our assumption, we expect that these two edited latent codes share certain information with respect to the steerable factors. Thus, we want to maximize the MI between $\bar{\boldsymbol{z}}_{i,\alpha}^u$ and $\bar{\boldsymbol{z}}_{i,\alpha}^v$. Since the MI is intractable to compute, we adopt the EBM lower bound from (Liu et al., 2022) as:

$$\mathcal{L}_{\text{MI}}(\bar{\boldsymbol{z}}_{i,\alpha}^u, \bar{\boldsymbol{z}}_{i,\alpha}^v) = \frac{1}{2}\mathbb{E}\Big[\log p(\bar{\boldsymbol{z}}_{i,\alpha}^u|\bar{\boldsymbol{z}}_{i,\alpha}^v) + \log p(\bar{\boldsymbol{z}}_{i,\alpha}^v|\bar{\boldsymbol{z}}_{i,\alpha}^u)\Big]. \tag{5}$$

The detailed derivation is in Appendix D. Till this step, we have transformed the graph data editing task into the estimation of two conditional log-likelihoods.

4.2 GraphCG with Energy-Based Model

Following Equation (5), maximizing the MI between $I\big(\bar{\boldsymbol{z}}_{i,\alpha}^u; \bar{\boldsymbol{z}}_{i,\alpha}^v\big)$ is equivalent to estimating the summation of two conditional log-likelihoods. We then model them using two conditional EBMs. Because these two views are in the mirroring direction, we may as well take one for illustration. For example, for the first conditional log-likelihood, we can model it with EBM as:

$$p(\bar{\boldsymbol{z}}_{i,\alpha}^u|\bar{\boldsymbol{z}}_{i,\alpha}^v) = \frac{\exp(-E(\bar{\boldsymbol{z}}_{i,\alpha}^u, \bar{\boldsymbol{z}}_{i,\alpha}^v))}{\int \exp(-E(\bar{\boldsymbol{z}}_{i,\alpha}^{u\prime}, \bar{\boldsymbol{z}}_{i,\alpha}^v))d\bar{\boldsymbol{z}}_{i,\alpha}^{u\prime}} = \frac{\exp(f(\bar{\boldsymbol{z}}_{i,\alpha}^u, \bar{\boldsymbol{z}}_{i,\alpha}^v))}{A_{ij}}, \tag{6}$$

where $E(\cdot)$ is the energy function, A_{ij} is the intractable partition function, and $f(\cdot)$ is the negative energy. The energy function is flexible and we use the dot-product:

$$f(\bar{\boldsymbol{z}}_{i,\alpha}^u, \bar{\boldsymbol{z}}_{i,\alpha}^v) = \langle h(\boldsymbol{z}^u, \boldsymbol{d}_i, \alpha), h(\boldsymbol{z}^v, \boldsymbol{d}_i, \alpha)\rangle, \tag{7}$$

where $h(\cdot)$ is the editing function introduced in Equation (2). Similarly for the other conditional log-likelihood term, and the objective becomes:

$$\mathcal{L}_{\text{GraphCG}} = \mathbb{E}\Big[\log\frac{\exp(f(\bar{\boldsymbol{z}}_{i,\alpha}^u, \bar{\boldsymbol{z}}_{i,\alpha}^v))}{A_{ij}} + \log\frac{\exp(f(\bar{\boldsymbol{z}}_{i,\alpha}^v, \bar{\boldsymbol{z}}_{i,\alpha}^u))}{A_{ji}}\Big]. \tag{8}$$

With Equation (8), we are able to learn the semantically meaningful direction vectors. We name this unsupervised graph controllable generation framework as GraphCG. In specific, GraphCG utilizes EBM for estimation, which yields a wide family of solutions, as introduced below.

4.3 GraphCG with Noise Contrastive Estimation

We solve Equation (8) using the noise contrastive estimation (NCE) (Gutmann & Hyvärinen, 2010). The high-level idea of NCE is to transform the density estimation problem into a binary classification problem that distinguishes if the data comes from the introduced noise distribution or from the true distribution. NCE has been widely explored for solving EBM (Song & Kingma, 2021), and we adopt it as GraphCG-NCE by optimizing the following objective function:

$$\begin{aligned}\mathcal{L}_{\text{GraphCG-NCE}} = -\Big(&\mathbb{E}_{p_n(\bar{\boldsymbol{z}}_{j,\beta}^u|\bar{\boldsymbol{z}}_{i,\alpha}^v)}\big[\log\big(1-\sigma(f(\bar{\boldsymbol{z}}_{j,\beta}^u, \bar{\boldsymbol{z}}_{i,\alpha}^v))\big)\big] + \mathbb{E}_{p_{\text{data}}(\bar{\boldsymbol{z}}_{i,\alpha}^u|\bar{\boldsymbol{z}}_{i,\alpha}^v))}[\log\sigma(f(\bar{\boldsymbol{z}}_{i,\alpha}^u, \bar{\boldsymbol{z}}_{i,\alpha}^v)))\big] \\ &+ \mathbb{E}_{p_n(\bar{\boldsymbol{z}}_{j,\beta}^v|\bar{\boldsymbol{z}}_{i,\alpha}^u)}\big[\log\big(1-\sigma(f(\bar{\boldsymbol{z}}_{j,\beta}^v, \bar{\boldsymbol{z}}_{i,\alpha}^u))\big)\big] + \mathbb{E}_{p_{\text{data}}(\bar{\boldsymbol{z}}_{i,\alpha}^v|\bar{\boldsymbol{z}}_{i,\alpha}^u))}[\log\sigma(f(\bar{\boldsymbol{z}}_{i,\alpha}^v, \bar{\boldsymbol{z}}_{i,\alpha}^u)))\big]\Big),\end{aligned} \tag{9}$$

where p_{data} is the empirical data distribution and p_n is the noise distribution (derivations are in Appendix D). Recall that the latent code pairs $(\boldsymbol{z}^u, \boldsymbol{z}^v)$ are given in advance, and the noise distribution is on the semantic directions and step sizes. In specific, the step sizes $(\alpha \neq \beta)$ are randomly sampled from [-3, 3], and the latent direction indices $(i \neq j)$ are randomly sampled from {0, 1, ..., D-1}. Equation (9) is for one latent code pair, and we take the expectation of it over all the pairs sampled from the dataset. Besides, we would like to consider extra similarity and sparsity constraints as:

$$\mathcal{L}_{\text{sim}} = \mathbb{E}_{i,j}[\text{sim}(\boldsymbol{d}_i, \boldsymbol{d}_j)], \qquad \mathcal{L}_{\text{sparsity}} = \mathbb{E}_i[\|\boldsymbol{d}_i\|], \tag{10}$$

where sim$(\cdot)$ is the similarity function between two latent directions, and we use the dot product. By minimizing these two regularization terms, we can make the learned semantic directions more diverse and sparse. Putting them together, the final objective function is:

$$\mathcal{L} = c_1 \cdot \mathbb{E}_{u,v}[\mathcal{L}_{\text{GraphCG-NCE}}] + c_2 \cdot \mathcal{L}_{\text{sim}} + c_3 \cdot \mathcal{L}_{\text{sparsity}}, \tag{11}$$

where c_1, c_2, c_3 are coefficients, and we treat them as three hyperparameters (check Appendix E). The above pipeline is illustrated in Figure 1, and for the next we will discuss certain key modules.

Latent code pairs, positive and negative views. We consider two options for obtaining the latent pairs. (1) *Perturbation (GraphCG-P)* is that for each data point $\boldsymbol{x}$, we obtain its latent code $\boldsymbol{z} = f(\boldsymbol{x})$. Then we apply two perturbations (*e.g.*, adding Gaussian noise) on $\boldsymbol{z}$ to get two perturbed latent codes as $\boldsymbol{z}^u$ and $\boldsymbol{z}^v$, respectively. (2) *Random sampling (GraphCG-R)* is that we encode two randomly sampled data points from the empirical data distribution as $\boldsymbol{z}^u$ and $\boldsymbol{z}^v$ respectively. Perturbation is one of the widely-used strategies (Karras et al., 2019) for data augmentation, and random sampling has been widely used in the NCE (Song & Kingma, 2021) literature. Then we can define the positive and negative pairs in GraphCG-NCE, where the goal is to align the positives and contrast the negatives. As described in Equation (9), the positive pairs are latent pairs moving with the same semantic direction and step size, while the negative pairs are the edited latent codes with different semantic directions and/or step sizes.

Semantic direction modeling. We first randomly draw a *basis vector* $\boldsymbol{e}_i$, and then model the semantic direction $\boldsymbol{d}_i$ as $\boldsymbol{d}_i = \text{MLP}(\boldsymbol{e}_i)$, where $\text{MLP}(\cdot)$ is the multi-layer perceptron network.

Design of editing function. Given the semantic direction and two views, the next task is to design the editing function $h(\cdot)$ in Equation (2). Since our proposed GraphCG is flexible, and the editing function determines the energy function Equation (7), we consider both the linear and non-linear editing functions as:

$$\bar{\boldsymbol{z}}_i = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i, \qquad \bar{\boldsymbol{z}}_i = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i + \text{MLP}(\boldsymbol{z} \oplus \boldsymbol{d}_i \oplus [\alpha]), \tag{12}$$

where $\oplus$ is the concatenation of two vectors. Noticing that for the non-linear case, we are adding an extra term by mapping from the latent code, semantic direction, and step-size simultaneously. We expect that this could bring in more modeling expressiveness in the editing function. For more details, *e.g.*, the ablation study to check the effect on the design of the views and editing functions, please refer to Appendices F and G, while more potential explorations are left for future work.

4.4 Implementations Details

During training, the goal of GraphCG is to learn semantically meaningful direction vectors together with an editing function in the latent space, as in Algorithm 1. Then we need to manually annotate the semantic directions concerning the corresponding factors using certain post-training evaluation metrics. Finally, for the inference phase, provided with the pretrained graph DGM and a selected semantic direction (together with a step size) learned by GraphCG, we can sample a graph -> conduction editing in the latent space -> decoding to generate the edited graph, as described in Equation (2). The detailed algorithm is illustrated in Algorithm 2. Next, we highlight several key concepts in GraphCG and briefly discuss the differences from other related concepts.

Algorithm 1 Learning Phase of GraphCG

```
1: Input: Given a pretrained generative model encoder, $f(\cdot)$.
2: Output: Learned direction vector $\boldsymbol{d}_i$ and function $h(\cdot)$.
3: Select latent codes $\boldsymbol{z}^u, \boldsymbol{z}^v \in \mathcal{Z}$ from empirical dataset and $f(\cdot)$.
4: for each step size $\alpha$ and each direction $i$ do
5:     Set $\bar{\boldsymbol{z}}^u_{i,\alpha} = h(\boldsymbol{z}^u, \boldsymbol{d}_i, \alpha)$.
6:     Set $\bar{\boldsymbol{z}}^v_{i,\alpha} = h(\boldsymbol{z}^v, \boldsymbol{d}_i, \alpha)$.
7:     Assign positive to pair $(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{i,\alpha})$.
8:     for step size $\beta \neq \alpha$ and direction $j \neq i$ do
9:         Set $\bar{\boldsymbol{z}}^u_{j,\beta} = h(\boldsymbol{z}^u, \boldsymbol{d}_j, \beta)$.
10:        Set $\bar{\boldsymbol{z}}^v_{j,\beta} = h(\boldsymbol{z}^v, \boldsymbol{d}_j, \beta)$.
11:        Assign negative to pair $(\bar{\boldsymbol{z}}^u_{i,\alpha}, \bar{\boldsymbol{z}}^v_{j,\beta})$.
12:        Assign negative to pair $(\bar{\boldsymbol{z}}^u_{j,\beta}, \bar{\boldsymbol{z}}^v_{i,\alpha})$.
13:    end for
14:    Do SGD w.r.t. GraphCG in Equation (11).
15: end for
```

NCE and contrastive representation learning. GraphCG-NCE is applying EBM-NCE, which is essentially a contrastive learning method, and another dominant contrastive loss is the InfoNCE (Oord et al., 2018). We summarize their relations below. (1) Both contrastive methods are doing the same thing: align the positive pairs and contrast the negative pairs. (2) EBM-NCE (Hassani & Khasahmadi, 2020; Liu et al., 2022) has been found to outperform InfoNCE on certain graph applications like representation learning. (3) What we want to propose here is a flexible framework. Specifically, EBM provides a more general framework by designing the energy functions, and EBM-NCE is just one effective solution. Other promising directions include the denoising score matching or denoising diffusion model (Song et al., 2020), while InfoNCE lacks such a nice extensibility attribute.

GraphCG and contrastive self-supervised learning (SSL). GraphCG shares certain similarities with the self-supervised learning (SSL) method, however, there are some inherent differences, as summarized below. (1) SSL aims at learning the data representation by operating data augmentation on the data space, such as node addition and edge deletion. GraphCG aims at learning the semantically meaningful directions by editing on the latent space (the representation function is pretrained and fixed). (2) Based on the first point, SSL aims at using different data points as the negative samples. GraphCG, on the other hand, is using different directions and step-sizes as negatives. Namely, SSL is learning data representation in the inter-data level, and GraphCG is learning the semantic directions in the inter-direction level.

Algorithm 2 Inference Phase of GraphCG

```
1: Input: Given a pre-trained generative model, $f(\cdot)$ and $g(\cdot)$, a learned direction vector $\boldsymbol{d}$.
2: Output: A sequence of edited graphs.
3: Sample $\boldsymbol{z}$ with DGM or $\boldsymbol{x}$ from a graph dataset.
4: If the latter, get a latent code $\boldsymbol{z} = f(x)$.
5: for step size $\alpha \in [-3, 3]$ do
6:     Do graph edit in the latent space to get $\bar{\boldsymbol{z}}_{i,\alpha} = h(\boldsymbol{z}, \boldsymbol{d}, \alpha)$.
7:     Decode to the graph space with $\bar{\boldsymbol{x}}' = g(\bar{\boldsymbol{z}}_{i,\alpha})$.
8: end for
9: Output is thus a sequence of edited graphs, $\{\bar{\boldsymbol{x}}'\}$.
```

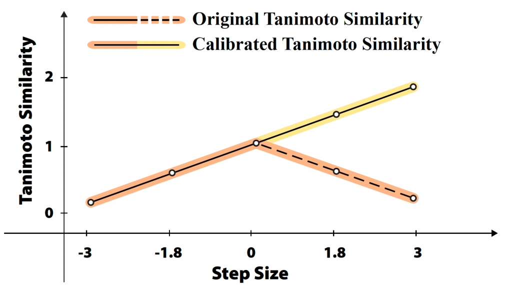
Output sequence in the discrete space. Recall that during inference time (Algorithm 2), GraphCG takes a DGM and the learned semantic direction to output a sequence of edited graphs. Compared to the vision domain, where certain models (Shen & Zhou, 2021; Shen et al., 2020b) have proven their effectiveness in many tasks, the backbone models in the graph domain have limited discussions. This is challenging because the graph data is in a discrete and structured space, and the evaluation of such space is non-trivial. Meanwhile, GraphCG essentially provides another way to verify the quality of graph DGMs. GraphCG paves the way for this potential direction, and we would like to leave this for future exploration.

5 Experiments

In this section, we show both the qualitative and quantitative results of GraphCG, on two types of graph data: molecular graphs and point clouds. Due to the page limit, We put the experiment and implementation details in Appendix E.

5.1 Graph Data: Molecular Graphs

Background of molecular graphs. A molecule can be naturally treated as a graph, where the atoms and bonds are nodes and edges, respectively. The unsupervised graph editing tasks can thus be formulated as editing the substructures of molecular graphs. In practice, people are interested in substructures that are critical components of molecules, which are called the 'fragments'. In recent years, graph representation



$$\phi_{\text{SMR}}\Big(\{s(\bar{\boldsymbol{x}}')\}_i^m, \gamma, \tau\Big) = \begin{cases} 1, & \text{len}\Big(\text{set}\big(\{s(\bar{\boldsymbol{x}}')\}_i^m\big)\Big) \geq \gamma \\ & \quad \wedge \text{ monotonic}_\tau\Big(\{s(\bar{\boldsymbol{x}}')\}_i^m\Big) \\ 0, & \text{otherwise} \end{cases}, \tag{13}$$

$$\phi_{\text{SMR}}(\gamma, \tau)_i = \frac{1}{M} \sum_{m=1}^{M} \phi_{\text{SMR}}\Big(\{s(\bar{\boldsymbol{x}}')\}_i^m, \gamma, \tau\Big), \tag{14}$$

$$\text{top-K}(\gamma, \tau) = \frac{1}{K} \sum_{i \in \text{top-K directions}} \Big(\phi_{\text{SMR}}(\gamma, \tau)_i\Big). \tag{15}$$

Figure 2: This illustrates the sequence monotonic ratio (SMR) on calibrated Tanimoto similarity (CTS). Equations (13) and (14) are the SMR on each sequence and each direction respectively, where M is the number of generated sequences for the i-th direction and $\{s(\bar{\boldsymbol{x}}')\}_i^m$ is the CTS of the m-th generated sequence with the i-th direction. Equation (15) is the average of top-K SMR on D directions.

learning has been extensively explored on the molecular graph (Duvenaud et al., 2015; Gilmer et al., 2017; Liu et al., 2018; Yang et al., 2019b; Corso et al., 2020).

Backbone DGMs. We consider two state-of-the-art DGMs for molecular graph generation. MoFlow (Zang & Wang, 2020) is a flow-based generative model on molecules that adopts an invertible mapping between the input molecular graphs and a latent prior. HierVAE (Jin et al., 2020a) is a hierarchical VAE model that encodes and decodes molecule atoms and motifs in a hierarchical manner. Besides, the pretrained checkpoints are also provided, on ZINC250K (Irwin & Shoichet, 2005) and ChEMBL (Mendez et al., 2019), respectively.

Editing sequences and anchor molecule. As discussed in Section 4, the output of the inference in GraphCG, is a sequence of edited molecules with the i-th semantic direction, $\{\bar{\boldsymbol{x}}'\}_i$. We first randomly generate a molecule using the backbone DGMs (without the editing operation), and we name such molecule as the anchor molecule, $\bar{\boldsymbol{x}}^*$. Then we take 21 step sizes from -3 to 3, with interval 0.3, to obtain a sequence of 21 molecules following Equation (2). Note that the edited molecule with step size 0 under the linear editing function is the same as the anchor molecule, *i.e.*, $\bar{\boldsymbol{x}}^*$.

Change of structure factors and evaluation metrics. We are interested in the change of the graph structure (the steerable factors) along the output sequence edited with the i-th semantic direction. To evaluate the structure change, we apply the Tanimoto similarity between each output molecule and the anchor molecule. Besides, for the ease of evaluating the monotonicity, we apply a Tanimoto similarity transformation on the output molecules with positive step sizes by taking the deduction from 2. We call this calibrated Tanimoto similarity (CTS) sequence, marked as $\{s(\bar{\boldsymbol{x}}')\}_i$. An illustration is shown in Figure 2. Further, we propose a metric called Sequence Monotonic Ratio (SMR), $\phi_{\text{SMR}}(\gamma, \tau)_i$, which measures the monotonic ratio of M generated sequences edited with the i-th direction. It has two arguments: the diversity threshold γ constrains the minimum number of distinct molecules, and the tolerance threshold τ controls the non-monotonic tolerance ratio along each sequence.

Evaluating the diversity of semantic directions. SMR can evaluate the monotonic ratio of output sequences generated by one direction. To better illustrate that GraphCG is able to learn multiple directions with diverse semantic information, we also consider taking the average of top-K SMR to reveal that all the best K directions are semantically meaningful, as in Equation (15).

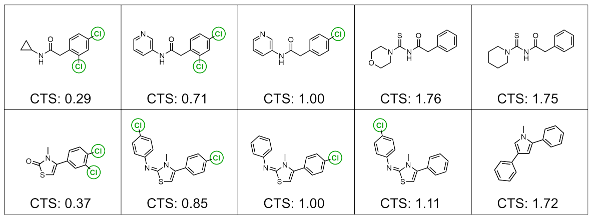
Baselines. For baselines, we consider four unsupervised editing methods. (1) The first is Random. It randomly samples a normalized random vector in the latent space as the semantic direction. (2) The second one is Variance. We analyze the variance on each dimension of the latent space, and select the highest one with one-hot encoding as the semantic direction. (3) The third one is SeFa (Shen & Zhou, 2021). It first decomposes the latent space into lower dimensions using PCA, and then takes the most principal components (eigenvectors) as the semantic-rich direction vectors. (4) The last one is DisCo (Ren et al., 2021). It maps each latent code back to the data space, followed by an encoder for contrastive learning, so it requires the backbone DGMs to be end-to-end and is infeasible for HierVAE.

Quantitative results. We take $D = 10$ to train GraphCG, and the optimal results on 100 sampled sequences are reported in Table 2. We can observe that GraphCG, can show consistently better structure change with both top-1 and top-3 directions. This can empirically prove the effectiveness of our proposed GraphCG. More comprehensive results are in Appendix F.

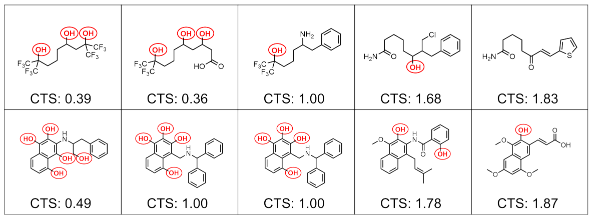
Analysis on steerable factors in molecules: functional group change. For visualization, we sample 8 molecular graph sequences along 4 selected directions in Figure 3, and the backbone DGM is HierVAE pretrained on ChEMBL. The CTS holds a good monotonic trend, and each direction shows certain unique changes in the molecular structures, *i.e.*, the steerable factors in molecules. Some structural changes are reflected in molecular properties. We expand all the details below. In Figures 3(a) and 3(b), the number of halogen atoms and hydroxyl groups (in alcohols and phenols) in the molecules decrease from left to right, respectively. In Figure 3(c), the number of amides in the molecules increases along the path. Because amides are polar functional groups, the topological polar surface area (tPSA) of the molecules also increases, which is a key molecular property for the prediction of drug transport properties, *e.g.*, permeability (Ertl et al., 2000). In Figure 3(d), the flexible chain length, marked by the number of ethylene (CH_2CH_2) units, increases

Table 2: This table lists the sequence monotonic ratio (SMR, %) on calibrated Tanimoto similarity (CTS) for the top-1 and top-3 directions. The best performances are marked in **bold**.

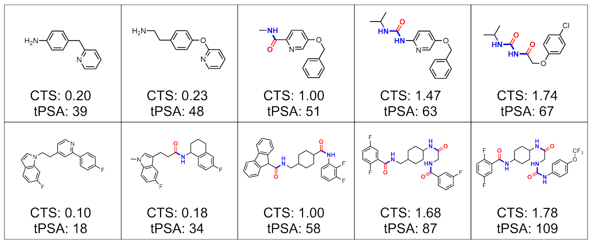
Model	**Dataset**		**Tanimoto top-1**				**Tanimoto top-3**			
		diversity γ	3		4		3		4	
		tolerance τ	0	0.2	0	0.2	0	0.2	0	0.2
MoFlow	ZINC250k	Random	23.0	25.0	12.0	15.0	22.0	24.0	11.0	13.7
		Variance	24.0	28.0	12.0	16.0	20.0	25.0	10.0	15.0
		SeFa Shen & Zhou (2021)	4.0	4.0	0.0	0.0	3.3	3.3	0.0	0.0
		DisCo Ren et al. (2021)	7.0	14.0	2.0	8.0	5.3	11.7	2.0	7.7
		GraphCG-P	**32.0**	**34.0**	**16.0**	**18.0**	**29.0**	**31.0**	**13.7**	**16.3**
		GraphCG-R	25.0	26.0	11.0	14.0	22.0	24.3	10.3	13.3
HierVAE	ChEMBL	Random	14.0	45.0	14.0	43.0	10.0	42.3	9.3	41.7
		Variance	23.0	59.0	19.0	55.0	18.3	52.7	15.7	50.3
		SeFa Shen & Zhou (2021)	4.0	41.0	4.0	41.0	2.3	36.0	2.3	36.0
		GraphCG-P	40.0	**73.0**	**32.0**	**65.0**	36.0	**64.3**	26.3	**57.7**
		GraphCG-R	**42.0**	67.0	30.0	55.0	**38.0**	62.3	**28.7**	53.7



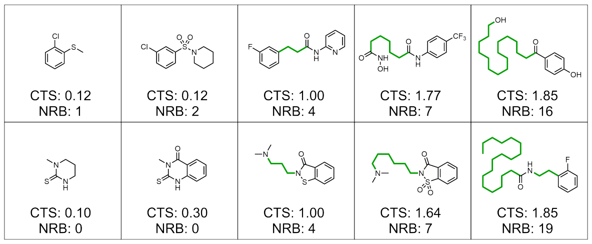
(a) Steerable factor: number of halogens.



(b) Steerable factor: number of hydroxyls.



(c) Steerable factor: number of amides.



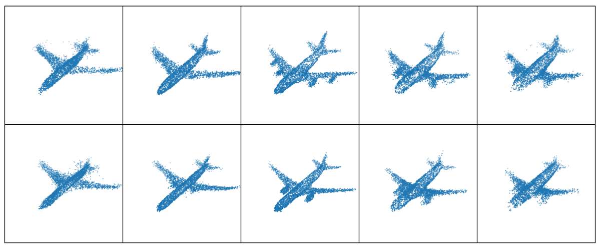
(d) Steerable factor: chain length.

Figure 3: GraphCG for molecular graph editing. We visualize the output molecules and CTS in four directions with two sequences each, where each sequence consists of five steps. The five steps correspond to five step sizes: -3, -1.8, 0, 1.8, and 3, where 0 marks the anchor molecule (center point of reach sequence). Figure 3(a) visualizes how the number of halogens (marked in green) decreses from -3 to 3. Figure 3(b) visualizes how the number of hydroxyls (marked in red) decreases from -3 to 3. Figure 3(c) visualizes how the number of amides (marked in red and blue) increases from -3 to 3. Figure 3(d) visualizes how the number of chains (marked in green) increases from -3 to 3. Notably, certain properties change with molecular structures accordingly, like topological polar surface area (tPSA) and the number of rotatable bonds (NRB).

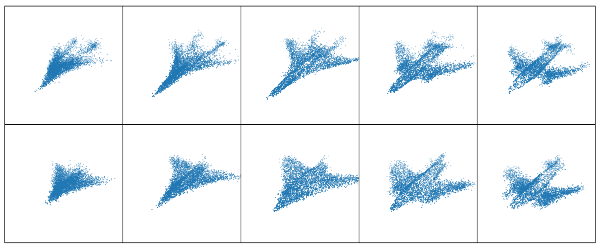
from left to right. Since the number of rotatable bonds (NRB) measures the molecular flexibility, it also increases accordingly (Veber et al., 2002).

5.2 Graph Data: Point Clouds

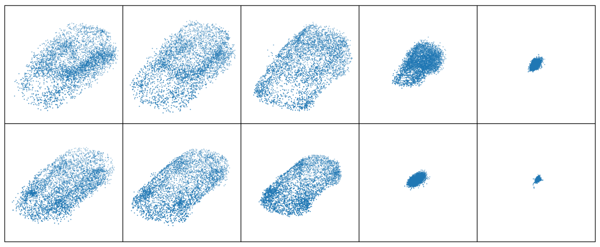
Background of point clouds. A point cloud is represented as a set of points, where each point P_i is described by a vector of 3D Euclidean coordinates possibly with extra channels (*e.g.*, colors, surface normals, and returned laser intensity). In 2017, Qi et. al (Qi et al., 2017) designed a deep learning framework called PointNet that directly consumes unordered point sets as inputs and can be used for various tasks such as classification and segmentation.



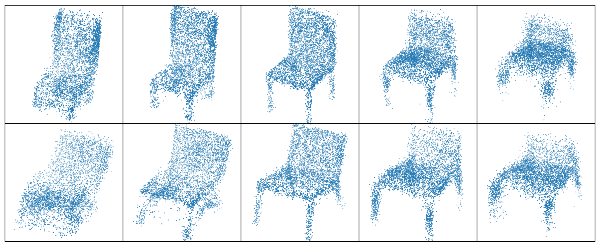
(a) Steerable factor: engine.



(b) Steerable factor: engine.



(c) Steerable factor: size.



(d) Steerable factor: leg height.

Figure 4: GraphCG for point clouds editing. We show four editing sequences, where each sequence consists of five point clouds, and the center one is the anchor point clouds, *i.e.*, with step size 0. The other four point clouds correspond to step size with -3, -1.8, 1.8, and 3, respectively. Figure 4(a) and Figure 4(b) refer the same semantic direction, and they are showing how to steer the factor engine: the number of engines will be decreased and increased with the negative (left) and positive (right) step size respectively. Similarly, Figures 4(c) and 4(d) illustrate the effect of the steerable factors on the car size and the chair leg height.

Backbone DGMs. We consider one of the latest DGMs on point clouds, PointFlow (Yang et al., 2019a). It is using the normalizing flow model for estimating the 3D point cloud distribution. Then we consider PointFlow pretrained on three datasets in ShapeNet (Chang et al., 2015): Airplane, Car, and Chair. All point clouds are obtained by sampling points uniformly from the mesh surface.

Analysis on steerable factors in point clouds: shape change. To train GraphCG, we take $D = 10$ directions, and we sample 8 point cloud sequences along 3 directions for visualization in Figure 4. More results are in Appendix G. It is observed that GraphCG, can steer the shape of the point clouds, *e.g.*, the size of cars and the height of chair legs. We also find it interesting that GraphCG, can steer more finger-trained factors, like modifying the number of jet engines.

6 Conclusion and Discussion

In this work, we are interested in unsupervised graph editing. It is a well-motivated task for various real-world applications, and we discuss two mainstream data types: molecular graphs and point clouds. We start with exploring the latent space of mainstream deep generative models and propose GraphCG, a model-agnostic unsupervised method for graph data editing. The key component of GraphCG, is EBM, and we take the GraphCG-NCE as the solution for now. For future work, we may as well extend it to more advanced solutions like denoising diffusion model (Ho et al., 2020).

One limitation of GraphCG, (as well as the solutions to general unsupervised data editing) (Härkönen et al., 2020; Shen & Zhou, 2021; Ren et al., 2021) is that we may need some post-training human selection (as shown in Algorithm 2) to select the most promising semantic vectors to steer factors. Another issue is the lack of open-sourced evaluation metrics. This requires both a deep understanding of the representation space of deep generative models and domain knowledge of the problem. For instance, activity cliff is a challenging task (Hu & Jurgen, 2012) for editing, while current measures fail to capture it. To set up constructive evaluation metrics can help augment our understandings from both the domain and technique perspectives. This is beyond the scope of our work, yet would be interesting to explore as a future direction.

Code and Data Availability

The codes and data download scripts are available at this GitHub repository.

References

Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. *arXiv:2003.05033*, 2020.

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Hans-Joachim Böhm, Alexander Flohr, and Martin Stahl. Scaffold hopping. *Drug discovery today: Technologies*, 1(3):217–224, 2004.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, et al. Shapenet: An information-rich 3d model repository. *arXiv:1512.03012*, 2015.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, et al. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *arXiv:2003.06060*, 2020.

Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. *Drug discovery today*, 23(6):1241–1250, 2018a.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *NeurIPS*, 2018b.

Edo Collins, Raja Bala, Bob Price, and Sabine Susstrunk. Editing in style: Uncovering the local semantics of gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5771–5780, 2020.

John Comer and Kin Tam. Lipophilicity profiles: theory and measurement. *Pharmacokinetic optimization in drug research: biological, physicochemical, and computational strategies*, pp. 275–304, 2001.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *arXiv:2004.05718*, 2020.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv:1410.8516*, 2014.

Jürgen Drews. Drug discovery: a historical perspective. *Science*, 287(5460):1960–1964, 2000.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, et al. Convolutional networks on graphs for learning molecular fingerprints. *arXiv:1509.09292*, 2015.

Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *ICLR*, 2018.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):1–11, 2009.

Peter Ertl, Bernhard Rohde, and Paul Selzer. Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties. *Journal of medicinal chemistry*, 43(20):3714–3717, 2000.

Peter Ertl, Eva Altmann, and Jeffrey M McKenna. The most common functional groups in bioactive molecules and how their popularity has evolved over time. *Journal of medicinal chemistry*, 63(15):8408–8418, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Laurent Gomez. Decision making in medicinal chemistry: The power of our intuition. *ACS Medicinal Chemistry Letters*, 9(10):956–958, 2018.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. Vision gnn: An image is worth graph of nodes. *arXiv preprint arXiv:2206.00272*, 2022.

Erik Härkönen, Aaron Hertzman, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *NeurIPS*, 2020.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *ICML*, 2020.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, et al. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Ye Hu and Bajorath Jurgen. Extending the activity cliff concept: structural categorization of activity cliffs and systematic identification of different types of cliffs in the chembl database. *Journal of chemical information and modeling*, 52(7):1806–1811, 2012.

Ye Hu, Dagmar Stumpfe, and Jurgen Bajorath. Recent advances in scaffold hopping: miniperspective. *Journal of medicinal chemistry*, 60(4):1238–1246, 2017.

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

John J Irwin and Brian K Shoichet. Zinc: a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182, 2005.

Ali Jahanian, Lucy Chai, and Phillip Isola. On the" steerability" of generative adversarial networks. In *ICLR*, 2019.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *ICML*, 2020a.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *ICML*, 2020b.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *ICML*, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. In *ICLR*, 2018.

Greg Landrum et al. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling, 2013.

Shengchao Liu, Mehmet Furkan Demirel, and Yingyu Liang. N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. *arXiv:1806.09206*, 2018.

Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, et al. Pre-training molecular graph representation with 3d geometry. In *ICLR*, 2022.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, et al. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*, 2019.

David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, et al. Chembl: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1):D930–D940, 2019.

Zlatko Mihalić and Nenad Trinajstić. A graph-theoretical approach to structure-property relationships. *Journal of Chemical Education*, 69(9):701, 1992.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

Weili Nie, Arash Vahdat, and Anima Anandkumar. Controllable and compositional generation with latent-space energy-based models. In *NeurIPS*, 2021.

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

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

Yael Pritch, Eitam Kav-Venaki, and Shmuel Peleg. Shift-map image editing. In *ICCV*, 2009.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

Xuanchi Ren, Tao Yang, Yuwang Wang, and Wenjun Zeng. Do generative models know disentanglement? contrastive learning is all you need. *arXiv:2102.10543*, 2021.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

Karl Ridgeway and Michael C Mozer. Learning deep disentangled embeddings with the f-statistic loss. In *NeurIPS*, 2018.

Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *ICRA*, 2011.

Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *IROS*, 2010.

Paul G. Seybold, Michael May, and Ujjvala A. Bagal. Molecular structure: Property relationships. *Journal of Chemical Education*, 64(7):575, 1987.

Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *CVPR*, 2021.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9243–9252, 2020a.

Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *IEEE TPAMI*, 2020b.

Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *CVPR*, 2020.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv:1907.05600*, 2019.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv:2101.03288*, 2021.

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

Jiangming Sun, Nina Jeliazkova, Vladimir Chupakhin, Jose-Felipe Golib-Dzib, Ola Engkvist, et al. Excape-db: an integrated large scale dataset facilitating big data analysis in chemogenomics. *Journal of cheminformatics*, 9(1):1–9, 2017.

Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zollhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6142–6151, 2020.

Daniel F Veber, Stephen R Johnson, Hung-Yuan Cheng, Brian R Smith, Keith W Ward, and Kenneth D Kopple. Molecular properties that influence the oral bioavailability of drug candidates. *Journal of medicinal chemistry*, 45(12):2615–2623, 2002.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, et al. Dynamic graph cnn for learning on point clouds. *ACM ToG*, 2020.

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

Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *CVPR*, 2021.

Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, et al. Gan inversion: A survey. *arXiv:2101.05278*, 2021.

Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, et al. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, 2019a.

Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019b.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *KDD*, 2020.

Reproducibility Statement

To ensure the reproducibility of the empirical results, we provide the implementation details (hyperparameters, dataset specifications, pretrained checkpoints, etc.) in Section 5 and Appendices C and E to G, and the source code will be released in the future. Besides, the complete derivations of equations and clear explanations are given in Section 4 and Appendix D.

Specifically, we provide the details for reproducing the results:

- In Table 5, GraphCG-P with Equation (26) and GraphCG-R with Equation (24) are reported in Table 2.
- In Table 6, GraphCG-P with Equation (25) and GraphCG-R with Equation (25) are reported in Table 2.

For the visualization in Figure 3, we take the GraphCG-P with Equation (25), and the backbone generative model is HierVAE pretrained on ChEMBL. Further, we provide an anonymous link here. In these CSV files:

- Direction 0 is the halogen fragment (data 4, 71).
- Direction 5 is the amide fragment (data 95, 61).
- Direction 6 is the chain length (data 57, 14).
- Direction 7 is the alcohol and phenol fragments (data 10, 8).

For the visualization in Figure 4, we take the GraphCG-R with Equation (25) on PointFlow, for all three datasets.

Scope of GraphCG: Why Not Image

In the current main story, we do not include GraphCG for image editing. We would like to highlight that image editing is indeed a different story in terms of editing with pretrained DGMs, and the details reasons are as follows:

- We want to clarify that graph is a very general data structure, including the molecular graph and point clouds.
- In this sense, the image is a very special case of graph/structured data. It has a very tight spatial correlation, which is in a rigid form. Feel free to check (Han et al., 2022; Corso et al., 2020) for recent explorations on GNNs on images. Such a nice spatial correlation enables disentangled generative model. For example, in vision, StyleGAN (Karras et al., 2019) has proven with nice disentanglement property (Collins et al., 2020; Shen et al., 2020a; Härkönen et al., 2020; Tewari et al., 2020; Wu et al., 2021).
- But possessing such a nice generative model is not the case for general graph data. I.e., we typically don't have such a nice disentangled generative model, as discussed in Sections 1 and 3.
  - If the pretrained graph DGMs have the nice disentanglement property, like StyleGAN, then the unsupervised steerable exploration can be quite straightforward, like SeFa. SeFa is the SOTA unsupervised editing method on the image, and it only needs a simple PCA operation.
  - But, in reality, and in the general setting, the generative models fail to provide nice properties like disentanglement (for the general graph data). Then we need GraphCG. It is a general framework for less ordered graph data (compared to images).
  - Further, when we compare SeFa with GraphCG, we can observe that GraphCG is better by a large margin. This also verifies that the general graph controllable generation is more challenging.
- So to get back to this question, GraphCG is indeed a general graph-specific method, aiming at exploring the steerable factors in the entangled generative model setting. And obviously, the image generation task is not included in the current draft because the SOTA image generative model is good enough (well-disentangled (Collins et al., 2020; Shen et al., 2020a; Härkönen et al., 2020; Tewari et al., 2020; Wu et al., 2021)) with simple yet efficient methods like SeFa.

A Graph Data

This section will be merged with Section 5 in the final version.

A.1 Molecules

Molecules can be naturally represented as the 2D molecular graphs, where atoms and bonds are nodes and edges in the graph, respectively. For the recent years, graph representation learning has been extensively explored on the molecular graph (Duvenaud et al., 2015; Gilmer et al., 2017; Liu et al., 2018; Yang et al., 2019b; Corso et al., 2020). Based on the graph representation of molecules, a variety of work have been done for molecule generation. The state-of-the-art ones include MoFlow (Zang & Wang, 2020) and HierVAE (Jin et al., 2020a).

A.2 Point Clouds

A point cloud is represented as a set of points, where each point P_i is described by a vector of 3D Euclidean coordinates possibly with extra channels (e.g., colors, surface normals and returned laser intensity). In 2017, Qi et. al (Qi et al., 2017) designed a deep learning framework called PointNet that directly consumes unordered point sets as inputs and can be used for various tasks such as classification and segmentation. For the generative models on point clouds, we consider one of the latest work, PointFlow (Yang et al., 2019a).

B Related Work

Image editing and image controllable generation Many existing works on controllable generation with DGMs mainly focus on image data. With the assumption that the learned latent space already includes rich semantic information (Karras et al., 2019; Jahanian et al., 2019; Shen et al., 2020b; Härkönen et al., 2020), the question then becomes how to identify semantic-rich directions from the latent space of DGMs. Depending on whether or using supervised signals to discover the semantic directions, existing works can be divided into two settings: *supervised* and *unsupervised*.

The *supervised* setting relies on the supervised signals to learn the pre-defined semantic-rich directions. For instance, InterfaceGAN (Shen et al., 2020b) identifies the semantic directions in the latent space via linear models that recognize semantic boundaries among data. LACE (Nie et al., 2021) uses energy-based models to learn the joint distribution of data and properties (*i.e.*, semantics) and formulate the sampling process as to solve an ordinary differential equation.

As supervised signals usually require domain knowledge and laborious annotations, the latest works are more focused on the *unsupervised* setting, either model-specific or data-specific. Specifically, GANSpace (Härkönen et al., 2020) applies PCA on the intermediate layers of GANs (instead of latent space) for learning the semantic-rich directions. SeFa (Shen & Zhou, 2021) exploits the pretrained GANs and extracts the semantic-rich directions by using PCA on the backbone layers. Nevertheless, as both methods are specifically designed for StyleGAN (Karras et al., 2019), it is nontrivial to generalize them to other DGMs. A more recent work DisCo (Ren et al., 2021) employs a different pipeline: it trains a new encoder after reconstruction and maps the generated images to another latent space for editing. However, training a new encoder introduces extra complexities.

Graph editing and graph controllable generation This is an emerging field with many downstream applications (Drews, 2000; Shi & Rajkumar, 2020). However, existing works are mainly focusing on the supervised setting. For example, conventional approaches, such as genetic algorithms (GAs), edit the molecule graphs in the data space via hand-crafted heuristics with the guidance of molecular property predictors. More recent learning-based methods perform the latent direction discovery, either by training a classifier on the latent space of DGMs on the graph data (Jin et al., 2020a) or by learning to retrieve exemplar samples from a retrieval database for guidance (Wang et al., 2022). Other works fine-tune a pre-trained graph DGM using the supervised property annotations as rewards, resulting in a controllable DGM specifically for the considered task (Olivecrona et al., 2017; You et al., 2018). To the best of our knowledge, our work is the first to explore unsupervised graph editing in an unsupervised manner. Besides, different from many previous approaches that may only work for molecule graphs or point cloud graphs, our method is generic and thus can be applied to various graph modalities.

C Analysis Experiments on Disentanglement

In Section 3, we conduct three analysis experiments to conclude that the representation space is not perfectly disentangled in such setting. In this section, we provide more details and complementary information of these experiments.

Table 3: The six mainstream disentanglement metrics on five DGMs and three data types. All measures range from 0 to 1, and higher scores mean more disentangled representation. MoFlow and HierVAE are for molecular graphs, PointFlow is for point clouds.

Data Type	Model	Dataset	BetaVAE ↑	FactorVAE ↑	MIG ↑	DCI ↑	Modularity ↑	SAP ↑
Molecular Graphs	MolFlow	QM9	0.251	0.165	0.038	0.727	0.599	0.017
		ZINC250k	0.264	0.175	0.030	0.958	0.620	0.009
	HierVAE	QM9	0.165	0.135	0.044	0.241	0.626	0.076
		ChEMBL	0.159	0.130	0.023	0.113	0.604	0.026
Point Clouds	PointFlow	Airplane	0.022	0.025	0.029	0.160	0.745	0.022
		Chair	0.019	0.014	0.032	0.149	0.721	0.019
		Car	0.019	0.023	0.021	0.120	0.713	0.021

C.1 Steerable Factors

As mentioned in Section 3, we consider measuring the disentanglement with respect to three types of structured data: molecular graphs and point clouds. Recall that we need to define steerable factors first, so as to measure the disentanglement.

Molecular Graph For molecular graphs, we treat the important substructures (a.k.a., motifs or fragments) as factors, and they are extracted using RDKit. To calculate the disentanglement for molecules, we randomly sample 10k molecules on QM9-MolFlow, ZINC250K-MolFlow, QM9-HierVAE, and ChEMBL-HierVAE, respectively. Most of the fragments do not show up or with very few times (less than 1% occurrence frequency). Removing these fragments leads to the following 11 motifs, and we consider the existence of them as binary labels:

- aliphatic hydroxyl groups.
- aliphatic hydroxyl groups excluding tert-OH.
- aromatic nitrogens.
- aromatic amines.
- carbonyl O.
- carbonyl O, excluding COOH.
- Tertiary amines.
- Secondary amines.
- amides.
- ether oxygens (including phenoxy).
- nitriles.

Point Clouds For point clouds, we adopt the viewpoint feature histogram (VFH) (Rusu et al., 2010) implemented in PCL (Rusu & Cousins, 2011). There are 308 bins in total, where each bin is a histogram of the angles that viewpoint direction makes with each normal. VFH has been widely used as point cloud descriptors, and here we binarize it into factors with:

- We collect the shared non-zero bins from all three datasets (Airplane, Car, and Chair), and ignore the bins where the values distribution are highly skewed. This can give us 75 bins.
- Then for each of these selected bins, we use the median value as the threshold to generate the binary factor labels.

C.2 Disentanglement Measures

We follow the (Locatello et al., 2019) on considering the following six disentanglement measures:

- β-VAE (Higgins et al., 2017) evaluates the prediction accuracy of a linear classifier for the index of a fixed factor of variation.
- FactorVAE (Kim & Mnih, 2018) addresses the limitations (i.e. corner case) of β-VAE via introducing a majority voting classifier on a different feature vector.
- MIG (Chen et al., 2018b) measures the normalized difference between the highest and second highest mutual information between latent dimensions and each steerable factor.
- DCI (Eastwood & Williams, 2018) disentanglement score measures the average difference from one of the entropy of probability that a latent dimension is useful for predicting a steerable factor (computed by the relative importance score).
- Modularity (Ridgeway & Mozer, 2018) measures whether each latent dimension is dependent on at most one single steerable factor. It computes the average normalized squared difference over the highest and second-highest mutual information between each steerable factor and each latent dimension.
- SAP (Kumar et al., 2018) calculates the R^2 score of the linear models trained to predict each steerable factor from each latent dimension.

Recall that all six disentanglement measures range from 0 to 1, and a higher value means the corresponding space is more disentangled. The results can be found in Table 3. We can conclude that all the values are indeed low on all datasets and models, revealing that DGMs are entangled in general.

D Mutual Information

In this section, we will briefly introduce mutual information (MI), and also a lower bound for maximizing MI. This has been previously proposed in (Liu et al., 2022) for self-supervised learning, and the comprehensive derivations are as follows. First for notation, without loss of generality, we use X and Y as the two views.

D.1 Formulation

The standard formulation for mutual information (MI) is

$$I(X;Y) = \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\big[\log \frac{p(\boldsymbol{x},\boldsymbol{y})}{p(\boldsymbol{x})p(\boldsymbol{y})}\big]. \tag{16}$$

Mutual information (MI) between random variables measures the corresponding non-linear dependence. As can be seen in the first equation in Equation (16), the larger the divergence between the joint ($p(\boldsymbol{x},\boldsymbol{y})$ and the product of the marginals $p(\boldsymbol{x})p(\boldsymbol{y})$, the stronger the dependence between X and Y.

D.2 A Lower Bound to MI

First we can get a lower bound of MI. Assuming that there exist (possibly negative) constants a and b such that $a \le H(X)$ and $b \le H(Y)$, *i.e.*, the lower bounds to the (differential) entropies, then we have:

$$\begin{aligned} I(X;Y) &= \frac{1}{2}\big(H(X) + H(Y) - H(Y|X) - H(X|Y)\big) \\ &\ge \frac{1}{2}\big(a + b - H(Y|X) - H(X|Y)\big) \\ &\ge \frac{1}{2}\big(a + b\big) + \mathcal{L}_{\text{MI}}. \end{aligned} \tag{17}$$

Thus, we transform the MI maximization problem into maximizing the following objective:

$$\mathcal{L}_{\text{MI}} = \frac{1}{2}\mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\log p(\boldsymbol{x}|\boldsymbol{y})\Big] + \frac{1}{2}\mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\Big[\log p(\boldsymbol{y}|\boldsymbol{x})\Big]. \tag{18}$$

Empirically, we use energy-based models to model the distributions. The existence of a and b can be understood as the requirements that the two distributions $(p_{\boldsymbol{x}}, p_{\boldsymbol{y}})$ are not collapsed. For the next, we will try to model the two conditional data distributions p with model distributions p_θ.

D.3 Derivation of conditional EBM with NCE

WLOG, let's consider modeling the $p_\theta(\boldsymbol{x}|\boldsymbol{y})$ first, and by EBM it is as follows:

$$p_\theta(\boldsymbol{x}|\boldsymbol{y}) = \frac{\exp(-E(\boldsymbol{x}|\boldsymbol{y}))}{\int \exp(-E(\tilde{\boldsymbol{x}}|\boldsymbol{y}))d\tilde{\boldsymbol{x}}} = \frac{\exp(f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y}))}{\int \exp(f_{\boldsymbol{x}}(\tilde{\boldsymbol{x}}|\boldsymbol{y}))d\tilde{\boldsymbol{x}}} = \frac{\exp(f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y}))}{A_{\boldsymbol{x}|\boldsymbol{y}}}. \tag{19}$$

Then we solve this using NCE. NCE handles the intractability issue by transforming it as a binary classification task. We take the partition function $A_{\boldsymbol{x}|\boldsymbol{y}}$ as a parameter, and introduce a noise distribution p_n. Based on this, we introduce a mixture model: $\boldsymbol{z}=0$ if the conditional $\boldsymbol{x}|\boldsymbol{y}$ is from $p_n(\boldsymbol{x}|\boldsymbol{y})$, and $\boldsymbol{z}=1$ if $\boldsymbol{x}|\boldsymbol{y}$ is from $p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})$. So the joint distribution is:

$$p_{n,\text{data}}(\boldsymbol{x}|\boldsymbol{y}) = p(\boldsymbol{z}=1)p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y}) + p(\boldsymbol{z}=0)p_n(\boldsymbol{x}|\boldsymbol{y})$$

The posterior of $p(\boldsymbol{z}=0|\boldsymbol{x},\boldsymbol{y})$ is

$$p_{n,\text{data}}(\boldsymbol{z}=0|\boldsymbol{x},\boldsymbol{y}) = \frac{p(\boldsymbol{z}=0)p_n(\boldsymbol{x}|\boldsymbol{y})}{p(z=0)p_n(\boldsymbol{x}|\boldsymbol{y}) + p(\boldsymbol{z}=1)p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})} = \frac{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y})}{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y}) + p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})},$$

where $\nu = \frac{p(\boldsymbol{z}=0)}{p(\boldsymbol{z}=1)}$.

Similarly, we can have the joint distribution under EBM framework as:

$$p_{n,\theta}(\boldsymbol{x}) = p(z=0)p_n(\boldsymbol{x}|\boldsymbol{y}) + p(z=1)p_\theta(\boldsymbol{x}|\boldsymbol{y})$$

And the corresponding posterior is:

$$p_{n,\theta}(\boldsymbol{z}=0|\boldsymbol{x},\boldsymbol{y}) = \frac{p(\boldsymbol{z}=0)p_n(\boldsymbol{x}|\boldsymbol{y})}{p(\boldsymbol{z}=0)p_n(\boldsymbol{x}|\boldsymbol{y}) + p(\boldsymbol{z}=1)p_\theta(\boldsymbol{x}|\boldsymbol{y})} = \frac{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y})}{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y}) + p_\theta(\boldsymbol{x}|\boldsymbol{y})}$$

We indirectly match $p_\theta(\boldsymbol{x}|\boldsymbol{y})$ to $p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})$ by fitting $p_{n,\theta}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$ to $p_{n,\text{data}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$ by minimizing their KL-divergence:

$$\begin{aligned}
&\min_\theta D_{\text{KL}}(p_{n,\text{data}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})||p_{n,\theta}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}))\\
&= \mathbb{E}_{p_{n,\text{data}}(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{y})}[\log p_{n,\theta}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})]\\
&= \int \sum_{\boldsymbol{z}} p_{n,\text{data}}(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{y}) \cdot \log p_{n,\theta}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{x}\\
&= \int \Big\{ p(\boldsymbol{z}=0)p_{n,\text{data}}(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{z}=0)\log p_{n,\theta}(\boldsymbol{z}=0|\boldsymbol{x},\boldsymbol{y})\\
&\qquad + p(\boldsymbol{z}=1)p_{n,\text{data}}(\boldsymbol{x}|\boldsymbol{z}=1,\boldsymbol{y})\log p_{n,\theta}(\boldsymbol{z}=1|\boldsymbol{x},\boldsymbol{y})\Big\} d\boldsymbol{x}\\
&= \nu \cdot \mathbb{E}_{p_n(\boldsymbol{x}|\boldsymbol{y})}\Big[\log p_{n,\theta}(\boldsymbol{z}=0|\boldsymbol{x},\boldsymbol{y})\Big] + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})}\Big[\log p_{n,\theta}(\boldsymbol{z}=1|\boldsymbol{x},\boldsymbol{y})\Big]\\
&= \nu \cdot \mathbb{E}_{p_n(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y})}{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y}) + p_\theta(\boldsymbol{x}|\boldsymbol{y})}\Big] + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{y})}{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y}) + p_\theta(\boldsymbol{x}|\boldsymbol{y})}\Big].
\end{aligned} \tag{20}$$

This optimal distribution is an estimation to the actual distribution (or data distribution), *i.e.*, $p_\theta(\boldsymbol{x}|\boldsymbol{y}) \approx p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})$. We can follow the similar steps for $p_\theta(\boldsymbol{y}|\boldsymbol{x}) \approx p_{\text{data}}(\boldsymbol{y}|\boldsymbol{x})$. Thus following Equation (20), the objective function is to maximize

$$\nu \cdot \mathbb{E}_{p_{\text{data}}(\boldsymbol{y})}\mathbb{E}_{p_n(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y})}{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y}) + p_\theta(\boldsymbol{x}|\boldsymbol{y})}\Big] + \mathbb{E}_{p_{\text{data}}(\boldsymbol{y})}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{y})}{\nu \cdot p_n(\boldsymbol{x}|\boldsymbol{y}) + p_\theta(\boldsymbol{x}|\boldsymbol{y})}\Big]. \tag{21}$$

The we will adopt three strategies to approximate Equation (21):

1. **Self-normalization.** When the EBM is very expressive, *i.e.*, using deep neural network for modeling, we can assume it is able to approximate the normalized density directly (Mnih & Teh, 2012; Song & Kingma, 2021). In other words, we can set the partition function $A = 1$. This is a self-normalized EBM-NCE, with normalizing constant close to 1, *i.e.*, $p(\boldsymbol{x}) = \exp(-E(\boldsymbol{x})) = \exp(f(\boldsymbol{x}))$.
2. **Exponential tilting term.** Exponential tilting term (Arbel et al., 2020) is another useful trick. It models the distribution as $\tilde{p}_\theta(\boldsymbol{x}) = q(\boldsymbol{x}) \exp(-E_\theta(\boldsymbol{x}))$, where $q(\boldsymbol{x})$ is the reference distribution. If we use the same reference distribution as the noise distribution, the tilted probability is $\tilde{p}_\theta(\boldsymbol{x}) = p_n(\boldsymbol{x}) \exp(-E_\theta(\boldsymbol{x}))$.
3. **Sampling.** For many cases (Liu et al., 2022), we only need to sample 1 negative points for each data, *i.e.*, $\nu = 1$.

Following these three disciplines, the objective function to optimize $p_\theta(\boldsymbol{x}|\boldsymbol{y})$ becomes

$$
\begin{aligned}
&\mathbb{E}_{p_n(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{p_n(\boldsymbol{x}|\boldsymbol{y})}{p_n(\boldsymbol{x}|\boldsymbol{y}) + \tilde{p}_\theta(\boldsymbol{x}|\boldsymbol{y})}\Big] + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{\tilde{p}_\theta(\boldsymbol{x}|\boldsymbol{y})}{p_n(\boldsymbol{x}|\boldsymbol{y}) + \tilde{p}_\theta(\boldsymbol{x}|\boldsymbol{y})}\Big] \\
=&\mathbb{E}_{p_n(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{1}{1 + p_\theta(\boldsymbol{x}|\boldsymbol{y})}\Big] + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{y})}{1 + p_\theta(\boldsymbol{x}|\boldsymbol{y})}\Big] \\
=&\mathbb{E}_{p_n(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{\exp(-f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y}))}{\exp(-f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y})) + 1}\Big] + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \frac{1}{\exp(-f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y})) + 1}\Big] \\
=&\mathbb{E}_{p_n(\boldsymbol{x}|\boldsymbol{y})}\Big[\log\big(1 - \sigma(f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y}))\big)\Big] + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})}\Big[\log \sigma(f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y}))\Big].
\end{aligned}
\tag{22}
$$

Thus, the final EBM-NCE contrastive SSL objective is

$$
\begin{aligned}
\mathcal{L}_{\text{EBM-NCE}} = &-\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\boldsymbol{y})}\Big[\mathbb{E}_{p_n(\boldsymbol{x}|\boldsymbol{y})} \log\big(1 - \sigma(f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y}))\big) + \mathbb{E}_{p_{\text{data}}(\boldsymbol{x}|\boldsymbol{y})} \log \sigma(f_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{y}))\Big] \\
&- \frac{1}{2}\mathbb{E}_{p_{\text{data}}(\boldsymbol{x})}\Big[\mathbb{E}_{p_n(\boldsymbol{y}|\boldsymbol{x})} \log\big(1 - \sigma(f_{\boldsymbol{y}}(\boldsymbol{y},\boldsymbol{x}))\big) + \mathbb{E}_{p_{\text{data}}(\boldsymbol{y},\boldsymbol{x})} \log \sigma(f_{\boldsymbol{y}}(\boldsymbol{y},\boldsymbol{x}))\Big].
\end{aligned}
\tag{23}
$$

E Implementation and Experiment Details

Editing function. For the editing function, we consider both linear (Equations (24) and (25)) and non-linear (Equation (26)) cases as below, *i.e.*, for $\bar{\boldsymbol{z}}_{i,\alpha} = h(\boldsymbol{z}, \boldsymbol{d}_i, \alpha)$:

$$\bar{\boldsymbol{z}}_{i,\alpha} = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i, \quad \boldsymbol{d}_i = \text{norm} \circ \text{Linear}(\boldsymbol{e}_i), \tag{24}$$

$$\bar{\boldsymbol{z}}_{i,\alpha} = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i, \quad \boldsymbol{d}_i = \text{sqrt} \circ \text{norm} \circ \text{ReLU} \circ \text{Linear}(\boldsymbol{e}_i), \tag{25}$$

$$\bar{\boldsymbol{z}}_{i,\alpha} = \boldsymbol{z} + \alpha \cdot \boldsymbol{d}_i + \text{norm} \circ \text{Linear} \circ \text{ReLU} \circ \text{Linear}(\boldsymbol{z} \oplus \boldsymbol{d}_i \oplus [\alpha]), \quad \boldsymbol{d}_i = \text{norm} \circ \text{Linear} \circ \text{ReLU} \circ \text{Linear}(\boldsymbol{e}_i), \tag{26}$$

where $\circ$ means the composition of two functions.

Objective function. The objective function is given as:

$$\mathcal{L} = c_1 \cdot \mathbb{E}_{u,v}[\mathcal{L}_{\text{GraphCG-NCE}}] + c_2 \cdot \mathcal{L}_{\text{sim}} + c_3 \cdot \mathcal{L}_{\text{sparsity}}, \tag{27}$$

where $\mathcal{L}_{\text{GraphCG-NCE}}$ is the MI estimation defined in Equation (9), $\mathcal{L}_{\text{sim}}$ is the direction similarity defined in Equation (10), and $\mathcal{L}_{\text{sparsity}}$. c_1, c_2 and c_3 are three coefficients accordingly.

Hyperparameters. We list the key hyperparameters in Table 4, and all the results are evaluated on 100 sampled sequences. We also want to highlight that DisCo is an unstable baseline, in the sense that once we add more training data (*e.g.*, from 100 to 500) or more training epochs (*e.g.*, from 1 epoch to 5 epochs), the model will collapse, with the nan loss. Thus, here we are reporting the most reasonable results for DisCo, *i.e.*, 100 training data with 1 epoch.

Table 4: Hyperparameter specifications.

	Hyperparameter	Value
Random	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
Variance	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
	# training data	{100, 500}
SeFa	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
	# training data	{100, 500}
DisCo	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
	# training data	{100}
	epochs	{1}
GraphCG	D	{10}
	α	{-3, -2.7, -2.4, ..., 2.7, 3}
	# training data	{100, 500}
	epochs	{20, 100}
	coefficient c_1	{1, 2}
	coefficient c_2	{0, 1}
	coefficient c_3	{0, 1}

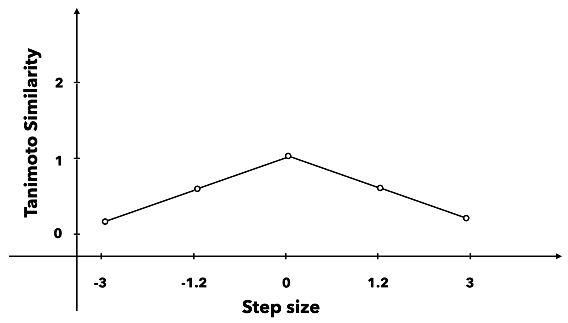
Hardware. We use V100 GPU cards, and each job (w.r.t. different hyperparameters) for GraphCG can be finished within 3 hours on a single GPU card.

Time complexity. The time complexity of GraphCG is $O(B \times D^2)$ for GraphCG-P and $O(B^2 \times D^2)$ for GraphCG-R, where B is the number of data points for each batch. Here we omit the constants for step-sizes.

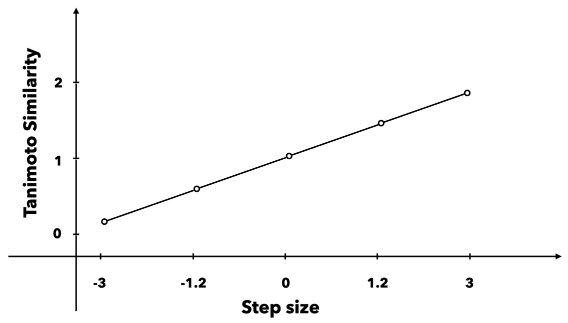
F Results: Molecular Graph

F.1 Evaluation Metrics

Change of Structure Factors and Calibrated Tanimoto Similarity (CTS). We are interested in the change of the graph structure (the steerable factors) along the output sequence edited with the i-th direction. To evaluate the structure change, we apply the Tanimoto similarity between each output molecule and the anchor molecule, as shown in Figure 5(a). Besides, for the ease of evaluating the monotonicity, we utilize a transformation (on the Tanimoto similarity) of output molecules with positive step size by taking the deduction from 2. We call this calibrated Tanimoto similarity sequence (CTS), *i.e.*, $\{s(\bar{\boldsymbol{x}}')\}_i$, as shown in Figure 5(b).



(a) Tanimoto Similarity Sequence



(b) Calibrated Tanimoto Similarity Sequence

Figure 5: Figure 5(a) is the original Tanimoto similarity sequence w.r.t. the anchor molecule, *i.e.*, step size with 0 in the figure. Yet, this is not easy to compute the monotonicity. We thus propose the calibrated Tanimoto similarity sequence, by taking the deduction from 2 for output molecules with positive step size, as shown in Figure 5(b).

Sequence Monotonic Ratio (SMR). For evaluation, we propose a metric called Sequence Monotonic Ratio (SMR), $\phi_{\text{SMR}}(\gamma, \tau)_i$. It measures the monotonic ratio of M generated sequences edited with the i-th direction. It has two arguments: the diversity threshold γ constrains the minimum number of distinct molecules, and the tolerance threshold τ controls the non-monotonic tolerance ratio along each sequence.

In specific, for each learned semantic direction i, we will generate M sequences of edited molecules, and the calibrated Tanimoto similarity for each sequence is marked as $\{s(\bar{\boldsymbol{x}}')\}_i^m$. Then we can define the SMR on each direction as:

$$\begin{aligned} \phi_{\text{SMR}}(\gamma, \tau)_i &= \frac{1}{M} \sum_{m=1}^{M} \phi_{\text{SMR}}\Big(\{s(\bar{\boldsymbol{x}}')\}_i^m, \gamma, \tau\Big), \\ \phi_{\text{SMR}}\Big(\{s(\bar{\boldsymbol{x}}')\}_i^m, \gamma, \tau\Big) &= \begin{cases} 1, & \text{len}\Big(\text{set}\Big\{s(\bar{\boldsymbol{x}}')\}_i^m\Big)\Big) \geq \gamma \wedge \text{ monotonic}_\tau\left(\{s(\bar{\boldsymbol{x}}')\}_i^m\right) \\ 0, & \text{otherwise} \end{cases}. \end{aligned} \tag{28}$$

Evaluating the Diversity of Semantic Directions. SMR can evaluate all the output sequences generated by one direction. To better illustrate that GraphCG is able to learn multiple directions with various semantic information, we also consider taking the average of top-K SMR w.r.t. directions to reveal that all the best K directions are semantically meaningful, as in Equation (29):

$$\text{top-K}(\gamma, \tau) = \frac{1}{K} \sum_{i \in \text{top-K directions}} \Big(\phi_{\text{SMR}}(\gamma, \tau)_i\Big). \tag{29}$$

F.2 Results on Molecular Structures

Next we would like to show the comprehensive SMR on the CTS results with respect to different backbone models, as in Tables 5 and 6.

Table 5: This table lists the sequence monotonic ratio (SMR, %) on calibrated Tanimoto similarity (CST) w.r.t. the top-1, top-2, and top-3 directions. The backbone model is the pretrained MoFlow on ZINC250k.

Edit Method	top-K	**Tanimoto top-1**						**Tanimoto top-2**						**Tanimoto top-3**					
	γ	2		3		4		2		3		4		2		3		4	
	τ	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2
Random		35.0	36.0	23.0	25.0	12.0	15.0	34.5	36.0	22.5	25.0	11.5	14.0	34.0	36.0	22.0	24.0	11.0	13.7
Variance		32.0	36.0	24.0	28.0	12.0	16.0	31.5	35.5	21.0	26.5	10.5	16.0	30.3	35.3	20.0	25.0	10.0	15.0
SeFa		23.0	23.0	4.0	4.0	0.0	0.0	19.0	19.0	4.0	4.0	0.0	0.0	17.3	17.3	3.3	3.3	0.0	0.0
DisCo		8.0	15.0	7.0	14.0	2.0	8.0	7.5	13.5	6.0	12.5	2.0	8.0	7.0	13.0	5.3	11.7	2.0	7.7
	Equation (24)	39.0	40.0	27.0	28.0	15.0	18.0	38.5	40.0	26.0	28.0	15.0	18.0	37.0	39.0	24.7	27.3	14.3	17.3
GraphCG-P	Equation (25)	35.0	37.0	19.0	22.0	8.0	11.0	33.5	36.5	18.5	21.5	7.0	10.5	31.7	34.7	17.7	20.7	6.3	9.3
	Equation (26)	44.0	46.0	32.0	34.0	16.0	18.0	42.5	44.5	30.0	32.0	15.0	17.5	41.7	44.0	29.0	31.0	13.7	16.3
	Equation (24)	37.0	42.0	25.0	26.0	11.0	14.0	37.0	40.0	23.0	25.5	11.0	13.5	36.3	39.0	22.0	24.3	10.3	13.3
GraphCG-R	Equation (25)	35.0	37.0	19.0	22.0	8.0	11.0	33.5	36.5	18.5	21.5	7.0	10.5	31.7	34.7	17.7	20.7	6.3	9.3
	Equation (26)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 6: This table lists the sequence monotonic ratio (SMR, %) on calibrated Tanimoto similarity (CST) w.r.t. the top-1, top-2, and top-3 directions. The backbone model is the pretrained HierVAE on ChEMBL.

Edit Method	top-K	**Tanimoto top-1**						**Tanimoto top-2**						**Tanimoto top-3**					
	γ	2		3		4		2		3		4		2		3		4	
	τ	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2	0	0.2
Random		14.0	45.0	14.0	45.0	14.0	43.0	11.0	43.5	11.0	43.5	11.0	42.5	10.0	42.3	10.0	42.3	9.3	41.7
Variance		28.0	64.0	23.0	59.0	19.0	55.0	22.5	59.5	19.5	57.0	17.5	55.0	20.3	54.3	18.3	52.7	15.7	50.3
SeFa		4.0	41.0	4.0	41.0	4.0	41.0	3.0	41.0	3.0	41.0	3.0	41.0	2.3	36.0	2.3	36.0	2.3	36.0
	Equation (24)	23.0	61.0	19.0	57.0	15.0	53.0	19.0	59.0	16.5	56.5	13.5	53.0	17.0	55.3	15.3	53.7	12.7	50.7
GraphCG-P	Equation (25)	62.0	77.0	40.0	73.0	32.0	65.0	60.5	74.0	38.5	67.5	29.0	61.0	59.3	70.3	36.0	64.3	26.3	57.7
	Equation (26)	29.0	71.0	28.0	70.0	27.0	69.0	22.0	62.0	21.5	61.5	20.5	61.0	18.7	57.7	18.3	57.3	17.7	56.7
	Equation (24)	16.0	56.0	16.0	56.0	15.0	55.0	13.5	47.0	13.5	47.0	12.0	47.0	11.7	44.7	11.7	44.7	10.7	43.3
GraphCG-R	Equation (25)	61.0	74.0	42.0	67.0	30.0	55.0	59.0	69.5	40.0	64.0	29.5	54.5	57.7	67.7	38.0	62.3	28.7	53.7
	Equation (26)	25.0	57.0	24.0	57.0	21.0	55.0	20.0	55.0	19.5	54.5	17.0	52.0	17.3	52.7	17.0	52.3	15.0	50.0

F.3 Results on Molecular Properties

By far, we have been mainly focusing on the structure change of the output sequence of molecules. Yet, some works (Seybold et al., 1987) also proved that certain key components of the molecules can be closely related to the molecular properties. We summarize the properties into roughly three categories, and the SMR on 8 properties with different γ, τ are listed below.

1. Physical properties, including water-octanol partition coefficient (LogP) (Comer & Tam, 2001), topological polar surface area (tPSA) (Ertl et al., 2000)), and molecular weight (MW).
2. Drug-related molecular properties, including drug-likeness (QED) (Bickerton et al., 2012) and synthetic accessibility (SA) (Ertl & Schuffenhauer, 2009).
3. Biological properties, including three binding affinity tasks (DRD2, JNK3, GSK3β). All the oracle labels are provided from the previous works (Olivecrona et al., 2017; Jin et al., 2020b; Sun et al., 2017).

Table 7: This table lists the sequence monotonic ratio (SMR, %) on 8 molecule properties with $\gamma = 2, \tau = 0$.

Model & Dataset	Edit Method	h	Physical			Drug-related		Bioactivity		
			LogP ↑	tPSA ↑	MW ↑	QED ↑	SA ↑	DRD2 ↑	JNK3 ↑	GSK3β ↑
MoFlow ZINC250k	Random	–	22.0	28.0	23.0	24.0	24.0	21.0	27.0	24.0
	Variance	–	24.0	24.0	18.0	21.0	19.0	20.0	26.0	22.0
	SeFa	–	20.0	20.0	23.0	21.0	22.0	22.0	21.0	22.0
	DisCo	–	5.0	5.0	5.0	4.0	4.0	5.0	6.0	5.0
	GraphCG-P	Equation (24)	26.0	30.0	24.0	25.0	22.0	23.0	27.0	26.0
		Equation (25)	25.0	29.0	25.0	24.0	27.0	24.0	25.0	27.0
		Equation (26)	27.0	39.0	28.0	27.0	28.0	26.0	39.0	32.0
	GraphCG-R	Equation (24)	27.0	27.0	26.0	26.0	25.0	23.0	33.0	28.0
		Equation (25)	25.0	29.0	25.0	24.0	27.0	24.0	25.0	27.0
		Equation (26)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HierVAE ChEMBL	Random	–	3.0	5.0	4.0	2.0	2.0	1.0	11.0	8.0
	Variance	–	9.0	9.0	8.0	11.0	9.0	7.0	25.0	22.0
	SeFa	–	1.0	0.0	2.0	0.0	0.0	0.0	1.0	2.0
	GraphCG-P	Equation (24)	7.0	12.0	9.0	6.0	5.0	7.0	21.0	15.0
		Equation (25)	52.0	56.0	50.0	50.0	54.0	52.0	55.0	53.0
		Equation (26)	7.0	12.0	3.0	3.0	4.0	2.0	30.0	20.0
	GraphCG-R	Equation (24)	2.0	6.0	3.0	2.0	2.0	2.0	11.0	9.0
		Equation (25)	52.0	54.0	49.0	49.0	52.0	51.0	54.0	53.0
		Equation (26)	6.0	10.0	4.0	6.0	3.0	3.0	24.0	19.0

Table 8: This table lists the sequence monotonic ratio (SMR, %) on 8 molecule properties with $\gamma = 2, \tau = 0.2$.

Model & Dataset	Edit Method	h	Physical			Drug-related		Bioactivity		
			LogP ↑	tPSA ↑	MW ↑	QED ↑	SA ↑	DRD2 ↑	JNK3 ↑	GSK3β ↑
MoFlow ZINC250k	Random	–	24.0	28.0	26.0	27.0	25.0	22.0	29.0	26.0
	Variance	–	25.0	25.0	22.0	23.0	21.0	22.0	27.0	23.0
	SeFa	–	20.0	20.0	23.0	21.0	22.0	22.0	21.0	22.0
	DisCo	–	7.0	7.0	8.0	7.0	5.0	8.0	9.0	9.0
	GraphCG-P	Equation (24)	31.0	32.0	27.0	27.0	26.0	28.0	29.0	27.0
		Equation (25)	28.0	29.0	27.0	25.0	28.0	26.0	27.0	29.0
		Equation (26)	29.0	39.0	31.0	30.0	29.0	27.0	39.0	33.0
	GraphCG-R	Equation (24)	28.0	29.0	28.0	28.0	27.0	25.0	34.0	29.0
		Equation (25)	28.0	29.0	27.0	25.0	28.0	26.0	27.0	29.0
		Equation (26)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HierVAE ChEMBL	Random	–	12.0	14.0	13.0	10.0	11.0	9.0	18.0	16.0
	Variance	–	24.0	16.0	20.0	24.0	24.0	18.0	31.0	26.0
	SeFa	–	18.0	4.0	35.0	1.0	14.0	3.0	7.0	3.0
	GraphCG-P	Equation (24)	19.0	20.0	24.0	15.0	16.0	13.0	25.0	20.0
		Equation (25)	53.0	56.0	52.0	51.0	56.0	53.0	57.0	54.0
		Equation (26)	16.0	29.0	15.0	16.0	17.0	22.0	32.0	25.0
	GraphCG-R	Equation (24)	14.0	14.0	26.0	8.0	15.0	12.0	20.0	16.0
		Equation (25)	53.0	54.0	51.0	50.0	54.0	52.0	56.0	54.0
		Equation (26)	21.0	21.0	16.0	18.0	21.0	17.0	29.0	27.0

Table 9: This table lists the sequence monotonic ratio (SMR, %) on 8 molecule properties with $\gamma = 3, \tau = 0$.

Model & Dataset	Edit Method	h	Physical			Drug-related		Bioactivity		
			LogP ↑	tPSA ↑	MW ↑	QED ↑	SA ↑	DRD2 ↑	JNK3 ↑	GSK3β ↑
MoFlow ZINC250k	Random	–	10.0	17.0	9.0	10.0	10.0	9.0	15.0	11.0
	Variance	–	10.0	10.0	6.0	7.0	6.0	6.0	18.0	11.0
	SeFa	–	1.0	1.0	4.0	2.0	3.0	3.0	2.0	3.0
	DisCo	–	3.0	3.0	3.0	2.0	2.0	3.0	4.0	4.0
	GraphCG-P	Equation (24)	11.0	16.0	9.0	10.0	7.0	9.0	15.0	13.0
		Equation (25)	10.0	13.0	9.0	8.0	11.0	8.0	11.0	11.0
		Equation (26)	14.0	26.0	14.0	14.0	16.0	10.0	26.0	16.0
	GraphCG-R	Equation (24)	10.0	17.0	8.0	13.0	10.0	10.0	16.0	12.0
		Equation (25)	10.0	13.0	9.0	8.0	11.0	8.0	11.0	11.0
		Equation (26)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HierVAE ChEMBL	Random	–	3.0	5.0	4.0	2.0	2.0	1.0	11.0	8.0
	Variance	–	5.0	6.0	4.0	7.0	6.0	4.0	21.0	17.0
	SeFa	–	1.0	0.0	2.0	0.0	0.0	0.0	1.0	2.0
	GraphCG-P	Equation (24)	5.0	8.0	8.0	2.0	2.0	3.0	17.0	13.0
		Equation (25)	11.0	16.0	14.0	11.0	16.0	8.0	26.0	20.0
		Equation (26)	6.0	11.0	2.0	3.0	3.0	2.0	30.0	19.0
	GraphCG-R	Equation (24)	2.0	6.0	3.0	2.0	2.0	2.0	11.0	9.0
		Equation (25)	14.0	17.0	15.0	12.0	14.0	13.0	24.0	21.0
		Equation (26)	6.0	9.0	3.0	5.0	3.0	3.0	23.0	19.0

Table 10: This table lists the sequence monotonic ratio (SMR, %) on 8 molecule properties with $\gamma = 3, \tau = 0.2$.

Model & Dataset	Edit Method	h	Physical			Drug-related		Bioactivity		
			LogP ↑	tPSA ↑	MW ↑	QED ↑	SA ↑	DRD2 ↑	JNK3 ↑	GSK3β ↑
MoFlow ZINC250k	Random	–	13.0	17.0	12.0	14.0	11.0	12.0	16.0	12.0
	Variance	–	12.0	11.0	9.0	9.0	8.0	8.0	19.0	14.0
	SeFa	–	1.0	1.0	4.0	2.0	3.0	3.0	2.0	3.0
	DisCo	–	5.0	6.0	6.0	5.0	5.0	6.0	8.0	8.0
	GraphCG-P	Equation (24)	16.0	18.0	11.0	13.0	11.0	13.0	15.0	17.0
		Equation (25)	12.0	13.0	11.0	9.0	12.0	10.0	13.0	13.0
		Equation (26)	16.0	26.0	16.0	17.0	17.0	12.0	26.0	17.0
	GraphCG-R	Equation (24)	12.0	17.0	10.0	14.0	12.0	11.0	18.0	12.0
		Equation (25)	12.0	13.0	11.0	9.0	12.0	10.0	13.0	13.0
		Equation (26)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HierVAE ChEMBL	Random	–	12.0	14.0	13.0	10.0	11.0	9.0	18.0	15.0
	Variance	–	20.0	15.0	16.0	20.0	20.0	14.0	27.0	21.0
	SeFa	–	18.0	4.0	35.0	1.0	14.0	3.0	7.0	3.0
	GraphCG-P	Equation (24)	15.0	17.0	23.0	14.0	15.0	12.0	21.0	19.0
		Equation (25)	23.0	24.0	23.0	22.0	28.0	21.0	30.0	31.0
		Equation (26)	15.0	28.0	14.0	16.0	16.0	21.0	32.0	24.0
	GraphCG-R	Equation (24)	14.0	14.0	26.0	8.0	15.0	12.0	20.0	16.0
		Equation (25)	23.0	27.0	23.0	23.0	23.0	24.0	33.0	32.0
		Equation (26)	20.0	20.0	15.0	17.0	20.0	16.0	29.0	27.0

Table 11: This table lists the sequence monotonic ratio (SMR, %) on 8 molecule properties with $\gamma = 4, \tau = 0$.

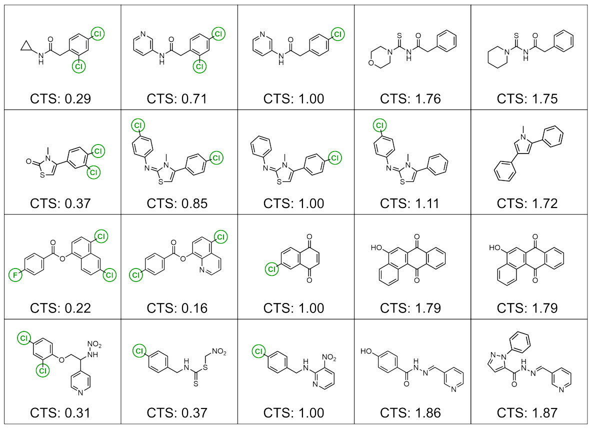
Model & Dataset	Edit Method	h	Physical			Drug-related		Bioactivity		
			LogP ↑	tPSA ↑	MW ↑	QED ↑	SA ↑	DRD2 ↑	JNK3 ↑	GSK3β ↑
MoFlow ZINC250k	Random	–	3.0	7.0	4.0	4.0	3.0	3.0	6.0	5.0
	Variance	–	3.0	4.0	2.0	2.0	2.0	2.0	7.0	4.0
	SeFa	–	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	DisCo	–	1.0	1.0	1.0	1.0	2.0	1.0	2.0	2.0
	GraphCG-P	Equation (24)	6.0	6.0	4.0	5.0	2.0	5.0	6.0	7.0
		Equation (25)	3.0	4.0	2.0	1.0	3.0	2.0	5.0	3.0
		Equation (26)	4.0	12.0	5.0	4.0	4.0	3.0	11.0	6.0
	GraphCG-R	Equation (24)	4.0	6.0	2.0	3.0	4.0	3.0	7.0	3.0
		Equation (25)	3.0	4.0	2.0	1.0	3.0	2.0	5.0	3.0
		Equation (26)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HierVAE ChEMBL	Random	–	2.0	4.0	3.0	2.0	1.0	1.0	9.0	7.0
	Variance	–	2.0	6.0	3.0	5.0	4.0	2.0	17.0	14.0
	SeFa	–	1.0	0.0	2.0	0.0	0.0	0.0	1.0	2.0
	GraphCG-P	Equation (24)	4.0	6.0	5.0	1.0	2.0	2.0	15.0	11.0
		Equation (25)	5.0	7.0	6.0	5.0	5.0	4.0	17.0	15.0
		Equation (26)	5.0	10.0	1.0	2.0	2.0	2.0	28.0	18.0
	GraphCG-R	Equation (24)	1.0	5.0	3.0	2.0	1.0	2.0	10.0	9.0
		Equation (25)	4.0	9.0	6.0	5.0	5.0	6.0	17.0	15.0
		Equation (26)	4.0	7.0	2.0	4.0	2.0	2.0	20.0	18.0

Table 12: This table lists the sequence monotonic ratio (SMR, %) on 8 molecule properties with $\gamma = 4, \tau = 0.2$.

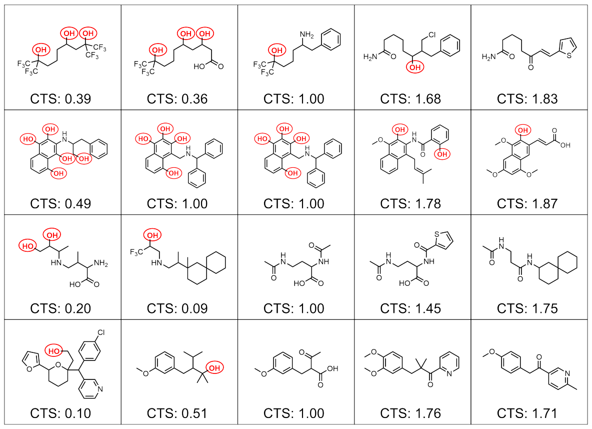
Model & Dataset	Edit Method	h	Physical			Drug-related		Bioactivity		
			LogP ↑	tPSA ↑	MW ↑	QED ↑	SA ↑	DRD2 ↑	JNK3 ↑	GSK3β ↑
MoFlow ZINC250k	Random	–	6.0	8.0	7.0	8.0	6.0	6.0	7.0	6.0
	Variance	–	6.0	6.0	5.0	4.0	4.0	4.0	8.0	7.0
	SeFa	–	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	DisCo	–	4.0	4.0	4.0	4.0	5.0	6.0	6.0	5.0
	GraphCG-P	Equation (24)	10.0	11.0	6.0	9.0	9.0	8.0	7.0	7.0
		Equation (25)	4.0	6.0	5.0	5.0	7.0	3.0	5.0	5.0
		Equation (26)	8.0	12.0	8.0	7.0	6.0	6.0	11.0	10.0
	GraphCG-R	Equation (24)	6.0	6.0	5.0	6.0	6.0	5.0	8.0	4.0
		Equation (25)	4.0	6.0	5.0	5.0	7.0	3.0	5.0	5.0
		Equation (26)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HierVAE ChEMBL	Random	–	10.0	14.0	13.0	10.0	11.0	9.0	18.0	15.0
	Variance	–	16.0	15.0	14.0	16.0	19.0	11.0	23.0	18.0
	SeFa	–	18.0	4.0	35.0	1.0	14.0	3.0	7.0	3.0
	GraphCG-P	Equation (24)	14.0	16.0	20.0	13.0	15.0	12.0	19.0	17.0
		Equation (25)	17.0	19.0	17.0	20.0	17.0	19.0	27.0	28.0
		Equation (26)	14.0	27.0	13.0	15.0	15.0	21.0	30.0	24.0
	GraphCG-R	Equation (24)	13.0	13.0	25.0	8.0	14.0	11.0	19.0	16.0
		Equation (25)	19.0	20.0	21.0	19.0	18.0	18.0	24.0	26.0
		Equation (26)	20.0	18.0	14.0	16.0	20.0	16.0	29.0	26.0

F.4 Visualization

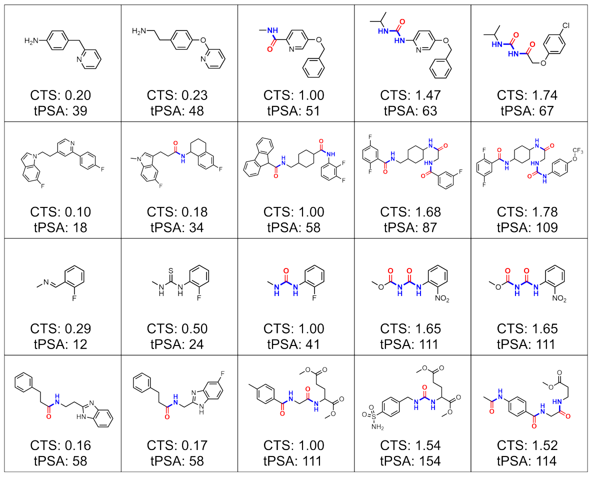
For a more comprehensive visualization of the steerable factors in molecular graphs, we demonstrate 16 molecular graph paths along the 4 selected directions in Figure 6, and the backbone DGM is HierVAE pretrained on ChEMBL. The CTS holds good monotonic trend in all these sequences. Each direction shows certain unique changes in the molecular structures, *i.e.*, the steerable factors in molecules. Some structural changes are reflected in molecular properties. We expand all the details below. In Figure 6(a) and Figure 6(b), the number of halogen atoms and hydroxyl groups (in alcohols and phenols) in the molecules decrease from left to right, respectively. In Figure 6(c), the number of amides in the molecules increases along the path. As a result, the topological polar surface area (tPSA) of the molecules increase accordingly, which is a key molecular property for the prediction of drug transport properties, *e.g.*, permeability (Ertl et al., 2000). In Figure 3(d), the flexible chain length, marked by the number of ethylene (CH_2CH_2) units, increases from left to right. Since the number of rotatable bonds (NRB) measures the molecular flexibility, it also increases accordingly (Veber et al., 2002).



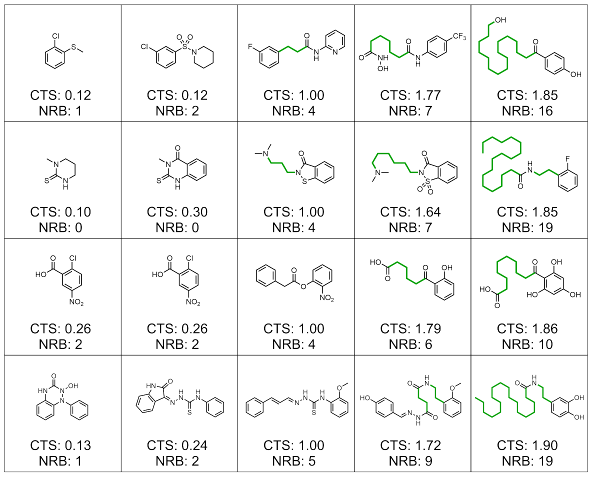
(a) Steerable factor: number of halogens.



(b) Steerable factor: number of hydroxyls.



(c) Steerable factor: number of amides.



(d) Steerable factor: chain length.

Figure 6: GraphCG for molecular graph editing. We visualize the output molecules and CTS on four directions with four sequences each, where each sequence consists of five steps. The center point is the anchor molecule, and the other four points correspond to step size with -3, -1.8, 1.8, and 3 respectively. Figure 6(a) to Figure 6(c) show how functional groups in the molecules can be viewed as the steerable factors as they change along the path, such as halogen atoms, hydroxyl groups and amides. Figure 6(d) illustrates the effect on the steerable factor on the length of flexible chains in the molecules. Notebly, certain properties change together with molecular structures, like topological polar surface area (tPSA) and number of rotatable bonds (NRB).

Since we have determined four directions with semantic information matching with the domain knowledge (fragments), then we can check if the disentanglement measure changes before and after editing. We show the results in Table 13.

Table 13: Disentanglement measure before and after editing. The corresponding model is the GraphCG-P with Equation (25), and the backbone generative model is HierVAE pretrained on ChEMBL. The better performance is marked in **bold**.

Fragment		BetaVAE ↑	MIG ↑	SAP ↑
Halogen	after editing	0.617	0.010	**0.017**
	before editing	**0.950**	**0.062**	**0.017**
Hydroxyls	after editing	0.833	0.031	0.017
	before editing	**0.933**	**0.113**	**0.067**
Amide	after editing	0.400	0.041	**0.017**
	before editing	**0.933**	**0.136**	**0.017**
Chain length	after editing	0.400	**0.051**	0.000
	before editing	**0.700**	0.020	**0.017**

F.5 Ablation Studies on Coefficients c_1, c_2, c_3

Table 14: Ablation studies on coefficients c_1, c_2, c_3. Backbone DGM is MoFlow, dataset is ZINC250K, and the editing model is GraphCG-R with Equation (24). The optimal values are $c_1 = 2, c_2 = 1, c_3 = 1$, and they are reported in Tables 2 and 5.

c_1	c_2	c_3	Tanimoto top-1				Tanimoto top-3			
			3		4		3		4	
			0	0.2	0	0.2	0	0.2	0	0.2
1	0	0	22.0	23.0	11.0	13.0	19.7	21.0	10.7	12.7
1	0	1	19.0	20.0	10.0	12.0	18.0	19.0	9.0	10.7
1	1	0	24.0	26.0	11.0	13.0	21.3	24.7	9.3	12.3
1	1	1	25.0	26.0	11.0	14.0	21.7	24.3	10.3	13.3
2	0	0	20.0	22.0	9.0	11.0	19.0	20.7	8.7	10.7
2	0	1	21.0	21.0	9.0	12.0	19.0	20.0	8.7	10.3
2	1	0	24.0	26.0	11.0	13.0	21.3	24.0	9.3	12.0
2	1	1	25.0	26.0	11.0	14.0	22.0	24.3	10.3	13.3

Table 15: Ablation studies on coefficients c_1, c_2, c_3. Backbone DGM is HierVAE, dataset is ChEMBL, and the editing model is GraphCG-P with Equation (25). The optimal values are $c_1 = 2, c_2 = 1, c_3 = 1$, and they are reported in Tables 2 and 6.

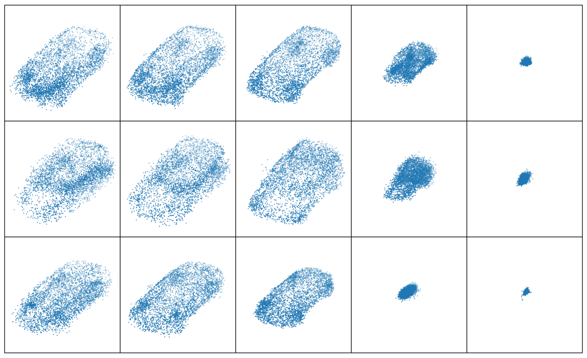
c_1	c_2	c_3	Tanimoto top-1				Tanimoto top-3			
			3		4		3		4	
			0	0.2	0	0.2	0	0.2	0	0.2
1	0	0	28.0	55.0	16.0	55.0	23.3	54.3	16.0	54.3
1	0	1	28.0	55.0	16.0	55.0	23.3	54.3	16.0	54.3
1	1	0	34.0	61.0	26.0	55.0	32.0	59.0	23.3	53.3
1	1	1	34.0	61.0	26.0	55.0	32.0	59.0	23.3	53.3
2	0	0	28.0	55.0	16.0	55.0	23.3	54.3	16.0	54.3
2	0	1	28.0	55.0	16.0	55.0	23.3	54.3	16.0	54.3
2	1	0	40.0	73.0	32.0	65.0	36.0	64.3	26.3	57.7
2	1	1	40.0	73.0	32.0	65.0	36.0	64.3	26.3	57.7

G Results: Point Clouds

Here we compare two editing functions, *i.e.*, the key function design in GraphCG. We provide the visualizations for the linear editing function in Equation (25), and non-linear editing function in Equation (26).

First, we want to highlight that all the samples are generated randomly. In the linear case in Appendix G.1, we can observe that the shape of the airplanes, cars, and chairs, are steerable using GraphCG. We also find it interesting that GraphCG can steer more finger-trained factors, like modifying the airplane engines. However, in the non-linear case, the diversity of the edited data is smaller. This can be observed from the middle columns in Appendix G.2. We conjecture that this is also related to the backbone DGMs and datasets since GraphCG on molecular data does not have this issue. We will leave this for future exploration.

G.1 Linear Editing Function


(a) Steerable factor: size (direction 1).

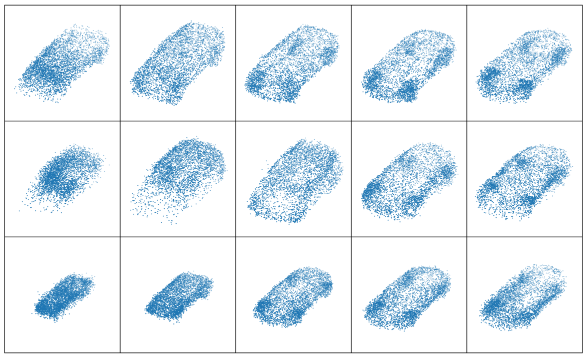
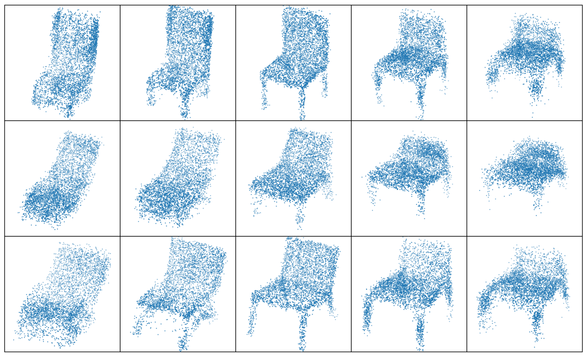

(b) Steerable factor: size (direction 2).

Figure 7: GraphCG for point clouds (Car) editing. The steerable factors on this dataset are not obvious, and here we only plot the car size editable with two directions.


(a) Steerable factor: leg height (direction 1).

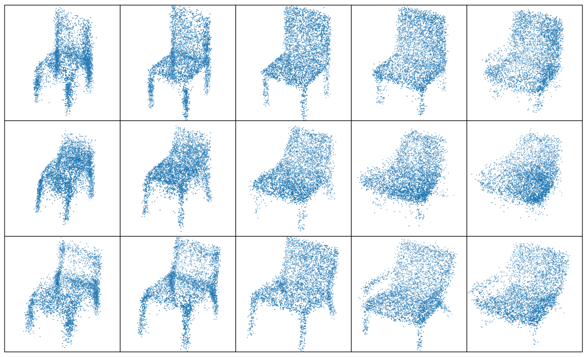
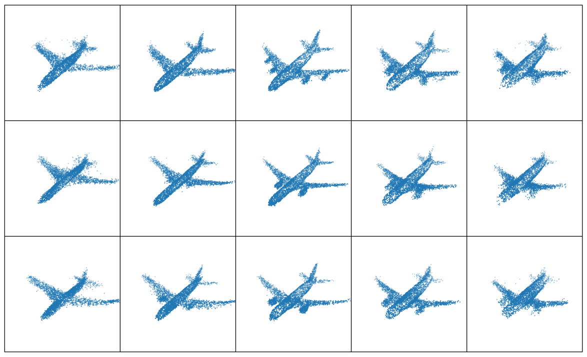
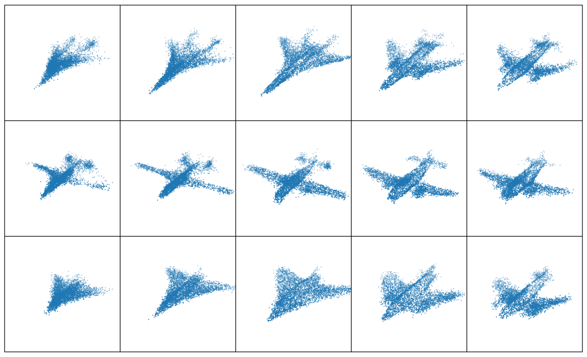
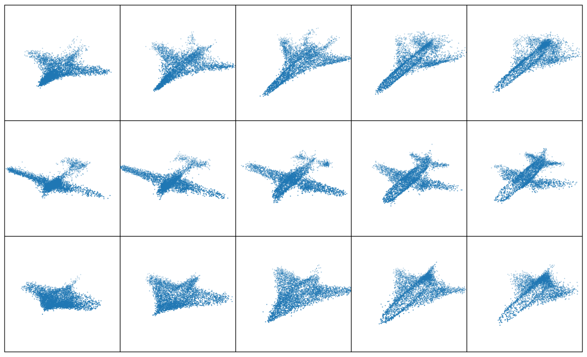
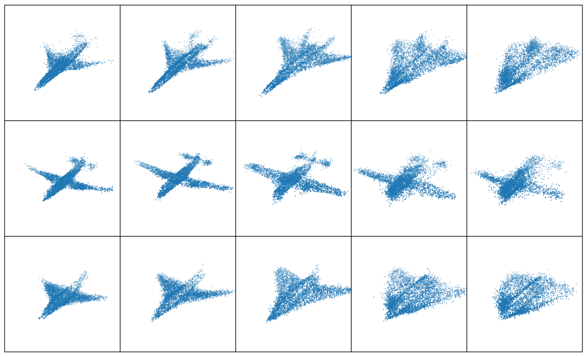

(b) Steerable factor: seat size (direction 2).

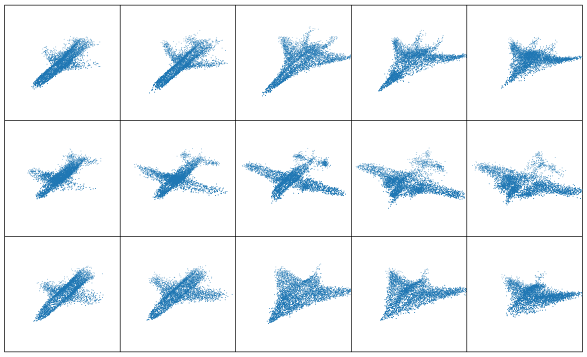
Figure 8: GraphCG for point clouds (Chair) editing. It can successfully reflect these steerable factors: leg height, and seat size.


(a) Steerable factor: engine (direction 1).


(b) Steerable factor: engine (direction 1).


(c) Steerable factor: fuselage length (direction 2).


(d) Steerable factor: wing size (direction 3).


(e) Steerable factor: wing shape(direction 4).

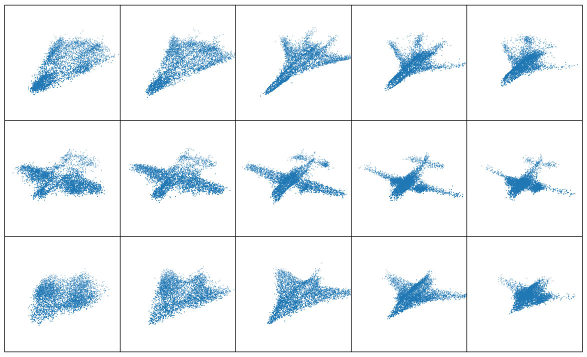
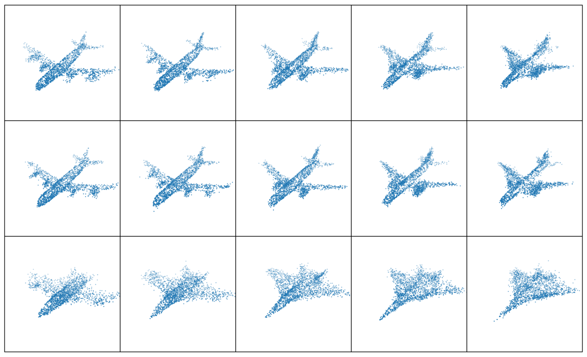

(f) Steerable factor: wing thickness (direction 5).

Figure 9: GraphCG for point clouds (Airplane) editing. It can successfully reflect these steerable factors: engine, fuselage length, wing size, wing shape, and wing thickness.

G.2 Non-linear Editing Function


(a) Steerable factor: wing shape (direction 1).

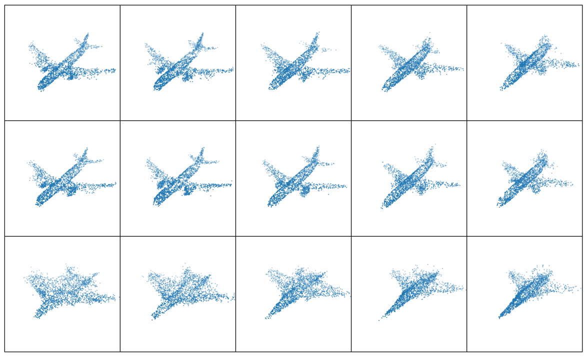
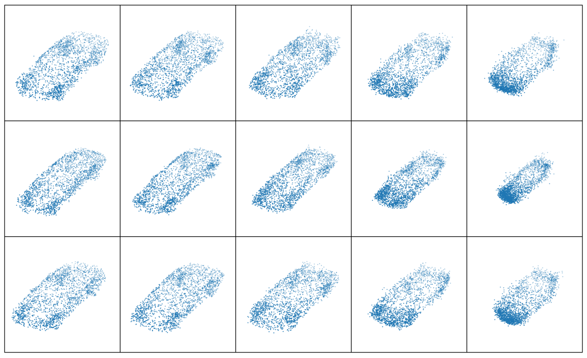

(b) Steerable factor: wing length (direction 2).

Figure 10: GraphCG for point clouds (Airplane) editing. It can successfully reflect these steerable factors: wing shape and wing length.


(a) Steerable factor: size (direction 1).

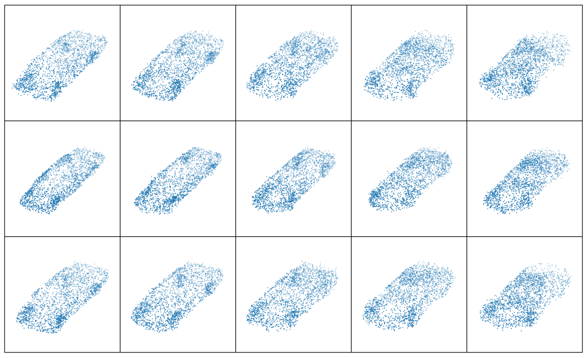
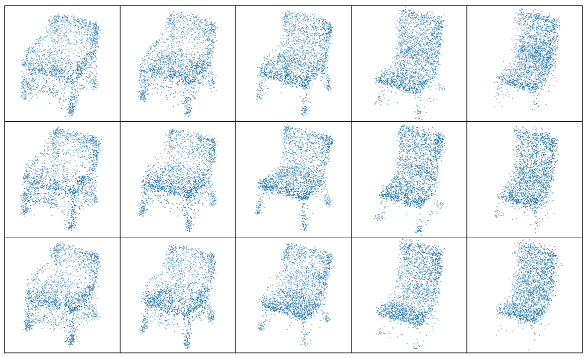

(b) No obvious steerable factor (direction 2).

Figure 11: GraphCG for point clouds (Car) editing. The steerable factors on this dataset are not obvious, and here we only plot the car size editable with one directions.


(a) Steerable factor: leg height (direction 1).

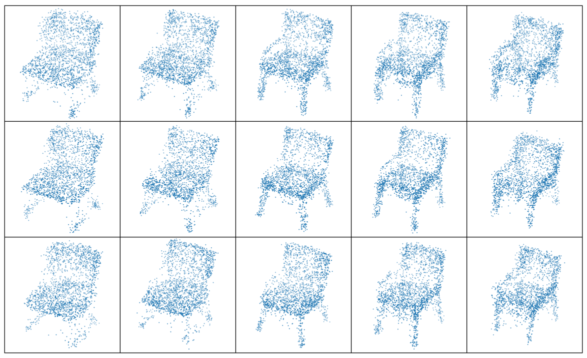

(b) Steerable factor: seat size (direction 2).

Figure 12: GraphCG for point clouds (Chair) editing. It can successfully reflect these steerable factors: leg height, and seat size.