Symmetry-induced Disentanglement on Graphs

Giangiacomo Mercatali
University of Manchester

André Freitas
Idiap Research Institute
University of Manchester

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

Abstract

Learning disentangled representations is important for unraveling the underlying complex interactions between latent generative factors. Disentanglement has been formalized using a symmetry-centric notion for unstructured spaces, however, graphs have eluded a similarly rigorous treatment. We fill this gap with a new notion of conditional symmetry for disentanglement, and leverage tools from Lie algebras to encode graph properties into subgroups using suitable adaptations of generative models such as Variational Autoencoders. Unlike existing works on disentanglement, the proposed models segregate the latent space into uncoupled and entangled parts. Experiments on synthetic and real datasets suggest that these models can learn effective disengaged representations, and improve performance on downstream tasks such as few-shot classification and molecular generation.

1 Introduction

Disentanglement represents a fundamental desideratum in learning with limited supervision because it captures information about the salient (or explanatory) factors of variation in the data, and isolates information about each specific factor in only a few dimensions, thus unraveling the interactions underlying complex data [2, 60]. Disentangled representations have often been deemed responsible for providing neural models with the ability to improve performance on real-world downstream tasks [2, 20, 36, 59, 60]. Empirically, they have been shown to enable, or facilitate, critical properties such as sample efficiency [21] and generalization [32, 42, 70].

An interesting perspective on disentanglement has been recently proposed by Higgins et al. [20]. Specifically, they advocate viewing disentanglement as decomposition of the latent space of embeddings into subspaces, each of which can be transformed independently under the action of a single subgroup specific to the subspace, without affecting others. These subgroups arise, in turn, from the decomposition of a symmetry group. This formalism bestows several benefits, e.g., it (a) aligns with the idea that auxiliary information can be exploited for imposing a structure on the latent space [25, 40], and (b) leads to a rigorous theoretical framework for analyzing disentangled models [65, 71], e.g., those based on Variational Autoencoders (VAEs) [27, 50].

While symmetry groups provide a suitable tool both for formalizing [20] and learning [58, 65, 71] disentangled representations in unstructured domains, these techniques have not been investigated in the context of more general and complex data such as graphs. Furthermore, most works on disentanglement for graphs [1, 16, 31, 55, 58, 60] are not designed for generative settings, and the existing deep generative models (DGMs) for graph disentanglement such as [10, 17, 57] do not consider group symmetries. Note that differently from other definitions of disentanglement [2, 21], leveraging group symmetries for disentanglement may unravel fundamental connections across several approaches based on graph neural networks (GNNs), i.e., the state-of-the-art models for embedding graphs [3, 5, 8, 13, 15, 59, 64].

*first-name.last-name@postgrad.manchester.ac.uk

We, therefore, pursue two main goals in this work: (a) providing a rigorous symmetry-based formalism for disentanglement on graphs, and 2) designing novel neural architectures that are able to leverage symmetry groups to learn disengaged representations for graphs. However, accomplishing these objectives requires overcoming some challenges. In particular, graphs often abstract complex interactions, so the underlying latent space may not factorize completely into only disentangled subspaces. Thus, the symmetry-based formalism introduced in [20] does not suffice for graphs. Therefore, we introduce a new, more flexible notion of conditional disentanglement, advocating segregating the latent space into uncoupled and entangled parts. Translating this idea into an efficient algorithm requires further work, and we appeal to a Lie algebra based parameterization to encode the graph properties into subgroups.

Contributions. Our contributions can be summarized as follows:

- a novel notion of symmetry-induced conditional disentanglement (Section 3.1) that generalizes the previous definition from [20] (which we call unconditional disentanglement);
- a parameterization centered on Lie algebra based on the intuition that each separable generative factor can correspond to a Lie algebra coordinate element, and control a single graph property (Section 3.2);
- two algorithms for Symmetry-Induced Disentanglement under unconditional (SIDU) and conditional (SIDC) settings, both using a two-level variational auto-encoding mechanism, namely, one level for the data and other for symmetry group representation (Section 4); and
- a systematic evaluation on several disentanglement metrics [7, 11, 19, 26, 44]. The results demonstrate that the proposed models can outperform contemporary GNN baselines [38, 66]; successfully learn to decouple the entangled part of latent space from the disentangled parts (Section 5.1); and improve performance on downstream tasks such as few-shot classification, compression (Section 5.2) and molecular graph generation (Section 5.3).

We begin with a review of the relevant literature, and then proceed to the proposed framework.

2 Related work

Symmetry-based disentanglement (SBD). Initial works on SBD, such as Caselles-Dupré et al. [6], Quessard et al. [48], propose to use reinforcement learning to achieve irreducible representations of group elements through observation of action transitions. Painter et al. [44] introduce a VAE method that does not require labelled action-transition pairs, and demonstrate its validity within the environment from Caselles-Dupré et al. [6].

More recently, some algebraic approaches have been introduced. Zhu et al. [71] propose to decompose a Lie group into multiple subgroups, and enforce commutativity and independence between subgroups. Yang et al. [65] impose a cyclic group structure, encouraging the group to be isomorphic to the ground truth. Bouchacourt et al. [4] invoke group representation theory for defining distributed latent operators that are able to learn disentangled representations. Tonnaer et al. [58] provide a new metric for evaluation, and a VAE-based model for learning symmetry-based disentangled representations.

Some works leverage data manifolds for SBD. For example, Fumero et al. [12] view disentanglement as a product of low-dimensional sub-manifolds underlying the data space, such that each sub-manifold encodes an explainable data factor. Pfau et al. [46] propose a non-parametric algorithm that aims to discover a decomposition of the data manifold by investigating its holonomy group.

Other works explore SBD as well. For instance, Higgins et al. [22] demonstrate the importance of symmetry transformations and disentangled representation learning in the context of neuroscience. Wang et al. [62] design a contrastive self-supervised learning method that iteratively updates a partition, modeled as a symmetry group, until the considered factor is invariant.

Disentangled GNNs. To our knowledge, Ma et al. [38] provide the first graph disentanglement approach, which involves a routing mechanism aimed at separating the underlying factors. More recently, Yang et al. [66] exploit an attention mechanism for GNNs, and provide two evaluation metrics for disentanglement on graphs. Liu et al. [35] achieve disentanglement by encouraging
we need to impose the condition that $f$ (i.e. data) is conditionally equivariant between the actions on $W$ and on $Z$, i.e., $f : W \times Z \rightarrow O$, such that each $Z_i$ is affected only by $G_i$ and is invariant to (i.e., does not change due to) $G_j$ for all $j \neq i$.

Definition 1 does not allow any part of the latent vector space to be entangled, which is likely to be restrictive for complex applications. Therefore, we propose conditional disentanglement.

Definition 2 (Conditional symmetry-based equivariance). Consider a generative process $b = W \times E \rightarrow O$ mapping the disentangle-able world states $W$ and non-disentangled states $E$ into observations (i.e. data) $O$, and an inference process $h = O \rightarrow Z$ mapping the data into a latent vector space. By composing $b$ and $h$, we obtain $f = h \circ b$, which maps $W \times E$ to $Z$. Given group actions on $W$, namely, $\cdot_W : G \times W \rightarrow W$ and on $Z$, namely, $\cdot_Z : G \times Z \rightarrow Z$, the function $f$ is said to be conditionally equivariant between the actions on $W$ and $Z$ given $E$ if the actions commute with $f$. That is, $g \cdot_Z f(w, e) = f(g \cdot_W w, e), \forall g \in G, \forall w \in W, \forall e \in E$.

This definition of conditional equivariance naturally leads to a new notion of conditional disentangle-ment using a characterization similar to the three conditions for the unconditional case. Specifically, we need to impose the condition that $f$ is conditionally equivariant between the actions of $W$ and $Z$ given $E$ (instead of simply being equivariant between the actions of $W$ and $Z$).

Note that while this definition results in a generalized framework for disentanglement, it does not directly translate into a learning algorithm. Toward that goal, we invoke tools from Lie algebra as described in the next section.
3.2 Disentanglement on graphs

Here, we aim to instantiate Definitions 1 and 2 in the context of models that encode graph-structured data. Drawing inspiration from the previous work on Lie groups and algebras for symmetry-induced disentangled VAEs by Zhu et al. [71], we develop a formalism for disentanglement on graphs induced by Lie groups.

**Lie groups and Lie algebras.** A Lie group $G$ is a group of continuous symmetries [52], and associated with a Lie algebra $g$, which is the tangent space to the identity element of $G$. We can thus parameterize a Lie algebra with a basis $\{A_i\}_{i=1}^k$, where every element in $g$ can be written as $A = A_1t_1 + \ldots + A_kt_k$ using coordinates $t_i$. Elements of the Lie algebra can be mapped back into the Lie group with a matrix exponential map $\exp: g \to G$.

We now provide some intuition into how we can connect the notion of a Lie group $G$ with respect to a Lie group $G$ (owing to the independence between different dimensions). Zhu et al. [71] adapt this penalty in a Lie algebra parameterization setup, as $H_{ij} = \delta^2 g(T)/\delta t_i \delta t_j$, where $g(T) = \exp \left(\sum_{i=1}^k t_iA_i\right)$, and show that if $A_iA_j = 0 \ \forall i, j$ then $H_{ij} = 0$.

3.3 Disentanglement constraints

It is known [65][71] that a symmetry group parameterization is usually not sufficient for a group to be decomposed into subgroups that are parameterized independently by a single coordinate. Therefore, following Zhu et al. [71], we add two additional constraints for enforcing disentanglement. These constraints can be relaxed, and included in the loss function as regularization terms.

**Commutative penalty.** We would like to enforce a coordinate $t_i$ to be identified by single group $\exp(t_iA_i)$, and thus represent a single property variation in the data $\tilde{z}_i$. Mathematically, one can state this desideratum as an equivalence between the exponential map of the sum, and the product over exponential maps. [71] proved this to be achieved under commutativity over the Lie algebra basis. Namely, if $A_iA_j = A_jA_i$ then $\exp \left(\sum_{i=1}^k t_iA_i\right) = \prod_{i=1}^k \exp(t_iA_i)$.

**Hessian penalty** Furthermore, disentanglement can be encouraged using a Hessian penalty [45] based on the fact the Hessian matrix with respect to a disentangled representation is always zero (owing to the independence between different dimensions). Zhu et al. [71] adapt this penalty in a Lie algebra parameterization setup, as $H_{ij} = \delta^2 g(T)/\delta t_i \delta t_j$ where $g(T) = \exp \left(\sum_{i=1}^k t_iA_i\right)$, and show that if $A_iA_j = 0 \ \forall i, j$ then $H_{ij} = 0$. 
We use the following notation for model variables: latent graph embeddings $Z$ (and $\hat{Z}$), graph features $X$, symmetry Lie group structure $T$, and the graph adjacency matrix $A$. We next derive lower bounds on log-likelihood for both the unconditional and conditional settings (see Appendix for details).

**Proposition 1** (Unconditional lower bound). Given two latent variables $Z$ and $T$, we instantiate the unconditional disentanglement from Definition 3 with a probabilistic model that maximizes the log-likelihood of the graph data $G = (X, A)$ by optimizing the following lower bound:

$$
\mathcal{L}_a = \mathbb{E}_{q(Z|G)q(T,Z,A)} \log p(Z|T)p(G|Z) - \mathbb{E}_{q(Z|G)} \text{KL}(q(T|Z,A)||p(T)) - \mathbb{E}_{q(Z|G)} \log q(Z|G).
$$

(1)

**Proposition 2** (Conditional lower bound). We extend Proposition 1 to account for conditional disentanglement as defined in Definition 4. We obtain the following lower bound

$$
\mathcal{L}_c = \mathbb{E}_{q(Z|G)q(T,Z,A)} \log p(\hat{Z}|T,Z)p(G|\hat{Z}) - \mathbb{E}_{q(Z|G)} \text{KL}(q(T|Z,A)||p(Z,T)) - \mathbb{E}_{q(Z|G)} \log q(Z|G).
$$

(2)

**Architecture.** Propositions 1 and 2 are derived respectively in Appendix B and C. The respective losses (1) and (2) are implemented with four neural network modules as follows. Two stochastic encoders ($E_{\text{graph}}$ and $E_{\text{group}}$) embed respectively the graph data $G$ into a latent variable $Z$ via $q(Z|G)$, and $Z$ into the Lie group coordinates $T$ via $q(T|Z)$. Two deterministic decoders ($D_{\text{group}}$ and $D_{\text{graph}}$) reconstruct, respectively, the graph embedding $\hat{Z}$ from $T$ via $p(\hat{Z}|T)$ for the unconditional case, and the graph $G$ from $\hat{Z}$ via $p(G|\hat{Z})$. The decoder network $D_{\text{group}}$ also plays a role in implementing the conditional lower bound, via $p(\hat{Z}|Z,T)$, which computes $\hat{Z}$ conditioned on $Z$ and $T$.

**4.2 Encoders**

**Graph encoder.** We first encode the graph features $X$ and the adjacency matrix $A$ in the latent space $Z$, using an inference network $E_{\text{graph}}$ $q(Z|X,A)$. Similarly to Kipf and Welling [28], we use a single 2-layer network, where the first layer leverages a graph convolutional network (GCN) to reduce the dimension of the graph features to the symmetry group size: $\overline{X} = \text{GCN}(X,A)$. The second layer computes the mean and variance vectors $(\mu = \text{GCN}_{\mu}(\overline{X},A), \log \sigma = \text{GCN}_{\sigma}(\overline{X},A))$, which are...
We evaluate disentanglement with a) an ablation study for assessing our models components, b) qualitative investigations, and c) quantitative metrics. For metrics, we follow the evaluation protocols from Zhu et al. [71]. We provide insights into the effect of parameters on the models and their performance with an ablation study in Section 5.1. For neural architecture, we set each encoder and decoder to consist of 3 layers, where each layer takes as input the graph representation and produces node features from the previous layer.

4.2 Group encoder. We employ $E_{\text{group}}$ to map the latent vector $Z$ into a vector $T$ of $k$ Lie algebra coordinates, where $k$ is the subspace size (e.g., the number of disentangled elements). In our settings, each Lie algebra coordinate $\{t_i\}_{i=1}^k$ is the output of a GCN with two layers. The first one computes a graph embedding $Z = \text{GCN}(Z, A)$, and the second one produces the vectors $\mu = \text{GCN}_{\mu}(Z, A)$, $\log \sigma = \text{GCN}_{\sigma}(Z, A)$, which in turn yield produce $\{t_i\}_{i=1}^k$ with the reparameterization trick. Finally, $\{t_i\}_{i=1}^k$ are combined into a single tensor $T$ using a multilayer perceptron (MLP):

$$q(T|Z, A) = \text{MLP}(q(t_i|Z, A)) \quad \text{with} \quad q(t_i|Z, A) = \mathcal{N}(t_i|\mu_i, \text{diag}(\sigma_i^2)) \quad \text{for} \ i = 1 \ldots, k \quad (3)$$

4.3 Decoders

Group decoder. The module $D_{\text{group}}$ reconstructs the latent variable $\hat{Z}$ using a Lie group $G$ and a Lie algebra $g$. Specifically, we first learn a Lie algebra basis element $\{A_i\}_{i=1}^k \in g$ for each coordinate $t_i$. We then aggregate the coordinates $t_i$ and basis elements $\hat{A}_i$ with an exponential map, to obtain a group representation as

$$p(\hat{Z}|T) = g(T) = \exp(\mathcal{A}(T)) \quad \text{where} \quad \mathcal{A}(T) = \sum_{i=1}^k t_i A_i \quad \text{for} \ g \in G, \ A \in g. \quad (4)$$

The conditional version of $D_{\text{group}}$ is obtained with a slight modification of Eq. 4. $\hat{Z}$ is computed by feeding the mean between $Z$ (e.g. the graph embeddings obtained from $E_{\text{graph}}$), and $\exp(\mathcal{A}(T))$ into a non-linear activation function $\sigma$ such as ReLU:

$$p(\hat{Z}|T, Z) = \sigma(\text{mean}(\exp(\mathcal{A}(T)), Z)), \quad \text{where} \quad Z = \{z_j\}_{j=1}^k. \quad (5)$$

Graph decoder. The network $D_{\text{graph}}$ reconstructs the graph data $\hat{G}$ from the latents $\hat{Z}$. This is achieved by first passing the latent feature to a GCN layer that maps $Z$ into a vector of the same dimension as the original feature vectors, and subsequently into a sigmoid activation function:

$$p(A, X|\hat{Z}) = p(A, X|\sigma(\text{GCN}(\hat{Z}))) \quad (6)$$

4.4 Training

The proposed unconditional and conditional models are trained by optimizing the loss given by

$$\mathcal{L} + \lambda h + \gamma c,$$

where $\mathcal{L}$ is the lower bound from Eq. 1 (unconditional) or Eq. 2 (conditional), and $h$ and $c$ are respectively the hessian and commutative penalties, obtained as regularization terms adapting the implementation from Zhu et al. [71]. We provide insights into the effect of parameters on the models and their performance with an ablation study in Section 5.1. For neural architecture, we set each encoder and decoder to consist of 3 layers, where each layer takes as input the graph features from the previous layer.

5 Experiments

We conducted extensive experiments that we describe now. Section 5.1 evaluates the disentanglement capabilities of the models, Section 5.2 provides a compression and few-shots classification experiment, and Section 5.3 assesses the generation capabilities on molecular datasets.

5.1 Disentanglement evaluation

We evaluate disentanglement with a) an ablation study for assessing our models components, b) qualitative investigations, and c) quantitative metrics. For metrics, we follow the evaluation protocols from Locatello et al. [56], which involve being able to randomly combine a number of generative factors to sample a new data. In our case, we provide the necessary generative factors to generate two types of synthetic random graphs (e.g. Erdos-Renyi and Watts-Strogatz). More details about our dataset construction procedure are provided in Appendix A. We compute 5 metrics, including $\beta$-VAE [19] (Beta), FactorVAE (FVM) [26], Mutual Information Gap (MIG) [7], DCI Disentanglement [11], and Factor Leakage (FL) [44].
Ablation study. We start by evaluating the impact of our model components on the disentanglement metrics, by maintaining all parameters fixed, except from one that varies. We probe the effect from different group sizes, hessian penalties, commutative penalties, and subspace sizes. In Figure 2 we report the mean over 10 runs for our unconditional model on the synthetic Watts-Strogatz dataset. We observe that when the group size is increased, there is an improvement in disentanglement across all metrics, and this can be explained by the idea that with a larger group size, the subgroups, represented by the Lie algebra coordinate $t_i$ can control different data variations. The hessian penalty results to be effectively enhancing disentangled representations, while the commutative penalty is less effective, and we can explain this result because the former requires the Lie algebra basis elements to have mutual products of zeros while subgroup decomposition only requires their commutators to be zeros, which also confirms the results obtained for image datasets [71].

Correlation analysis. Following previous work [66, 38], we report the correlation matrix of the hidden graph features, the goal of which is to show whether the features capture mutually exclusive information, which is achieved when there is a block-wise correlation pattern. In our models we use a group-size of 32 (to match with hidden features size of the other methods), a hessian penalty of 40 and commutative penalty of 5. In Figure 3 we note that both our SID$_U$ and SID$_C$ models achieve a block-wise correlation pattern, however the conditional version has also some other highlighted regions, because in this model involves fixing a latent vector $Z$ to condition the reconstruction. As a result, the conditioning may be responsible for the visualized correlation on the hidden features.

Learned graph variations. In this experiment, we train the SID$_C$ model on Watts-Strogatz (WS) random graphs, which have 3 generative factors, $n$ for number of nodes, $p$ for connectivity (the probability of rewiring each edge), and $k$ for neighborhood ($k$ nearest neighbors in a ring topology). We adapt our model to the visualization implementations provided by Stoehr et al. [57], which report adjacency matrices (black) and graph (white) information, where variations in node attribute values are indicated in blue. In Figure 4 we report the results for the SID$_C$ model by projecting pairs of latent variable traversals to see how the factors vary, and we observe the following. 1) The variable $z_0$ controls the node attribute value. For example, in $\text{adj}_0$ we see the color changing from blue to white along the horizontal axis. 2) The variable $z_1$ controls connectivity, for example in $\text{graph}_0$, in the $z_1$ axis (top to bottom) we see that the number of edges decrease, while in $\text{graph}_2$, along the $z_1$ axis (left to right), the number of edges increase. 3) The variable $z_2$ controls the number of nodes, and we see that in $\text{graph}_1$ and $\text{graph}_2$ the number of nodes decreases along the $z_2$ axis (bottom to top).

Disentanglement metrics. We perform 10 random seed runs, by training for 200 epochs on the proposed synthetic datasets and measure the quantitative metrics on 1000 data points. In terms of
We notice that the FL metric shows a significant performance increase, justified by its ability to capture the disentanglement on each generative factor, which, in our models is controlled by the group size, while in previous approaches is not considered. In terms of the other previously proposed disentangled approaches, DIS and FAC demonstrate higher disentanglement in most cases, when compared to GCN and GAT, however they are outperformed by our methods.

5.2 Compression and few-shots classification

Following the setup from Ge et al. [14], we consider the task of learning a compressed model via multiple pooling operations, and test on few-shots classification capabilities on datasets with multiple classes, including FRANKENSTEIN [43] (2 classes), COLORS-3 [30] (11 classes), Mutagenicity [51] (2 classes), NCI1 [54] (2 c). The baselines include MIA [14], which propose an attention-based pooling for compression, as well as GCN [29], and GAT [61] based autoencoders, equipped with the same pooling layer. All models are set to have a pooling rate of 0.8, and a depth of 3 layers. Our models are set with a group size of 81, hessian penalty of 40, and a commutative penalty of 5.

In Table 1 we observe that our models are able to improve on previous baselines on all metrics on the evaluated datasets, and the conditional version achieves the best performance in most cases. We notice that the FL metric shows a significant performance increase, justified by its ability to capture the disentanglement on each generative factor, which, in our models is controlled by the group size, while in previous approaches is not considered. In terms of the other previously proposed disentangled approaches, DIS and FAC demonstrate higher disentanglement in most cases, when compared to GCN and GAT, however they are outperformed by our methods.

In Table 2 we report the mean and variance across 10 runs for MSE, the classification accuracy, and the size of the trained model. The results show that SIDC and SIDF outperform the considered baselines in terms of few-shots classification accuracy on all datasets, while achieving a smaller compressed size. We hypothesize that the superior accuracy performance of our models is due to a more efficient learning provided by disentangled representations, which have previously demonstrated the ability to enhance the accuracy of predictions in the context of generalization tasks [41, 70, 32]. In terms of MSE, our models show an improvement compared to GCN- and GAT-based autoencoders, but they are outperformed by MIA in most cases.

![Figure 4: Graph traversals (attributes and node values) for the combinations of 3 latent variables.](Figure.png)

Table 1: Disentanglement results.

<table>
<thead>
<tr>
<th>Model</th>
<th>FVM</th>
<th>Beta</th>
<th>MIG</th>
<th>DCI</th>
<th>FL</th>
<th>FVM</th>
<th>Beta</th>
<th>MIG</th>
<th>DCI</th>
<th>FL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>69.6±0.8</td>
<td>66.5±0.7</td>
<td>23.2±2.5</td>
<td>43.8±1.6</td>
<td>84.1±0.5</td>
<td>57.0±4.5</td>
<td>61.0±1.4</td>
<td>22.5±3.3</td>
<td>54.1±2.1</td>
<td>71.7±2.6</td>
</tr>
<tr>
<td>GAT</td>
<td>72.6±1.3</td>
<td>66.2±3.4</td>
<td>17.4±1.0</td>
<td>53.2±2.7</td>
<td>79.0±1.1</td>
<td>56.0±3.2</td>
<td>66.3±2.5</td>
<td>31.4±1.9</td>
<td>54.0±2.9</td>
<td>70.2±2.1</td>
</tr>
<tr>
<td>FAC</td>
<td>74.2±1.3</td>
<td>65.7±2.6</td>
<td>26.4±2.0</td>
<td>53.4±4.0</td>
<td>80.3±0.9</td>
<td>67.2±1.7</td>
<td>66.3±3.3</td>
<td>35.1±2.4</td>
<td>53.8±1.5</td>
<td>78.1±3.0</td>
</tr>
<tr>
<td>DIS</td>
<td>73.0±3.5</td>
<td>69.3±2.1</td>
<td>27.1±1.7</td>
<td>52.2±3.9</td>
<td>85.2±2.0</td>
<td>68.1±1.8</td>
<td>64.1±0.5</td>
<td>41.3±1.4</td>
<td>53.5±2.3</td>
<td>72.4±2.3</td>
</tr>
<tr>
<td>SIDC</td>
<td>80.5±1.5</td>
<td>71.6±4.5</td>
<td>28.5±2.8</td>
<td>55.4±2.0</td>
<td>91.0±2.3</td>
<td>83.0±2.9</td>
<td>71.8±3.2</td>
<td>53.2±0.8</td>
<td>58.6±1.9</td>
<td>87.1±4.0</td>
</tr>
<tr>
<td>SIDF</td>
<td>81.9±1.2</td>
<td>74.5±3.2</td>
<td>49.1±3.1</td>
<td>59.7±1.2</td>
<td>92.3±0.9</td>
<td>83.3±0.7</td>
<td>74.1±1.2</td>
<td>52.4±3.3</td>
<td>60.5±1.2</td>
<td>89.8±2.3</td>
</tr>
</tbody>
</table>

baselines, we consider graph VAEs with GAT [61] and GCN [29] encoders, as well as disentangled encoders such as FAC [66] and DIS [38]. We set our models to have a group size of 81, a hessian penalty of 40, and a commutative penalty of 5.
5.3 Molecular graph generation

Molecular generation involves three tasks, random generation, property optimization and constrained optimization, and a suitable setup can be obtained following the approaches implemented via TorchDrug[2] and DIG[33]. The generation process involves 1) learning the distribution of the molecular data and 2) fine-tuning the pre-trained model on one of the tasks. We employ our models during the training phase, using a group size of 81, a hessian penalty of 40 and commutative penalty of 5, and then we follow the reinforcement learning fine-tuning method from GCPN[67]. Similar setups are used from previous molecule generation methods, which we include as baselines: JT-V AE[24], GraphAF[55], GraphDF[37], GCPN[67].

**Random generation.** This task evaluates the quality of randomly generated samples on 4 standard metrics in percentage, including: valid molecules with resampling (Val), valid molecules without resampling (WR), unique molecules (Uni), novel molecules that not appear in the training data (Nov). We train our models for 10 epochs, with a batch size of 32 and the learning rate of 0.001, and compute the metrics over 10K generated molecules. In Table 3 we report results on ZINC[23], QM9[49], and MOSES[47]. We observe that both SIDU and SIDC are able to improve the performance of GraphAF and GCPN on most metrics, while achieving comparable performance with GraphDF. Furthermore, the SIDC model achieves the highest score of validity without resampling on the QM9 dataset. Since our model is not optimized for molecule generation, the positive results of our models indicate that disentangled representations may represent a flexible tool for generating more realistic graphs, and that encouraging the separation of semantic factors into different latent codes, may be beneficial for improving the quality of the samples.

**Property optimization** The goal of the task is to generate novel molecules with high property scores for the penalized logP property, and the quantitative estimation of drug-likeness property (QED). The setup involves pretraining our models for 300 epochs on ZINC random generation, applying the RL fine-tuning procedure from GCPN, and measuring the scores from the top 3 generated molecules. In Table 4 we observe that our models are able to outperform GCPN and GraphAF for both penalized logP and QED, and achieve comparable performance to the more advanced GraphDF model. In Figure 5 we show the molecules generated using SIDC. While our model does not outperform GraphDF, our results are relevant because they demonstrate that the disentangled representations learned from our models can be leveraged to improve the performance of the downstream task of property optimization.

---

Table 2: Compression and few shot classification results.

<table>
<thead>
<tr>
<th>Model</th>
<th>FRANKENSTEIN</th>
<th>COLORS-3</th>
<th>Mutagenicity</th>
<th>NCI1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>Size</td>
<td>Acc.</td>
<td>MSE</td>
</tr>
<tr>
<td>MIA</td>
<td>1.6±0.5</td>
<td>4.9M</td>
<td>65±1</td>
<td>1.8±0.2</td>
</tr>
<tr>
<td>GCN</td>
<td>12.6±2.5</td>
<td>4.2M</td>
<td>39±3</td>
<td>11.2±1.1</td>
</tr>
<tr>
<td>GAT</td>
<td>9.0±1.2</td>
<td>4.2M</td>
<td>26±2</td>
<td>10.1±3.1</td>
</tr>
<tr>
<td>SIDU</td>
<td>4.3±1.1</td>
<td>4.1M</td>
<td>68±2</td>
<td>5.1±1.5</td>
</tr>
<tr>
<td>SIDC</td>
<td>2.8±0.5</td>
<td>4.1M</td>
<td>69±2</td>
<td>1.9±0.4</td>
</tr>
</tbody>
</table>

Table 3: Random graph generation.

<table>
<thead>
<tr>
<th>Model</th>
<th>ZINC Val</th>
<th>WR</th>
<th>Uni</th>
<th>Nov</th>
<th>QM9 Val</th>
<th>WR</th>
<th>Uni</th>
<th>Nov</th>
<th>MOSES Val</th>
<th>WR</th>
<th>Uni</th>
<th>Nov</th>
</tr>
</thead>
<tbody>
<tr>
<td>JT-V AE</td>
<td>100</td>
<td>n/a</td>
<td>100</td>
<td>n/a</td>
<td>100</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>100</td>
<td>n/a</td>
<td>99.96</td>
<td>91.43</td>
</tr>
<tr>
<td>GCPN</td>
<td>100</td>
<td>99.97</td>
<td>100</td>
<td>n/a</td>
<td>100</td>
<td>99.1</td>
<td>n/a</td>
<td>n/a</td>
<td>100</td>
<td>99.9</td>
<td>100</td>
<td>99.99</td>
</tr>
<tr>
<td>GraphAF</td>
<td>100</td>
<td>68</td>
<td>99.1</td>
<td>100</td>
<td>100</td>
<td>67</td>
<td>94.15</td>
<td>88.83</td>
<td>100</td>
<td>71</td>
<td>99.99</td>
<td>100</td>
</tr>
<tr>
<td>GraphDF</td>
<td>100</td>
<td>89.03</td>
<td>99.16</td>
<td>100</td>
<td>100</td>
<td>82.67</td>
<td>97.62</td>
<td>98.1</td>
<td>100</td>
<td>87.58</td>
<td>99.55</td>
<td>100</td>
</tr>
</tbody>
</table>

**SIDU**

<table>
<thead>
<tr>
<th>Model</th>
<th>ZINC Val</th>
<th>WR</th>
<th>Uni</th>
<th>Nov</th>
<th>QM9 Val</th>
<th>WR</th>
<th>Uni</th>
<th>Nov</th>
<th>MOSES Val</th>
<th>WR</th>
<th>Uni</th>
<th>Nov</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIDU</td>
<td>100</td>
<td>78.21</td>
<td>99.12</td>
<td>100</td>
<td>100</td>
<td>75.43</td>
<td>95.22</td>
<td>90.33</td>
<td>100</td>
<td>81.62</td>
<td>99.62</td>
<td>100</td>
</tr>
<tr>
<td>SIDC</td>
<td>100</td>
<td>86.02</td>
<td>99.14</td>
<td>100</td>
<td>100</td>
<td>82.92</td>
<td>96.43</td>
<td>92.28</td>
<td>100</td>
<td>84.29</td>
<td>99.79</td>
<td>100</td>
</tr>
</tbody>
</table>

---

2https://torchdrug.ai/ (Apache license)
Table 4: Property optimization performance.

<table>
<thead>
<tr>
<th></th>
<th>Penalized logP</th>
<th>QED</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st</td>
<td>2nd</td>
<td>3rd</td>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>ZINC</td>
<td>4.52</td>
<td>4.3</td>
<td>4.23</td>
<td>0.948</td>
<td>0.948</td>
</tr>
<tr>
<td>JT-VAE</td>
<td>5.3</td>
<td>4.93</td>
<td>4.49</td>
<td>0.925</td>
<td>0.911</td>
</tr>
<tr>
<td>GCNP</td>
<td>7.98</td>
<td>7.85</td>
<td>7.80</td>
<td>0.948</td>
<td>0.947</td>
</tr>
<tr>
<td>GraphAF</td>
<td>12.23</td>
<td>11.29</td>
<td>11.05</td>
<td>0.948</td>
<td>0.947</td>
</tr>
<tr>
<td>GraphDF</td>
<td>13.7</td>
<td>13.18</td>
<td>13.17</td>
<td>0.948</td>
<td>0.947</td>
</tr>
<tr>
<td>SID_U</td>
<td>12.56</td>
<td>12.46</td>
<td>12.03</td>
<td>0.947</td>
<td>0.948</td>
</tr>
<tr>
<td>SID_C</td>
<td>12.89</td>
<td>12.82</td>
<td>12.27</td>
<td>0.948</td>
<td>0.948</td>
</tr>
</tbody>
</table>

Figure 5: Molecules from SID

Table 5: Constrained optimization performance on 800 molecules used in GraphAF.

<table>
<thead>
<tr>
<th>δ</th>
<th>Imp</th>
<th>Sim</th>
<th>Suc</th>
<th>Imp</th>
<th>Sim</th>
<th>Suc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>13.15±0.89</td>
<td>0.29±0.15</td>
<td>100</td>
<td>14.15±0.86</td>
<td>0.29±0.13</td>
<td>100</td>
</tr>
<tr>
<td>0.2</td>
<td>11.90±0.68</td>
<td>0.33±0.12</td>
<td>100</td>
<td>12.77±0.59</td>
<td>0.32±0.11</td>
<td>100</td>
</tr>
<tr>
<td>0.4</td>
<td>8.21±0.51</td>
<td>0.49±0.09</td>
<td>99.8</td>
<td>9.19±0.43</td>
<td>0.48±0.08</td>
<td>99.6</td>
</tr>
<tr>
<td>0.6</td>
<td>4.98±0.49</td>
<td>0.66±0.05</td>
<td>96.8</td>
<td>4.51±0.80</td>
<td>0.65±0.05</td>
<td>92.1</td>
</tr>
</tbody>
</table>

6 Discussion

Limitations. For quantitative evaluation, we rely on synthetic data, since current metrics are based on the ability to sample the factors and combine them into observations Kim and Mnih [26]. Future work should investigate how to provide metrics that can evaluate disentanglement on real-world datasets. A recent line of work from Khemakhem et al. [25] shows that disentanglement is closely related to model identifiability, i.e., a representation is disentangled when a set of learned parameters is uniquely identified with the parameters in the true distribution. Our approach has not considered or provided proofs for, identifiability, and we leave this as future work.

Conclusion. We formalize the notion of conditional disentanglement on graphs and propose a novel framework for graph disentanglement by leveraging tools from Lie algebras. Based on the new definition of disentanglement, we design a graph VAE based on a Lie group parameterization, and provide a novel ELBO criteria for optimizing conditional disentanglement. Our method achieves superior performance on quantitative disentanglement benchmarks when compared to contemporary disentangled GNNs and other convolutional layers. Finally, we demonstrate strong capabilities on few-shots classification and molecular generation experiments.

7 Acknowledgements

We thank the anonymous reviewers for their helpful comments. GM’s research was partially funded by EPSRC and the BBC under iCASE.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Provided in Appendix

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Provided at https://mercatali.gitlab.io/sid/
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specify the experimental setups in Section 5
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See beginning of Section 5
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See beginning of Section 5

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.3
   (b) Did you mention the license of the assets? [Yes] See Section 5.3
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include our assets at https://mercatali.gitlab.io/sid/
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]
A  Synthetic datasets

We construct two datasets with the PyTorch Geometric library\(^3\) and the NetworkX library\(^4\) based respectively on Watts-Strogatz, and Erdos-Renyi random graphs. The datasets consist in a list of graphs, where we control the generative factors necessary for synthesizing node feature and adjacency matrices in each graph. This allows us to create labels and latent codes for each graph, similarly to the canonical dataset for disentanglement such as Dsprites\(^5\).

**Node feature factors**  The node feature matrix is computed for both datasets as a random tensor of dimension \(N\) nodes \(\times N\) features. The number of node is fixed to 20, while the features are 64. The elements of the feature matrix are sampled from a normal distribution with a given mean and variance, which are two of the generative factors that we control.

**Adjacency matrix factors**  The adjacency matrices are computed via the NetworkX library, where we are able to control the factors responsible for generating the graph. For Watts-Strogatz graphs we have (1) a factor \(k\), which represents the \(k\) nearest neighbors in a ring topology, and (2) factor \(p\), which is the probability of rewiring each edge. For Erdos-Renyi graphs, we control only one factor \(p\), that represents the probability for edge creation.

**Combination**  Watts-Strogatz data has 2 adjacency factors, while Erdos-Renyi has only one, thus, in order to achieve the same number of graphs in both datasets, we fix the variance to 2 on Watts-Strogatz, while we provide 20 values in Erdos-Renyi. The final Watts-Strogatz and Erdos-Renyi datasets have \(40 \times 40 \times 20 = 32,000\) graphs. The combinations for the factors: mean, variance, \(k\), and \(p\) are provided in Table 6. Note that number of nodes is not considered a factor.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Num nodes</th>
<th>NODE FEATURE Mean</th>
<th>Variance</th>
<th>ADJACENCY MATRIX (k)</th>
<th>(p)</th>
<th>Num graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watts-Strogatz</td>
<td>20</td>
<td>40</td>
<td>1</td>
<td>20</td>
<td>40</td>
<td>32K</td>
</tr>
<tr>
<td>Erdos-Renyi</td>
<td>20</td>
<td>40</td>
<td>20</td>
<td>n/a</td>
<td>40</td>
<td>32K</td>
</tr>
</tbody>
</table>

B  Unconditional Disentanglement

The unconditional ELBO (Proposition 1), states that given two latent variables \(Z\) and \(T\) modeling the log-likelihood of the graph data \(G\) is bounded by the following \(L_u\) ELBO:

\[
L_u = \mathbb{E}_{q(Z|G)q(T|Z)} \log p(G|Z)p(Z|T) - \mathbb{E}_{q(Z|G)} \text{KL}(q(T|G, Z)||p(T)) - \mathbb{E}_{q(Z|G)} \log q(Z|G)
\]  

(7)

*Proof.* Using Jensen inequality, we have:

\[
\log p(G) = \log \int_X \int_T p(G, Z, T)
= \log \int_X \int_T p(G, Z, T) \frac{q(T|G, Z)q(Z|G)}{q(T|G, Z)q(Z|G)}
\geq \int_X q(Z|G) \log \int_T p(G, Z, T) \frac{q(T|G, Z)}{q(T|G, Z)q(Z|G)}
\geq \int_X q(Z|G) \log \int_T p(G, Z, T) \frac{q(T|G, Z)}{q(T|G, Z)q(Z|G)} - \mathbb{E}_{q(Z|G)} \log q(Z|G)
\]
By applying Jensen inequality again, we obtain:

\[
\begin{align*}
\log p(G) & \geq \int_Z q(Z|G) \int_T q(T|G,Z) \log \frac{p(G,Z,T)}{q(T|G,Z)} - \mathbb{E}_{q(Z|G)} \log q(Z|G) \\
& = \int_Z q(Z|G) \int_T q(T|G,Z) \log \frac{p(G,Z,T)p(T)}{q(T|G,Z)} - \mathbb{E}_{q(Z|G)} \log q(Z|G) \\
& = \mathbb{E}_{q(Z|G)} \mathbb{E}_{q(T|G,Z)} \log p(G,Z,T) - \mathbb{E}_{q(Z|G)} \int_T q(T|G,Z) \log \frac{q(T|G,Z)}{p(T)} - \mathbb{E}_{q(Z|G)} \log q(Z|G) \\
& = \mathbb{E}_{q(Z|G)} \mathbb{E}_{q(T|G,Z)} \log p(G,Z|T) - \mathbb{E}_{q(Z|G)} \mathbb{KL}(q(T|G,Z)||p(T)) - \mathbb{E}_{q(Z|G)} \log q(Z|G)
\end{align*}
\]

The results follows assuming \(p(G|Z,T) = p(G|Z)\), and noting that for computing \(T\), we only use the adjacency information \(\mathcal{A}\) from \(G\).

\[\square\]

### C Conditional Disentanglement

Using Jensen’s inequality, we note that for any fixed \(\hat{Z}\)

\[
\log p(G, \hat{Z}) = \log \int_Z \int_T p(G, Z, T, \hat{Z})
\]

\[
= \log \int_Z \int_T p(G, Z, T, \hat{Z}) \frac{q(Z,T|G)}{q(Z,T|G)}
\]

\[
= \log \int_Z \int_T p(G, Z, T, \hat{Z}) \frac{q(T|G,Z)q(Z|G)}{q(T|G,Z)q(Z|G)}
\]

\[
\geq \int_Z q(Z|G) \log \int_T p(G, Z, T, \hat{Z}) \frac{q(T|G,Z)}{q(T|G,Z)q(Z|G)}
\]

\[
\geq \int_Z q(Z|G) \log \int_T \frac{p(G,Z,T,\hat{Z})}{q(T|Z,A)} q(T|Z,A) - \mathbb{E}_{q(Z|G)} \log q(Z|G)
\]

\[
= \mathbb{E}_{q(Z|G)} \mathbb{E}_{q(T|Z,A)} \log \frac{p(G,Z,T,\hat{Z})}{q(T|Z,A)} - \mathbb{E}_{q(Z|G)} \log q(Z|G)
\]

\[
= \mathbb{E}_{q(Z|G)} \mathbb{E}_{q(T|Z,A)} \log \left( p(\hat{Z}|Z,T)p(G|\hat{Z},T,Z) \right) - \mathbb{E}_{q(Z|G)} \mathbb{KL}(q(T|Z,A)||p(T,Z)) - \mathbb{E}_{q(Z|G)} \log q(Z|G)
\]

Assuming \(G\) is independent of \(Z\) and \(T\) given \(\hat{Z}\), we immediately get

\[
\log p(G, \hat{Z}) \geq \mathbb{E}_{q(Z|G)} \mathbb{E}_{q(T|Z,A)} \log \left( p(\hat{Z}|Z,T)p(G|\hat{Z}) \right) - \mathbb{E}_{q(Z|G)} \mathbb{KL}(q(T|Z,A)||p(T,Z)) - \mathbb{E}_{q(Z|G)} \log q(Z|G).
\]

\[\square\]
D  Scalability experiment

We perform a graph classification task on large scale datasets, and report their dimensions in Table 7. In Table 8, we report the results for our SID₀ model on social network datasets including IMDB binary, IMDB-multi, COLLAB, as well as macro molecules such as PROTEINS and MUTAG. The baselines for this experiment include FactorGCN [66] and DGCL [31]. For our model we set group-size to 81, hessian penalty to 40 and commutative penalty to 5. We follow the classification setup from Yang et al. [66], with a ten-fold cross-validation procedure, and report accuracy and standard deviation. The results show that both SID₀ and SID₁ are able to improve the performance of the baselines. In particular, SID₁ achieves the highest accuracy on all the datasets.

Table 7: Dimensions of datasets

<table>
<thead>
<tr>
<th></th>
<th>IMDB-B</th>
<th>IMDB-M</th>
<th>COLLAB</th>
<th>PROTEINS</th>
<th>MUTAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphs</td>
<td>1000</td>
<td>500</td>
<td>5000</td>
<td>1113</td>
<td>188</td>
</tr>
<tr>
<td>Classes</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Avg. Nodes</td>
<td>19.77</td>
<td>13.00</td>
<td>74.49</td>
<td>39.06</td>
<td>17.93</td>
</tr>
<tr>
<td>Avg. Edges</td>
<td>96.53</td>
<td>65.94</td>
<td>2457.78</td>
<td>72.82</td>
<td>19.79</td>
</tr>
</tbody>
</table>

Table 8: Scalability

<table>
<thead>
<tr>
<th></th>
<th>IMDB-B</th>
<th>IMDB-M</th>
<th>COLLAB</th>
<th>PROTEINS</th>
<th>MUTAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>FactorGCN</td>
<td>75.3 ± 2.7</td>
<td>-</td>
<td>81.2 ± 1.4</td>
<td>-</td>
<td>89.9 ± 6.5</td>
</tr>
<tr>
<td>DGCL</td>
<td>75.9 ± 0.7</td>
<td>51.9 ± 0.4</td>
<td>81.2 ± 0.3</td>
<td>76.4 ± 0.5</td>
<td>92.1 ± 0.8</td>
</tr>
<tr>
<td>SID₀</td>
<td>76.1 ± 0.2</td>
<td>51.2 ± 0.9</td>
<td>81.9 ± 0.8</td>
<td>76.5 ± 1.5</td>
<td>91.4 ± 1.2</td>
</tr>
<tr>
<td>SID₁</td>
<td>76.5 ± 0.3</td>
<td>52.5 ± 0.2</td>
<td>82.5 ± 0.2</td>
<td>76.9 ± 0.1</td>
<td>92.5 ± 0.5</td>
</tr>
</tbody>
</table>

E  Molecular experiment

In order to have a fair comparison with Flow models, such as GraphAF [55] and GraphDF [37], we incorporate our parameterization into the training process of a Flow model. Both GraphAF and GraphDF are designed using relational GCN [53] (RGCN) as building block to compute an embedding of a graph using a layer R-GCN:

\[ H^L_i = \text{R-GCN}(G_i), \tilde{h}_i = \text{sum}(H^L_i) \]

where sum denotes the sum-pooling operation, and \( H^L_{i,j} \in \mathbb{R}^k \) denotes the embedding of the \( j \)-th node in the embeddings \( H^L_i \).

We incorporate some parts of our models after the RGCN encoding, as follows. From the embedding \( H \), we compute the lie algebra coordinates, with an encoder \( q(T|H) \), following our E\text{group} network from Section 4. We then apply the lie algebra exponential mapping to reconstruct the embedding \( H \) as \( p(H|T) \), following the D\text{group} network from Section 4. Furthermore, we incorporate the hessian penalty as a regularization. In all the three tasks, we follow the experimental setup from GraphAF, we set the model with group-size of 81, and a hessian penalty of 40.

We first perform the random graph generation task, which involves quantifying 4 metrics in percentage, including: valid molecules with resampling (Val), valid molecules without resampling (Res), unique molecules (Uni), and novel molecules not appearing in the training data (Nov). In Table 9, we report results for GCPN, GraphAF, GraphDF and our models. The results indicate that with the inclusion of a lie algebra reparameterization and the hessian penalty, the performance on random generation is enhanced, and the Flow + SID₁ model achieves the best performance on all metrics a part from uniqueness, where it achieves comparable results.

Secondly, we report the results for the property optimization task in Table 10, which show the scores from the top 3 generated molecules for two selected properties (penalized logP and QED). The baselines are the same ones as in the random generation task. We observe that our Flow + SID₁ model is the top performing method on penalized logP for the top 3 scores, and it achieves the top results, together with the other methods, for QED.
Table 9: Random graph generation.

<table>
<thead>
<tr>
<th>Model</th>
<th>ZINC</th>
<th>QM9</th>
<th>MOSES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Val</td>
<td>Res</td>
<td>Uni</td>
</tr>
<tr>
<td>GCPN</td>
<td>100</td>
<td>20</td>
<td>99.97</td>
</tr>
<tr>
<td>GraphAF</td>
<td>100</td>
<td>68</td>
<td>99.1</td>
</tr>
<tr>
<td>GraphDF</td>
<td>100</td>
<td>89.03</td>
<td>99.16</td>
</tr>
<tr>
<td>SID_U</td>
<td>100</td>
<td>78.21</td>
<td>99.12</td>
</tr>
<tr>
<td>SID_C</td>
<td>100</td>
<td>86.02</td>
<td>99.14</td>
</tr>
<tr>
<td>Flow + SID</td>
<td>100</td>
<td>90.12</td>
<td>99.53</td>
</tr>
</tbody>
</table>

Table 10: Property optimization performance.

<table>
<thead>
<tr>
<th></th>
<th>Penalized logP</th>
<th>QED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1st</td>
<td>2nd</td>
</tr>
<tr>
<td>ZINC</td>
<td>4.52</td>
<td>4.3</td>
</tr>
<tr>
<td>GCPN</td>
<td>7.98</td>
<td>7.85</td>
</tr>
<tr>
<td>GraphAF</td>
<td>12.23</td>
<td>11.29</td>
</tr>
<tr>
<td>GraphDF</td>
<td>13.7</td>
<td>13.18</td>
</tr>
<tr>
<td>SID_U</td>
<td>12.56</td>
<td>12.46</td>
</tr>
<tr>
<td>SID_C</td>
<td>12.89</td>
<td>12.82</td>
</tr>
<tr>
<td>Flow + SID_C</td>
<td>13.97</td>
<td>13.35</td>
</tr>
</tbody>
</table>

Finally, we show in Table 11 the results for the constrained optimization task. We report only GraphDF as baseline, which is the top performing model among the baselines. The evaluation involves reporting the mean and standard deviation of metrics including: the largest property improvement (Imp), and similarities (Sim) between them and their corresponding input molecules, as well as the success rate (Suc).

We observe that by combining the disentangled principles developed in our models into a Flow-based model, we are able to further improve the results, and achieve the top performance.

Table 11: Constrained optimization performance on 800 molecules used in GraphAF.

<table>
<thead>
<tr>
<th>δ</th>
<th>Flow + SID_C</th>
<th>GraphDF</th>
<th>SID_C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Imp</td>
<td>Sim</td>
<td>Suc</td>
</tr>
<tr>
<td>0.0</td>
<td>14.28±0.24</td>
<td>0.29±0.10</td>
<td>100</td>
</tr>
<tr>
<td>0.2</td>
<td>12.89±4.33</td>
<td>0.34±0.09</td>
<td>99.93</td>
</tr>
<tr>
<td>0.4</td>
<td>9.32±5.45</td>
<td>0.50±0.03</td>
<td>99.3</td>
</tr>
<tr>
<td>0.6</td>
<td>4.98±6.21</td>
<td>0.67±0.04</td>
<td>97.3</td>
</tr>
</tbody>
</table>