Lift Your Molecules: Molecular Graph Generation in Latent Euclidean Space

Mohamed Amine Ketata [*1] Nicholas Gao [*1] Johanna Sommer [*1] Tom Wollschläger [1] Stephan Günnemann [1]

Abstract

We introduce a new framework for molecular graph generation using 3D molecular generative models. Our Synthetic Coordinate Embedding (SyCo) framework maps molecular graphs to Euclidean point clouds via synthetic conformer coordinates and learns the inverse map using an E(n)-Equivariant Graph Neural Network (EGNN). The induced point cloud-structured latent space is well-suited to apply existing 3D molecular generative models. This approach simplifies the graph generation problem – without relying on molecular fragments nor autoregressive decoding – into a point cloud generation problem followed by node and edge classification tasks. As a concrete implementation of our framework, we develop EDM-SyCo based on the E(3) Equivariant Diffusion Model (EDM). It achieves state-of-the-art performance in distribution learning of molecular graphs, outperforming the best non-autoregressive methods by more than 30% on ZINC250K and 16% on GuacaMol while improving conditional generation by up to 3.9 times.

1. Introduction

Machine learning-based drug discovery has recently shown great potential to accelerate the drug development process while reducing the costs associated with clinical trials (Jiménez-Luna et al., 2020; Kim et al., 2020). Among the steps of this process, generating novel molecules with relevant properties such as drug-likeness and synthesizability, or optimizing existing ones are of crucial importance. Molecular graph generation is an active area of research that aims to solve these tasks (Gómez-Bombarelli et al., 2018; Stokes et al., 2020; Bilodeau et al., 2022).

[*]Equal contribution [1]Department of Computer Science & Munich Data Science Institute, Technical University of Munich, Germany. Correspondence to: Mohamed Amine Ketata <m.ketata@tum.de>.

Accepted as an extended abstract for the Geometry-grounded Representation Learning and Generative Modeling Workshop at the 41st International Conference on Machine Learning, ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

Molecular graph generation is challenging due to the discrete, sparse and symmetric nature of graphs. Many methods have been proposed to tackle this problem, offering different tradeoffs (Jin et al., 2018; Shi et al., 2020). A common approach within these methods is to train a Variational Autoencoder (VAE) to encode the graph into a continuous fixed-size latent space, then decode it back – oftentimes autoregressively – node-by-node (Liu et al., 2018) or fragment-by-fragment (Jin et al., 2018; 2020; Maziarz et al., 2021; Kong et al., 2022). While such methods achieve good generation and optimization performance, they have inherent limitations. First, using a fixed-size latent space does not account for the variable size of molecular graphs ranging from a few atoms for small molecules to thousands for macromolecules. Second, autoregressive decoding requires a concrete generation order (Schneider et al., 2015), which prevents learning permutation-invariant graph distributions. Fueled by recent developments in diffusion models (Ho et al., 2020; Austin et al., 2021), new approaches to graph generation have emerged that overcome these limitations (Jo et al., 2022). Notably, DiGress (Vignac et al., 2022) defines a discrete diffusion process directly on the node and edge attributes and learns to reverse this process to generate graphs all at once, i.e., not autoregressively. However, it fails to accurately estimate the joint distribution of nodes and edges, and discrete optimization remains a challenge.

In this work, we develop SyCo[1], a new framework that combines the benefits of all-at-once generation with the benefits of a continuous latent space for optimization tasks while avoiding the information bottleneck of fixed-size latent spaces. Inspired by the adjacent field of 3D molecule generation, we propose to embed molecular graphs as latent 3D point clouds, implicitly encoding the discrete graph structure in *synthetic* coordinates, and use point cloud generative models to learn their distribution. We leverage this framework to propose EDM-SyCo, an atom-based, all-at-once, and permutation-invariant generative model for molecular graphs based on EDM (Hoogeboom et al., 2022), depicted in Figure 1. Compared to DiGress on ZINC250K, EDM-SyCo improves upon its distribution learning performance by 30% and its conditional generation performance by up to 15.6 times, highlighting the benefit of our continuous latent space on generation and optimization.

[1]Our code is available at https://github.com/ketatam/SyCo.

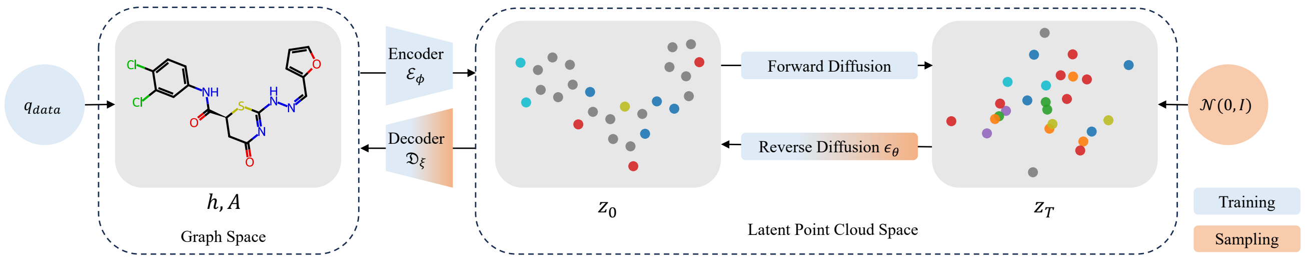

Figure 1. Overview of EDM-SyCo. *(Training)* First, the autoencoder is trained to map between molecular graphs and latent Euclidean point clouds. Then, the diffusion model is trained on the fixed latent space. *(Sampling)* Starting with a Gaussian sample, the diffusion model denoises it for T steps to predict the clean point cloud, which is mapped to a molecular graph using the decoder.

2. Related Work

Molecular Graph Generation Early work on molecule generation (Gómez-Bombarelli et al., 2018; Segler et al., 2018) used language models to generate SMILES (Weininger, 1988), a text-based representation of molecules. However, this type of representation cannot capture the inherent structure of molecules well, which are more accurately depicted as graphs. For instance, small changes in the graph structure may correspond to large differences in the SMILES string (Jin et al., 2018). Therefore, recent methods have focused on graph-based approaches that can be broadly classified into two categories. All-at-once generation methods generate nodes and edges in parallel, though possibly in a multi-step process such as diffusion models (Niu et al., 2020; Jo et al., 2022; Vignac et al., 2022). Autoregressive methods specify a generation order and generate molecules either atom-by-atom (Li et al., 2018; Shi et al., 2020) or fragment-by-fragment based on a pre-computed fragment vocabulary (Jin et al., 2018; 2020; Maziarz et al., 2021; Kong et al., 2022; Geng et al., 2023). EDM-SyCo is an all-at-once molecular graph generative model that operates on the atom level.

Molecule Generation in 3D Another line of work aims to generate molecules in 3D Euclidean space and solve a point cloud generation problem (Gebauer et al., 2019; Luo & Ji, 2022; Garcia Satorras et al., 2021). For instance, EDM (Hoogeboom et al., 2022) and GeoLDM (Xu et al., 2023) are diffusion-based approaches for 3D molecule generation that jointly generate atomic features and coordinates. While tangentially related, these methods solve a different problem from molecular graph generation and require molecules with 3D information for training. Our proposed SyCo framework is an attempt to connect these two lines of work by enabling the training of 3D generative models on graph datasets through synthetic coordinates, formally introducing new graph generation methods. While we are the first to explore synthetic coordinates for generative models, Gasteiger et al. (2021) demonstrated that synthetic coordinates improve molecular property prediction tasks.

Latent Generative Models The idea of defining the generative model on a space different from the original data space is reminiscent of latent generative models (Dai & Wipf, 2019; Vahdat et al., 2021). Recently, this approach has become increasingly popular with latent diffusion models. Stable Diffusion models (Rombach et al., 2022) achieve impressive results on text-guided image generation, and GeoLDM (Xu et al., 2023) extends this to molecular point cloud generative models. While EDM-SyCo operates on a similar latent space to GeoLDM, their fundamental difference lies in the original data space. Our model maps discrete molecular graphs to continuous point clouds and can be seen as a cross-modality latent generative model. In contrast, GeoLDM embeds point clouds as new point clouds with a reduced feature dimension. Further, as explained in the previous paragraph, GeoLDM is a 3D molecule generative model and is not directly applicable to generating graphs.

3. Synthetic Coordinate Embedding (SyCo) Framework

The main idea behind SyCo is to model molecular graph distributions through distributions of Euclidean point clouds that implicitly encode the graph structure into their coordinates. For this, we propose a novel autoencoder architecture that maps between molecular graphs and 3D point cloud representations. By training it using a reconstruction objective, we can reformulate the problem of generating discrete molecular graphs to an equivalent problem of generating 3D point clouds defined on the induced latent space. We can then use any 3D generative model on this latent space.

Notation We introduce helpful notation for the two molecule representations used in this work. Let N denote the number of atoms of a given molecule. On the one hand, the molecular graph $\mathcal{G} = (\boldsymbol{h}, \boldsymbol{A}) \in \mathbb{G}$ consists of atoms as nodes and chemical bonds as edges. Each atom has one of a atom types and one of c formal charges, while each bond has one of b bond types. We represent atoms as stacked one-hot encodings $\boldsymbol{h} \in \{0,1\}^{N\times(a+c)}$ and bonds as one-hot vectors

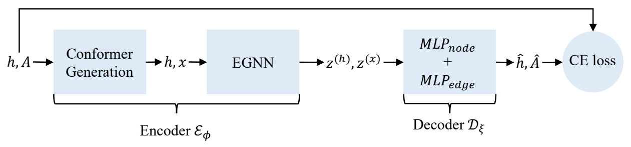

Figure 2. Autoencoder architecture. The encoder maps a molecular graph to a point cloud, and the decoder learns the inverse. Both are trained jointly to minimize the reconstruction loss.

in an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N\times N\times b}$. Note that the molecular graph is sometimes called a "2D molecule" as it can be typically drawn as a planar graph. On the other hand, the molecular point cloud $\mathcal{P} = (\boldsymbol{h}, \boldsymbol{x}) \in \mathbb{P}$ is a 3D point cloud, where $\boldsymbol{x} \in \mathbb{R}^{N\times 3}$ represents the coordinates of atoms' nuclei in Euclidean space. Note that the point cloud encodes the bond information implicitly in the atomic coordinates. We now present our autoencoder model, whose overview is shown in Figure 2.

Encoding The encoder $\mathcal{E}_\phi : \mathbb{G} \to \mathbb{R}^{N\times(d+3)}$ with parameters ϕ maps a molecule's graph representation $\mathcal{G} = (\boldsymbol{h}, \boldsymbol{A})$ to its point cloud representation $\mathcal{P} = (\boldsymbol{h}, \boldsymbol{x})$, then further compresses it into a node-structured latent representation $\boldsymbol{z} = (\boldsymbol{z}^{(h)}, \boldsymbol{z}^{(x)})$, where $\boldsymbol{z}^{(h)} \in \mathbb{R}^{N\times d}$ and $\boldsymbol{z}^{(x)} \in \mathbb{R}^{N\times 3}$ are continuous embeddings for $\boldsymbol{h}$ and $\boldsymbol{x}$ with $d < a + c$ and $\boldsymbol{z} \in \mathbb{R}^{N\times(d+3)}$ is their concatenation.

In the first encoding step, the coordinates $\boldsymbol{x}$ are computed using the ETKDG algorithm (Riniker & Landrum, 2015) available in RDKit (Landrum et al., 2006), which uses distance geometry and torsion angle preferences to generate 3D conformations of a molecule. These coordinates encode the bond information of the molecular graph and ideally define an injective mapping from $\mathbb{G}$ to $\mathbb{P}$. Note that this step is not learned and, thus, not required to be differentiable. In practice, we generate conformers for the whole training set in a preprocessing step and reuse them throughout training.

Then, we apply an EGNN (see Appendix A.2) that maps the intermediate representation $\mathcal{P} = (\boldsymbol{h}, \boldsymbol{x})$ to the latent representation $\boldsymbol{z} = (\boldsymbol{z}^{(h)}, \boldsymbol{z}^{(x)})$, defining the space of the generative model. The main benefit of this is to incorporate the neighborhood information into each atom's embeddings and coordinates while mapping the discrete features $\boldsymbol{h}$ into a continuous and lower-dimensional embedding $\boldsymbol{z}^{(h)}$.

Decoding The decoder $\mathcal{D}_\xi : \mathbb{R}^{N\times(d+3)} \to \mathbb{G}$ with parameters ξ aims to approximate the inverse function $\mathcal{E}_\phi^{-1}$ and maps the latent point cloud $\boldsymbol{z}$ of a molecule back to its graph representation $\mathcal{G}$. It consists of an atom and a bond classifier. For each node i, the decoder applies a 2-layer Multi-Layer Perceptron (MLP) on its latent embedding to predict the logits for the atom type and the formal charge as $\hat{h}_i = \text{MLP}_{node}(z_i^{(h)})$. For each pair of nodes i and j, the bond type is inferred by running a different 2-layer MLP on their edge features to obtain the corresponding logits

Table 1. Graph reconstruction accuracy on ZINC250K.

METHOD	ACCURACY
EGNN + MLP	**99.93%**
MLP	11.77%
RULE-BASED	11.78%

$$\hat{A}_{ij} = \frac{1}{2}\left(\text{MLP}_{edge}\left(\begin{bmatrix} z_i^{(h)} \\ z_j^{(h)} \\ d_{ij}^2 \end{bmatrix}\right) + \text{MLP}_{edge}\left(\begin{bmatrix} z_j^{(h)} \\ z_i^{(h)} \\ d_{ij}^2 \end{bmatrix}\right)\right),$$

where $d_{ij} = \|z_i^{(x)} - z_j^{(x)}\|_2$ and we use the simple averaging trick to ensure that $\hat{\boldsymbol{A}}$ is symmetric.

Autoencoder Training The encoder and decoder are jointly optimized in a variational autoencoder (VAE) framework. We define probabilistic encoding and decoding processes as $q_\phi(\boldsymbol{z}|\mathcal{G}) = \mathcal{N}(\mathcal{E}_\phi(\mathcal{G}), \sigma_0^2\boldsymbol{I})$ and $p_\xi(\mathcal{G}|\boldsymbol{z}) = \prod_{i=1}^N p_\xi(h_i|\boldsymbol{z}) \prod_{j=1}^N p_\xi(A_{ij}|\boldsymbol{z})$, respectively, where $\sigma_0 \in \mathbb{R}$ controls the variance in the latent space. The VAE is trained by minimizing the loss function

$$\mathcal{L}_{VAE} = -\mathbb{E}_{q(\mathcal{G})q_\phi(\boldsymbol{z}|\mathcal{G})}\left[p_\xi(\mathcal{G}|\boldsymbol{z})\right], \tag{1}$$

which, in practice, is computed as the cross-entropy loss. The decoder performs three classification tasks to reconstruct a molecular graph from its point cloud representation: atom types, formal charges, and bond types. The total loss is the sum of the corresponding three cross-entropy terms. Algorithm 1 (First Stage) gives the training details of the autoencoder.

We show an ablation study on the autoencoder architecture. We replace the encoder's EGNN and the decoder with (i) only the decoder and (ii) a rule-based system akin to Hoogeboom et al. (2022). We report the molecule reconstruction accuracy on the validation set, where a molecule is considered correctly reconstructed from its latent representation if all of its atom features (atom types and formal charges) and bond types are correctly classified. Results on ZINC250K are shown in Table 1, and similar results are observed for GuacaMol. Our autoencoder model achieves an almost perfect reconstruction accuracy, highlighting the importance of the EGNN in our architecture. This experiment supports our assumptions (i) that the used mapping from $\mathbb{G}$ to $\mathbb{P}$ is injective and (ii) that the decoder can accurately approximate its inverse.

4. EDM-SyCo

Our SyCo framework lifts the graph generation problem to a point cloud generation problem in 3D. Concretely, we are interested in learning the distribution of latent point clouds $\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\mathcal{G})q(\mathcal{G})$ defined by the underlying molecular graph distribution and the trained encoder by approximating it with a parametric distribution p_θ. While this problem can, in principle, be approached by *any* 3D molecular generative method, this work adapts EDM under the SyCo framework due to its simplicity and empirical performance in 3D molecule generation tasks (Hoogeboom et al., 2022).

Training of EDM-SyCo After training the autoencoder, we train EDM in a second stage on the latent point cloud representations of the molecular graphs from the training set. We give an overview of diffusion models on point clouds in Appendix B.1 and outline the training procedure in Algorithm 1. Consistent with previous work on latent diffusion models (Rombach et al., 2022; Xu et al., 2023), we found this two-stage training procedure to perform better than jointly training the autoencoder and the diffusion model.

Sampling from EDM-SyCo For sampling, we need access to the trained EDM and the decoder but not the encoder. First, we sample the number of atoms N from a categorical distribution fitted on the training set (Hoogeboom et al., 2022). Then, starting with a Gaussian sample $\boldsymbol{z}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ of size N, we iteratively denoise it with the diffusion model:

$$\boldsymbol{z}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{z}_t, t) + \sigma_t \boldsymbol{\epsilon}_t, \tag{2}$$

where $\boldsymbol{\mu}_\theta : \mathbb{R}^{N\times(d+3)} \to \mathbb{R}^{N\times(d+3)}$ is the denoising network, σ_t depends on the variance schedule, and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Finally, we apply the decoder on the clean sample $\boldsymbol{z}_0$ to get the corresponding molecular graph $\boldsymbol{h}, \boldsymbol{A} = \mathcal{D}_\xi(\boldsymbol{z}_0)$. Algorithm 2 formally describes this procedure, and Figure 1 further illustrates it. This sampling procedure defines a permutation-invariant molecular graph distribution (see Appendix A.1). The following proposition, which we prove in Appendix D.2, formalizes this:

Proposition 4.1. *The marginal distribution of molecular graphs $p_{\theta,\xi}(\mathcal{G}) = \mathbb{E}_{p_\theta(\boldsymbol{z}_0)}[p_\xi(\mathcal{G}|\boldsymbol{z}_0)]$ defined by the EDM and the decoder, is an S_N-invariant distribution, i.e. for any molecular graph $\mathcal{G} = (\boldsymbol{h}, \boldsymbol{A})$, $p_{\theta,\xi}(\boldsymbol{h}, \boldsymbol{A}) = p_{\theta,\xi}(P\boldsymbol{h}, P\boldsymbol{A}P)$ for any permutation matrix $P \in S_N$.*

Conditional Generation with EDM-SyCo To sample from EDM-SyCo in a conditional generation setting, we take advantage of our continuous latent space and adapt the classifier guidance algorithm (Dhariwal & Nichol, 2021) to condition on continuous molecular properties using a property regressor and the same generative model described in the previous sections (Bao et al., 2022). The modified sampling procedure reads:

$$\boldsymbol{z}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{z}_t, t) \boxed{-s(t)\nabla_{\boldsymbol{z}_t}(g_\eta(\boldsymbol{z}_t) - c)^2} + \sigma_t \boldsymbol{\epsilon}_t, \tag{3}$$

where $g_\eta : \mathbb{R}^{N\times(d+3)} \to \mathbb{R}$ is the regressor, c is the target property, and $s(t)$ is a time-dependent scaling function. This algorithm, along with a novel procedure for constrained optimization, is described in detail in Appendix B.3.

Table 2. FCD and KL scores on ZINC250K. We split the methods into autoregressive and all-at-once. The **best scores within each category are in bold**, and the <u>best overall are underlined</u>. All models are trained from scratch, and we report mean and standard deviation using 3 random seeds.

	Method	FCD (↑)	KL (↑)
Autoreg.	GraphAF (2020)	0.05 ± 0.00	0.67 ± 0.01
	PS-VAE (2022)	0.28 ± 0.01	0.84 ± 0.00
	HierVAE (2020)	0.50 ± 0.14	0.92 ± 0.00
	MiCaM (2023)	0.63 ± 0.02	0.94 ± 0.00
	JT-VAE (2018)	0.75 ± 0.01	0.93 ± 0.00
	MAGNet (2023)	0.76 ± 0.00	0.95 ± 0.00
	MoLeR (2021)	**0.83 ± 0.00**	<u>**0.97 ± 0.00**</u>
AAO	DiGress (2022)	0.65 ± 0.00	0.91 ± 0.00
	EDM-SyCo (Ours)	<u>**0.85 ± 0.01**</u>	**0.96 ± 0.00**

Limitations

EDM-SyCo can be seen as an extension to EDM, which carries the limitations of diffusion models like high inference cost. To mitigate this issue in future work, we can incorporate more efficient sampling strategies (Karras et al., 2022; 2023), or use more efficient models such as recent conditional flow matching approaches (Tong et al., 2023; Song et al., 2023). In addition, our sampling approach requires specifying a priori the number of nodes. While that works well empirically, we leave tackling the fundamental issue of generating variable-sized graphs or point clouds to future work.

5. Experiments

In our experiments, we compare the performance of EDM-SyCo to several autoregressive baselines and to all-at-once diffusion-based model DiGress (Vignac et al., 2022) on de novo and conditional generation tasks. Specifically, comparing the two diffusion-based models, EDM-SyCo and DiGress, highlights the effect of our SyCo framework on generation and optimization performance. We use two datasets of different sizes and complexities: ZINC250K (Irwin et al., 2012) containing 250K molecules with up to 40 atoms, and GuacaMol (Brown et al., 2019) containing ≈1.5M drug-like molecules with up to 88 atoms. We train EDM-SyCo as described in Sections 3 and 4 on these two datasets. Training details and hyperparameters are given in Appendix E.

De novo Generation We leverage the GuacaMol benchmark (Brown et al., 2019), an evaluation framework for de novo molecular graph generation. We consider two metrics that measure the similarity between the generated molecules and the training set. The Fréchet ChemNet Distance (FCD)

Table 3. FCD and KL scores on GuacaMol. * means numbers are from (Maziarz et al., 2021), and DiGress is from Vignac et al. (2022). For our model, we additionally report standard deviation across 3 validations of the same model.

		FCD (↑)	KL (↑)
AUTOREG.	CGVAE* (2018)	0.26	-
	HIERVAE* (2020)	0.62	-
	JT-VAE* (2018)	0.73	-
	MOLER* (2021)	<u>**0.81**</u>	-
AAO	DIGRESS (2022)	0.68	<u>**0.93**</u>
	EDM-SYCO (OURS)	**0.79** ± **0.002**	<u>**0.93** ± **0.006**</u>

compares the hidden representations of ChemNet, a neural network capturing chemical and biological features of molecules (Preuer et al., 2018), and the KL divergence (KL) compares the probability distributions of a variety of physiochemical descriptors.

Tables 2 and 3 show the FCD and KL scores of EDM-SYCO and the baselines on the ZINC250K and GuacaMol datasets, respectively. Notably, EDM-SYCO outperforms DiGress on the FCD metric by 30% on ZINC250K and 16% on GuacaMol. This shows the capacity of our model to more accurately capture the joint distribution of nodes and edges encoded in the latent features and coordinates. EDM-SYCO sets the new state-of-the-art FCD score on ZINC250K, outperforming all autoregressive and fragment-based baselines, which incorporate more domain knowledge. Note that we use the same hyperparameters from ZINC250K to train our model on GuacaMol, while DiGress and MoLeR have been tuned for both datasets. We do not report novelty and uniqueness scores because all methods achieve near-perfect results on these metrics. Regarding validity, all autoregressive baselines achieve 100% validity scores due to valency checks after each intermediate step. DiGress achieves 85%, and EDM-SYCO achieves 88%. We show some sample molecules generated by our model in Figures 3 and 4 for the two datasets.

Property-Conditioned Generation To evaluate the conditional generation performance of our approach, we follow Ninniri et al. (2023) and condition the generation of 1000 molecules on LogP and molecular weight (MW) values sampled from ZINC250K. We report the mean absolute error (MAE) between the target and estimated values for the generated molecules (using RDKit) and the validity rate. We run the regressor guidance algorithm, discussed in Section B.3, by using EDM-SYCO from the previous experiment and a single regressor trained to predict LogP and MW values. As baselines, we compare to DiGress, which uses a discrete guidance scheme, and FreeGress (Ninniri et al., 2023), which proposes a regressor-free guidane mechanism. Table 4 shows the results. While EDM-SYCO achieves a comparable score to FreeGress on LogP (within standard deviation), it outperforms FreeGress by a factor of 3.9 on MW. Compared to DiGress, it improves on LogP and MW by factors 2.7 and 15.6, respectively. This experiment highlights the benefit of our continuous latent space on conditional generation compared to operating directly on the discrete graph space. It also shows that graph-level properties are represented sufficiently well in this space.

Table 4. Conditional generation results.

METHOD	LOGP		MW	
	MAE (↓)	VALID (↑)	MAE (↓)	VALID (↑)
DIGRESS	0.49 ± 0.05	65%	60.30 ± 6.10	73%
FREEGRESS	**0.15** ± **0.02**	**85%**	15.18 ± 2.71	75%
EDM-SYCO	0.18 ± 0.01	76%	**3.86** ± **0.08**	**88%**

6. Conclusion

Despite the common practice of using fragment-based autoregressive models for molecular graph generation, we demonstrated with SYCO that a mapping between molecular graphs and latent Euclidean point clouds enables atom-based all-at-once approaches to be competitive with special-purpose graph generators. Based on this framework, we developed EDM-SYCO, which sets a new state-of-the-art FCD score on ZINC250K. In our conditional generation experiments, we found our guidance-based approach to accurately guide the generation process, outperforming all baselines. With these results, we conclude that latent Euclidean generative models hold significant promise for advancing molecular graph generation and accelerating drug discovery.

Broader Impact

This paper advances the field of computational drug discovery by proposing new efficient methods for generative models. While there may be unintended consequences, such as the misuse of chemical weapons, we firmly believe that their benefits significantly outweigh the minuscule chance of misuse.

Acknowledgements

This project is supported by the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research. Further, it is funded by the Federal Ministry of Education and Research (BMBF) and the Free State of Bavaria under the Excellence Strategy of the Federal Government and the Länder. This project is also supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy with funds from the Hightech Agenda Bayern

References

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

Bao, F., Zhao, M., Hao, Z., Li, P., Li, C., and Zhu, J. Equivariant energy-guided sde for inverse molecular design. *arXiv preprint arXiv:2209.15408*, 2022.

Bilodeau, C., Jin, W., Jaakkola, T., Barzilay, R., and Jensen, K. F. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 12(5):e1608, 2022.

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

Dai, B. and Wipf, D. Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*, 2019.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Garcia Satorras, V., Hoogeboom, E., Fuchs, F., Posner, I., and Welling, M. E (n) equivariant normalizing flows. *Advances in Neural Information Processing Systems*, 34: 4181–4192, 2021.

Gasteiger, J., Yeshwanth, C., and Günnemann, S. Directional message passing on molecular graphs via synthetic coordinates. *Advances in Neural Information Processing Systems*, 34:15421–15433, 2021.

Gebauer, N., Gastegger, M., and Schütt, K. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. *Advances in neural information processing systems*, 32, 2019.

Geng, Z., Xie, S., Xia, Y., Wu, L., Qin, T., Wang, J., Zhang, Y., Wu, F., and Liu, T.-Y. De novo molecular generation via connection-aware motif mining. *arXiv preprint arXiv:2302.01129*, 2023.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Hetzel, L., Sommer, J., Rieck, B., Theis, F., and Günnemann, S. Magnet: Motif-agnostic generation of molecules from shapes. *arXiv preprint arXiv:2305.19303*, 2023.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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

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

Jiménez-Luna, J., Grisoni, F., and Schneider, G. Drug discovery with explainable artificial intelligence. *Nature Machine Intelligence*, 2(10):573–584, 2020.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Jin, W., Barzilay, R., and Jaakkola, T. Hierarchical generation of molecular graphs using structural motifs. In *International conference on machine learning*, pp. 4839–4848. PMLR, 2020.

Jo, J., Lee, S., and Hwang, S. J. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, pp. 10362–10383. PMLR, 2022.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577, 2022.

Karras, T., Aittala, M., Lehtinen, J., Hellsten, J., Aila, T., and Laine, S. Analyzing and improving the training dynamics of diffusion models. *arXiv preprint arXiv:2312.02696*, 2023.

Kim, H., Kim, E., Lee, I., Bae, B., Park, M., and Nam, H. Artificial intelligence in drug discovery: a comprehensive review of data-driven and machine learning approaches. *Biotechnology and Bioprocess Engineering*, 25:895–930, 2020.

King, G. and Zeng, L. Logistic regression in rare events data. *Political analysis*, 9(2):137–163, 2001.

Kong, X., Huang, W., Tan, Z., and Liu, Y. Molecule generation by principal subgraph mining and assembling. *Advances in Neural Information Processing Systems*, 35: 2550–2563, 2022.

Landrum, G. et al. Rdkit: Open-source cheminformatics, 2006.

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

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

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11461–11471, 2022.

Luo, Y. and Ji, S. An autoregressive flow model for 3d molecular geometry generation from scratch. In *International Conference on Learning Representations (ICLR)*, 2022.

Maziarz, K., Jackson-Flux, H., Cameron, P., Sirockin, F., Schneider, N., Stiefl, N., Segler, M., and Brockschmidt, M. Learning to extend molecular scaffolds with structural motifs. *arXiv preprint arXiv:2103.03864*, 2021.

Mishchenko, K. and Defazio, A. Prodigy: An expeditiously adaptive parameter-free learner. *arXiv preprint arXiv:2306.06101*, 2023.

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

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

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

Riniker, S. and Landrum, G. A. Better informed distance geometry: using what we know to improve conformation generation. *Journal of chemical information and modeling*, 55(12):2562–2574, 2015.

Rogers, D. and Hahn, M. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50 (5):742–754, 2010.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Schneider, N., Sayle, R. A., and Landrum, G. A. Get your atoms in order an open-source implementation of a novel and robust molecular canonicalization algorithm. *Journal of chemical information and modeling*, 55(10):2111–2120, 2015.

Schneuing, A., Du, Y., Harris, C., Jamasb, A., Igashov, I., Du, W., Blundell, T., Lió, P., Gomes, C., Welling, M., et al. Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695*, 2022.

Schuffenhauer, A., Ertl, P., Roggo, S., Wetzel, S., Koch, M. A., and Waldmann, H. The scaffold tree- visualization of the scaffold universe by hierarchical scaffold classification. *Journal of chemical information and modeling*, 47(1):47–58, 2007.

Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4 (1):120–131, 2018.

Serre, J.-P. et al. *Linear representations of finite groups*, volume 42. Springer, 1977.

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

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

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

Song, Y., Gong, J., Xu, M., Cao, Z., Lan, Y., Ermon, S., Zhou, H., and Ma, W.-Y. Equivariant flow matching with hybrid probability transport for 3d molecule generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., Bloom-Ackermann, Z., et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4): 688–702, 2020.

Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., and Bengio, Y. Conditional flow matching: Simulation-free dynamic optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. arxiv, 2021.

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

Weininger, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Xu, M., Yu, L., Song, Y., Shi, C., Ermon, S., and Tang, J. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

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

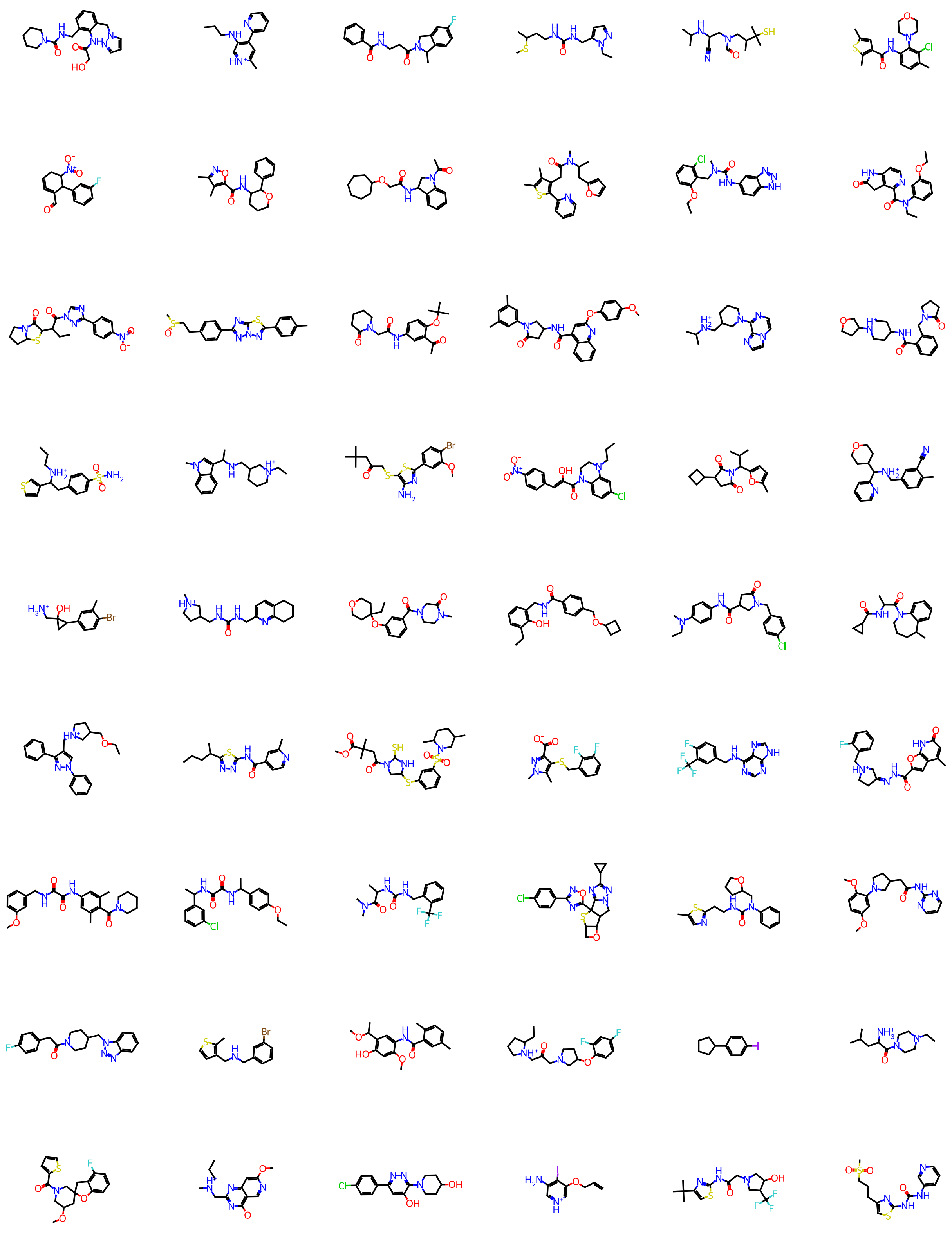

Figure 3. Sample molecules generated by EDM-SyCo trained on ZINC250K.

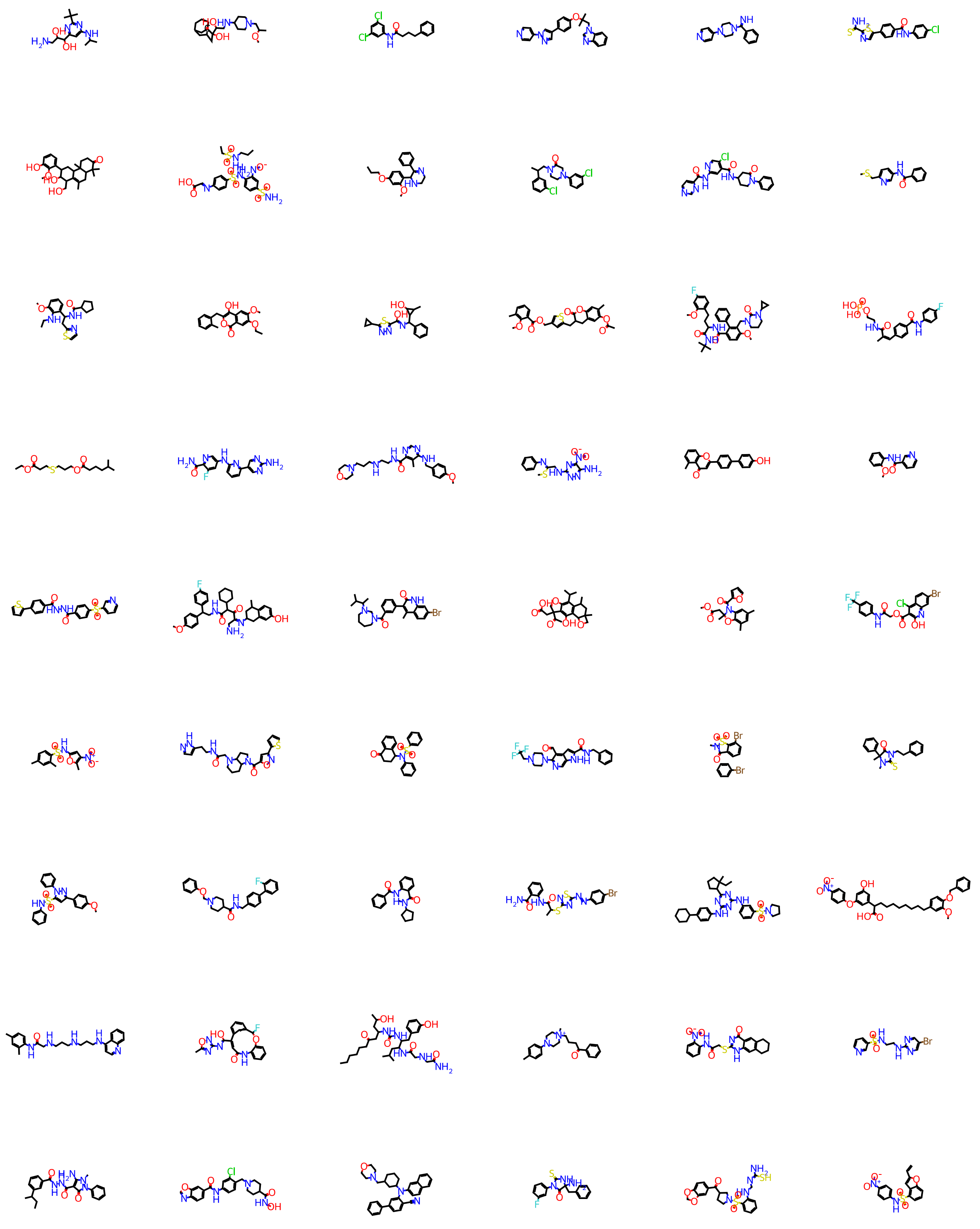

Figure 4. Sample molecules generated by EDM-SyCo trained on GuacaMol.

A. Background

A.1. Equivariance and Invariance

When dealing with physical objects like molecules, it is important to consider *equivariance*. Formally, a function $f : A \to B$ is *equivariant* to the action of a group G iff $f(T_g^A(\boldsymbol{z})) = T_g^B(f(\boldsymbol{z}))$ for all $g \in G$, where T_g^A and T_g^B are the representations of the group element g in A and B respectively (Serre et al., 1977). As a special case, it is *invariant* iff $f(T_g^A(\boldsymbol{z})) = f(\boldsymbol{z})$ with T_g^B being the identity map. In the context of generative modeling, we aim to learn distributions invariant to the data's symmetries, namely graph permutations in our case. In practice, this has the benefit of not relying on any generation order and enabling efficient likelihood estimation since an invariant likelihood is identical for all permutations of the same graph. We leverage equivariant architectures, namely EGNNs, to learn such distributions, removing the need for data augmentation during training, thus further increasing efficiency. EGNNs are defined in the next section.

A.2. $E(n)$ Equivariant Graph Neural Networks (EGNNs)

To perform equivariant updates in our latent space, we use the $E(n)$ Equivariant Graph Neural Network (EGNN) architecture (Satorras et al., 2021). EGNNs (Satorras et al., 2021) are a special type of graph neural networks that learn functions equivariant to rotations, translations, reflections, and permutations of their input point cloud.

An EGNN operates on a point cloud of size N, where each point has features $h_i \in \mathbb{R}^d$ and coordinates $x_i \in \mathbb{R}^3$ associated with it. Let $\boldsymbol{h} \in \mathbb{R}^{N\times d}$ and $\boldsymbol{x} \in \mathbb{R}^{N\times 3}$ be the stacked features and coordinates, respectively, of all points. An EGNN is composed of Equivariant Graph Convolutional Layers (EGCLs) $\boldsymbol{h}^{l+1}, \boldsymbol{x}^{l+1} = \text{EGCL}(\boldsymbol{h}^l, \boldsymbol{x}^l)$. A single EGCL layer is defined as

$$m_{ij} = \phi_e(h_i^l, h_j^l, d_{ij}^2, a_{ij}) \tag{4}$$

$$h_i^{l+1} = \phi_h(h_i^l, \sum_{j\neq i} \tilde{e}_{ij} m_{ij}) \tag{5}$$

$$x_i^{l+1} = x_i^l + \sum_{j\neq i} \frac{x_i^l - x_j^l}{d_{ij}+1} \phi_x(h_i^l, h_j^l, d_{ij}^2, a_{ij}), \tag{6}$$

where l denotes the layer index, $d_{ij} = \|x_i^l - x_j^l\|_2$ is the Euclidean distance between points i and j, and a_{ij} are optional edge attributes which we set to $\|x_i^0 - x_j^0\|_2^2$. $\tilde{e}_{ij}$ serves as an attention mechanism that infers a soft estimation of the edges $\tilde{e}_{ij} = \phi_{inf}(m_{ij})$. Note that to update each point's features and coordinates, the EGCL layers consider all the other nodes, effectively treating the point cloud as a fully connected graph. All learnable components (ϕ_e, ϕ_h, ϕ_x, and ϕ_{inf}) are fully connected neural networks. An EGNN architecture is then composed of L EGCL layers, denoted as $\boldsymbol{h}^L, \boldsymbol{x}^L = \text{EGNN}(\boldsymbol{h}^0, \boldsymbol{x}^0)$ with $\boldsymbol{h}^0 = \boldsymbol{h}$ and $\boldsymbol{x}^0 = \boldsymbol{x}$. The main hyperparameters of the EGNN architectures are the number of layers L and a feature dimension n_f used to control the width of the fully connected neural networks.

B. Method Details

B.1. Equivariant Diffusion on Latent Point Clouds

A diffusion model generates samples by reversing a diffusion process, the process of adding noise to data (Sohl-Dickstein et al., 2015; Ho et al., 2020). For ease of notation, we flatten the latent representation of the point clouds $\boldsymbol{z}$, namely their latent coordinates and features, into a one-dimensional vector denoted as $\boldsymbol{z}_0 \in \mathbb{R}^{d'}$ with $d' = N(d+3)$. The *diffusion* or *forward process* is a fixed Markov chain of Gaussian updates that transform $\boldsymbol{z}_0$ into latent variables $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T$ of the same dimension as $\boldsymbol{z}_0$, defined as

$$q(\boldsymbol{z}_{1:T}|\boldsymbol{z}_0) := \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}), \text{ with } q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) := \mathcal{N}(\boldsymbol{z}_t; \sqrt{1-\beta_t}\boldsymbol{z}_{t-1}, \beta_t \boldsymbol{I}), \tag{7}$$

where β_t is typically chosen such that $q(\boldsymbol{z}_T)$ converges to $\mathcal{N}(\boldsymbol{z}_T; \boldsymbol{0}, \boldsymbol{I})$. By defining $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{i=1}^{t} \alpha_i$, we can also map $\boldsymbol{z}_0$ directly to $\boldsymbol{z}_t$ through

$$q(\boldsymbol{z}_t|\boldsymbol{z}_0) = \mathcal{N}(\boldsymbol{z}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{z}_0, (1-\bar{\alpha}_t)\boldsymbol{I}). \tag{8}$$

To reverse the above process, we start from a Gaussian noise sample $\boldsymbol{z}_T$ and iteratively sample from the posterior distributions $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$, which is also Gaussian if β_t is small (Sohl-Dickstein et al., 2015). However, estimating them requires using the entire dataset. Therefore, diffusion models learn a *reverse process* $p_\theta(\boldsymbol{z}_{0:T}) \coloneqq p(\boldsymbol{z}_T)\prod_{t=1}^T p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$, with $p(\boldsymbol{z}_T) = \mathcal{N}(\boldsymbol{z}_T; \mathbf{0}, \boldsymbol{I})$ and

$$p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) \coloneqq \mathcal{N}(\boldsymbol{z}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{z}_t, t), \sigma_t^2 \boldsymbol{I}), \tag{9}$$

where the mean $\boldsymbol{\mu}_\theta(\boldsymbol{z}_t, t) \in \mathbb{R}^{d'}$ is parametrized by a neural network.

We follow Ho et al. (2020) and, instead of directly predicting the mean, predict the Gaussian noise added to the latent point cloud representation during the diffusion process. The mean then becomes $\boldsymbol{\mu}_\theta(\boldsymbol{z}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)\right)$, where $\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t) \in \mathbb{R}^{d'}$ is the output of the neural network. Following Hoogeboom et al. (2022), we parametrize $\boldsymbol{\epsilon}_\theta$ via an EGNN (Satorras et al., 2021). Additionally, the reverse process ensures equivariance to translations by defining the learned distribution $p_\theta(\boldsymbol{z}_0)$ on the zero center of gravity subspace.

This network can then be trained by optimizing the L_2 loss between the sampled Gaussian noise at each step and the network's output,

$$\mathcal{L}_{DM} = \mathbb{E}_{\boldsymbol{z}_0, \boldsymbol{\epsilon}, t}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)\|_2^2\right]. \tag{10}$$

During sampling, we start from a Gaussian sample $\boldsymbol{z}_T \sim p(\boldsymbol{z}_T)$ and iteratively denoise it by sampling $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and transforming z_t to z_{t-1}:

$$\boldsymbol{z}_{t-1} = \underbrace{\frac{1}{\sqrt{\alpha_t}}(\boldsymbol{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t))}_{\boldsymbol{\mu}_\theta(\boldsymbol{z}_t, t)} + \underbrace{\frac{\sqrt{1-\bar{\alpha}_{t-1}}}{\sqrt{1-\bar{\alpha}_t}}\sqrt{\beta_t}}_{\sigma_t}\,\boldsymbol{\epsilon}_t. \tag{11}$$

B.2. Training and Sampling from EDM-SyCo

Here, we describe in more detail the training and sampling algorithms for EDM-SyCo in Algorithms 1 and 2.

Algorithm 1 Training Algorithm

```
Input: a dataset of molecular graphs $\mathcal{G} = (\boldsymbol{h}, \boldsymbol{A})$
Initial: Encoder network $\mathcal{E}_\phi$, Decoder network $\mathcal{D}_\xi$, denoising network $\boldsymbol{\epsilon}_\theta$
First Stage: Autoencoder Training
repeat
    $\boldsymbol{\mu}_{\boldsymbol{z}} \leftarrow \mathcal{E}_\phi(\boldsymbol{h}, \boldsymbol{A})$ {Encoder}
    $\boldsymbol{\epsilon}_{\boldsymbol{z}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and subtract center of mass from $\boldsymbol{\epsilon}_{\boldsymbol{z}}^{(x)}$
    $\boldsymbol{z} \leftarrow \boldsymbol{\mu}_{\boldsymbol{z}} + \sigma_0 \boldsymbol{\epsilon}_{\boldsymbol{z}}$
    $\hat{\boldsymbol{h}}, \hat{\boldsymbol{A}} \leftarrow \mathcal{D}_\xi(\boldsymbol{z})$
    $\mathcal{L}_{VAE}(\phi, \xi) \leftarrow$ cross-entropy $\left(\begin{bmatrix}\hat{\mathbf{h}} \\ \hat{\mathbf{A}}\end{bmatrix}, \begin{bmatrix}\mathbf{h} \\ \mathbf{A}\end{bmatrix}\right)$
    Take an optimizer step on $\mathcal{L}_{AE}(\phi, \xi)$
until $\phi$ and $\xi$ have converged
Second Stage: Diffusion Model Training
Fix Autoencoder parameters $\phi$ and $\xi$
repeat
    $\boldsymbol{z}_0 \sim q_\phi(\boldsymbol{z}|\boldsymbol{h}, \boldsymbol{A})$
    $\boldsymbol{\epsilon}_{\boldsymbol{z}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and subtract center of mass from $\boldsymbol{\epsilon}_{\boldsymbol{z}}^{(x)}$
    $t \sim \mathcal{U}(0, 1, \ldots, T)$
    $\boldsymbol{z}_t \leftarrow \sqrt{\bar{\alpha}_t}\boldsymbol{z}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_{\boldsymbol{z}}$
    $\mathcal{L}_{DM}(\theta) \leftarrow \|\boldsymbol{\epsilon}_{\boldsymbol{z}} - \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)\|^2$
    Take an optimizer step on $\mathcal{L}_{DM}(\theta)$
until $\theta$ has converged
return $\mathcal{E}_\phi$, $\mathcal{D}_\xi$, $\boldsymbol{\epsilon}_\theta$
```

Algorithm 2 Sampling Algorithm

```
Input: Decoder network $\mathcal{D}_\xi$, denoising network $\boldsymbol{\epsilon}_\theta$
$\boldsymbol{z}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and subtract center of mass from $\boldsymbol{z}_T^{(x)}$
for $t = T$ downto $1$ do
    $\boldsymbol{\epsilon}_{\boldsymbol{z}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and subtract center of mass from $\boldsymbol{\epsilon}_{\boldsymbol{z}}^{(x)}$
    $\boldsymbol{z}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)\right) + \sigma_t \boldsymbol{\epsilon}_{\boldsymbol{z}}$
end for
$\boldsymbol{h}, \boldsymbol{A} \leftarrow \mathcal{D}_\xi(\boldsymbol{z}_0)$
return 2D molecule $\mathcal{G} = (\boldsymbol{h}, \boldsymbol{A})$
```

B.3. Controllable Generation and Similarity-constrained Optimization

In the main text, we have introduced our approach to distribution learning on molecular graphs. However, many tasks in drug discovery require generating molecules with specific conditions such as chemical properties, similarity constraints, or the presence of certain desirable substructures. In this section, we describe how we can adapt the sampling procedure of EDM-SyCo (or other models combined with SyCo) – trained in an unconditional setting – to different conditioning modalities. While previous works on 3D molecule generative models approached conditional generation (Hoogeboom et al., 2022; Bao et al., 2022) and scaffolding (Schneuing et al., 2022), the common graph generation task of similarity-constrained optimization (Jin et al., 2020) was absent. We fill this gap by introducing a novel sampling algorithm for diffusion models to perform similarity-constrained optimization.

Property-Conditional Generation via Diffusion Guidance In a conditional generation setting, we want to generate a molecule with a specific property c by sampling from the conditional distribution $q(\boldsymbol{z}|c)$. To achieve this, we adapt the classifier guidance algorithm (Dhariwal & Nichol, 2021), initially proposed to guide the generation of an image diffusion model using a classifier. Instead, we use a property regressor to guide the generation towards molecules with specific continuous properties similar to Bao et al. (2022). One can show that the denoising network's output relates to the score function of the data distribution via $\nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t) \approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)$ (Dhariwal & Nichol, 2021) and the sampling procedure of diffusion models (Equation 11) is equivalent to running Langevin dynamics using the score function $\nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t)$ (Welling & Teh, 2011; Song & Ermon, 2019). This provides a way to sample from the distribution q only through its score function. For the case of $q(\boldsymbol{z}_t|c)$, its score function can be written using Bayes' rule and approximated as:

$$\begin{aligned}\nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t|c) &= \nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t) + \nabla_{\boldsymbol{z}_t} \log q(c|\boldsymbol{z}_t)\\ &\approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}}(\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t) - \sqrt{1-\bar{\alpha}_t}\nabla_{\boldsymbol{z}_t} \log p_\eta(c|\boldsymbol{z}_t)),\end{aligned}$$

where $p_\eta(c|\boldsymbol{z}_t) \sim \exp(-s(g_\eta(\boldsymbol{z}_t) - c)^2)$ is a Gaussian distribution approximating the probability of molecule $\boldsymbol{z}_t$ having property c, g_η is a property regressor trained to predict the property c given a noisy molecule $\boldsymbol{z}_t$, and s is a scale factor that controls the skewness of the distribution. With this, we can rewrite the conditional score function as $\nabla_{\boldsymbol{z}_t} \log q(\boldsymbol{z}_t|c) \approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}}(\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t) + \sqrt{1-\bar{\alpha}_t}s\nabla_{\boldsymbol{z}_t}(g_\eta(\boldsymbol{z}_t) - c)^2)$. To sample from this distribution, we use the same sampling algorithm defined by Equation 11 and replace $\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)$ by

$$\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t) + \sqrt{1-\bar{\alpha}_t}s\nabla_{\boldsymbol{z}_t}(g_\eta(\boldsymbol{z}_t) - c)^2. \tag{12}$$

Intuitively, this can be seen as minimizing the loss $(g_\eta(\boldsymbol{z}_t) - c)^2$.

Unconstrained Property Optimization via Diffusion Guidance With the formulation introduced above, we can sample molecules with high property values by simply using the denoising step

$$\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t) - \sqrt{1-\bar{\alpha}_t}s\nabla_{\boldsymbol{z}_t}g_\eta(\boldsymbol{z}_t). \tag{13}$$

Notice that we replaced the loss with the negative gradient of the regressor.

Scaffold-Constrained Generation via Inpainting It is often desirable to generate molecules that contain a specific substructure, usually called a *scaffold* (Schuffenhauer et al., 2007; Maziarz et al., 2021), for tasks such as structure-based

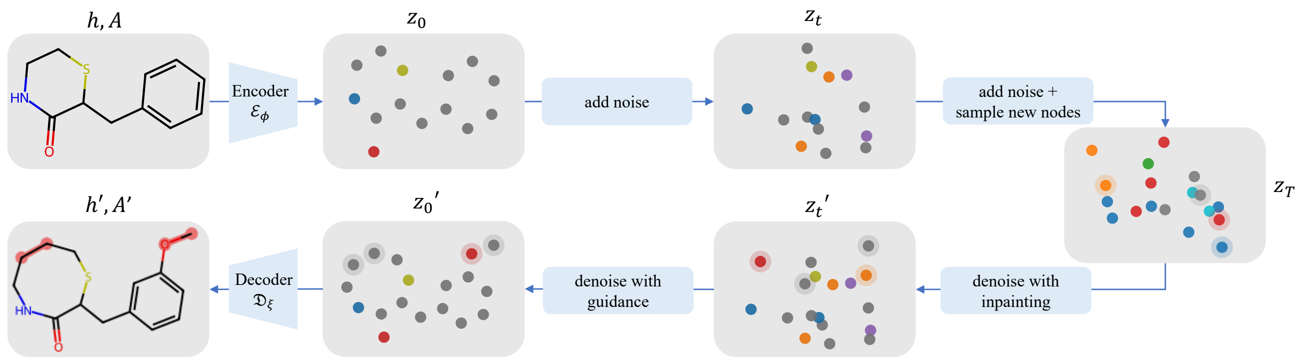

Figure 5. Overview of our constrained optimization procedure. Based on a noising/denoising approach, we run the first steps of the reverse diffusion process using the inpainting algorithm to add new atoms and the remaining steps using the guidance algorithm to increase the target property. The depicted molecules have QED values of 0.79 (initial) and 0.91 (optimized), with a 53% similarity.

drug design. This problem can be viewed as a completion task and is reminiscent of inpainting, which aims to complete missing parts of an image (Song et al., 2020; Lugmayr et al., 2022). Lugmayr et al., 2022 propose RePaint, a simple modification to the sampling procedure of denoising diffusion models, enabling them to perform inpainting. This approach has already been successfully applied to molecule generation in 3D (Schneuing et al., 2022).

The core idea of RePaint is to start the sampling procedure from a random sample $\boldsymbol{z}_T$ and, at each step, enforce the part that we want to be present in the final sample. To illustrate this for the case of molecules, let $\tilde{\boldsymbol{z}}_0$ denote the point cloud of a molecule containing the desired scaffold. Let m be a binary mask that indicates which nodes from $\tilde{\boldsymbol{z}}_0$ belong to the scaffold such that $m \odot \tilde{\boldsymbol{z}}_0$ denotes the scaffold point cloud and $(1-m) \odot \tilde{\boldsymbol{z}}_0$ the unknown part. We aim to sample a new point cloud $\boldsymbol{z}$ containing this scaffold while extending it to get a harmonized point cloud. Formally, we require that $m \odot \boldsymbol{z}_0 = m \odot \tilde{\boldsymbol{z}}_0$. To achieve this, we iteratively apply the following modified sampling step starting from $t = T$:

$$\boldsymbol{z}_{t-1}^{\text{scaffold}} \sim \mathcal{N}(\sqrt{\bar{\alpha}_{t-1}}\tilde{\boldsymbol{z}}_0, (1-\bar{\alpha}_{t-1})\boldsymbol{I}) \tag{14}$$

$$\boldsymbol{z}_{t-1}^{\text{unknown}} \sim \mathcal{N}(\boldsymbol{\mu}_\theta(\boldsymbol{z}_t, t), \sigma_t^2\boldsymbol{I}) \quad \text{(as in Equation 11)} \tag{15}$$

$$\boldsymbol{z}_{t-1} = m \odot \boldsymbol{z}_{t-1}^{\text{scaffold}} + (1-m) \odot \boldsymbol{z}_{t-1}^{\text{unknown}} \tag{16}$$

Similarity-Constrained Optimization In this task, we aim to improve the target properties of a given molecule while satisfying a similarity constraint. We propose a novel approach to this task by combining the regressor guidance algorithm and the inpainting algorithm discussed above. The key idea is to add noise to the initial molecule and then denoise it with the regressor guidance algorithm to improve the target property. Formally, starting from the initial point cloud $\boldsymbol{z}_0$ with N atoms, we sample a noised $\boldsymbol{z}_t$ following Equation 8 for $t \in (0, T)$. t is a hyperparameter that controls the optimization quality. On the one hand, increasing t results in more reverse diffusion steps with the guidance algorithm, thus increasing the chance of improving the target property. On the other hand, it also results in losing information about the initial molecule due to noise. Intuitively, the best t yields the highest property improvement while satisfying the similarity constraints.

However, since the diffusion model operates on N points, the denoised molecule has N atoms. This limits the optimization of a molecule's property as it prevents adding new atoms. To overcome this limitation, we sample a Gaussian point cloud $\boldsymbol{z}_T$ with N' atoms, where $N' > N$, and run the inpainting algorithm from T to t with $\boldsymbol{z}_t$ as the scaffold, meaning that $\boldsymbol{z}_t$ will replace $m \odot \tilde{\boldsymbol{z}}_0$ in the inpainting algorithm described above. This results in a new intermediate representation $\boldsymbol{z}_t'$ whose first N points correspond to $\boldsymbol{z}_t$, and the rest are the newly added atoms, as desired. Finally, $\boldsymbol{z}_t'$ is denoised from t to 0 using the guidance algorithm with maximization objective (Equation 13) to get the point cloud $\boldsymbol{z}_0'$ of the optimized molecule, which is then decoded to the graph using the decoder $\mathcal{D}_\xi$. An overview of this procedure and an example test molecule are shown in Figure 5.

Experiments We follow Jin et al. (2020) and evaluate our optimization procedure from Section B.3 with EDM-SyCo on their constrained optimization tasks: LogP, QED, and DRD2 on ZINC250K. These tasks consist of improving the properties of a set of 800 test molecules under the constraint that the Tanimoto similarity with Morgan fingerprints (Rogers & Hahn, 2010) between the optimized and initial molecule is above a given threshold. We report the average improvement for the

Table 5. Similarity-constrained optimization results from (Jin et al., 2020). We split all learning-based baslines into translation and optimization approaches.

	Method	LogP (sim ≥ 0.4)		LogP (sim ≥ 0.6)		QED (sim ≥ 0.4)		DRD2 (sim ≥ 0.4)	
		Improv.	Div.	Improv.	Div.	Succ.	Div.	Succ.	Div.
Transl.	Seq2Seq	3.37 $\pm$ 1.75	0.471	2.33 $\pm$ 1.17	0.331	58.5%	0.331	75.9%	0.176
	JTNN	3.55 $\pm$ 1.67	0.480	2.33 $\pm$ 1.24	0.333	59.9%	0.373	77.8%	0.156
	AtomG2G	**3.98** $\pm$ **1.54**	0.563	2.41 $\pm$ 1.19	0.379	73.6%	0.421	75.8%	0.128
	HierG2G	**3.98** $\pm$ **1.46**	**0.564**	**2.49** $\pm$ **1.09**	**0.381**	**76.9%**	**0.477**	**85.9%**	**0.192**
Optim.	JT-VAE	1.03 $\pm$ 1.39	-	0.28 $\pm$ 0.79	-	8.8%	-	3.4%	-
	CG-VAE	0.61 $\pm$ 1.09	-	0.25 $\pm$ 0.74	-	4.8%	-	2.3%	-
	GCPN	2.49 $\pm$ 1.30	-	0.79 $\pm$ 0.63	-	9.4%	**0.216**	4.4%	**0.152**
	EDM-SyCo	**3.11** $\pm$ **1.27**	0.555	**1.51** $\pm$ **1.10**	0.360	**46.4%**	0.163	**27.3%**	0.083

LogP tasks and success rates for the binary tasks of QED and DRD2 consisting of translating molecules from a low range into a higher range. Additionally, we report the average diversity among the optimized molecules.

We split the learning-based baselines from (Jin et al., 2020) into optimization and translation approaches. Translation approaches are specifically designed for such tasks and are trained on pairs of molecules exhibiting the improvement and similarity constraints, giving them an advantage over optimization methods. Since we leverage the same generative model and regressor from previous experiments without further training on this task, optimization approaches are the main point of comparison. Table 5 shows that EDM-SyCo outperforms all optimization baselines by an average factor of 3.5 across all tasks while achieving comparable performance to the translation approaches. These results further support the effectiveness of our SyCo framework and our novel constrained optimization algorithm. The relatively low diversity scores indicate that EDM-SyCo tends to find more similar molecules for a given starting molecule, which might be an artifact of our noising/denoising procedure. Figures 6 and 7 illustrate some of EDM-SyCo's optimized molecules. Thanks to our atom-based approach, the optimized molecules frequently only change in a few atoms from the starting molecule, which would be harder to achieve for fragment-based models.

C. Effect of Conformer Generation Method on Generation Performance

To analyze the effect of different conformer generation algorithms on the performance of our model, we perform an ablation study using 2 different algorithms/packages from RDKit to compute the synthetic coordinates: the ETKDG algorithm and the Pharm3D package. We perform two training runs using the same setup except for the conformer generation method. For these two runs, we use smaller models (we halve the number of layers) than those reported in the main text and train for 500 epochs, so they are not directly comparable to the model in the main text. Results are shown in Table 6.

Table 6. Model performance on ZINC250K using different conformer generation methods from RDKit.

	ETKDG	Pharm3D
VAE Reconstruction Accuracy	99.92%	99.90%
KL	0.94	0.93
FCD	0.75	0.70

D. Invariance and Equivariance Properties of EDM-SyCo

In this work, we are dealing with two types of symmetries arising from how we represent the data and/or physical symmetries. Specifically, graphs are invariant to permuting their nodes, meaning that a single graph of N nodes can be represented by $N!$ different representations corresponding to $N!$ different orderings of its nodes. We refer to this transformation as the elements of the symmetric group S_N. Similarly, a point cloud representing a molecule in 3D can be arbitrarily rotated, reflected, and/or translated and still refer to the same molecule. This corresponds to an *infinite* number of different representations for the same object. We refer to this transformation as the action of the Euclidean group $E(3)$.

D.1. S_N- and $E(3)$-Invariant Training Process

To efficiently learn from such data, we need to develop training methods invariant to these symmetries without needing data augmentation techniques. Concretely, we must ensure that the gradient updates used to train our model do not change if the molecular graph is permuted and/or the latent point cloud representation is rotated/reflected/translated. We require two ingredients to achieve this: an equivariant architecture and an invariant loss (Vignac et al., 2022). Based on the model architecture and the loss function described in the main text, we can show that EDM-SyCo satisfies both requirements and that its training process is, thus, invariant to the action of the permutation group S_N and the Euclidean group $E(3)$.

As our ultimate goal is to generate molecular graphs, we provide more details on the permutation invariance properties of the distribution of molecular graphs learned by EDM-SyCo in the next section.

D.2. Proof of Proposition 4.1 (S_N-Invariant Marginal Distribution)

In this section, we provide a formal proof for the statement from the main text that the molecular graph distribution defined by EDM-SyCo is invariant to permutations. For convenience, we provide the proposition again:

Proposition D.1. *The marginal distribution of molecular graphs $p_{\theta,\xi}(\mathcal{G}) = \mathbb{E}_{p_\theta(\boldsymbol{z}_0)}\left[p_\xi(\mathcal{G}|\boldsymbol{z}_0)\right]$ defined by the EDM and the decoder, is an S_N-invariant distribution, i.e. for any molecular graph $\mathcal{G} = (\boldsymbol{h}, \boldsymbol{A})$, $p_{\theta,\xi}(\boldsymbol{h}, \boldsymbol{A}) = p_{\theta,\xi}(P\boldsymbol{h}, P\boldsymbol{A}P)$ for any permutation matrix $P \in S_N$.*

Note: This invariance property is required for efficient likelihood computation, as the likelihood of a graph is the sum of the likelihoods of its $N!$ permutations, which is intractable to compute. However, when ensuring that all these permutations are assigned equal likelihood, it suffices to compute the likelihood of an arbitrary permutation.

Proof. The idea of the proof is when the initial distribution of the diffusion model $p_\theta(\boldsymbol{z}_T)$ is invariant and the transition distributions $p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ are equivariant, then all marginal distributions at all diffusion time steps $p_\theta(\boldsymbol{z}_t)$ are invariant, including $p_\theta(\boldsymbol{z}_0)$ (Xu et al., 2022; 2023; Hoogeboom et al., 2022). With the same logic and with the equivariant decoder taking the place of the transition distribution, the graph distribution $p_{\theta,\xi}(\mathcal{G})$ will be invariant.

We prove this result by induction:

Base case: $p_\theta(\boldsymbol{z}_T) = \mathcal{N}(\boldsymbol{z}_T; \boldsymbol{0}, \boldsymbol{I})$ is permutation-invariant, i.e. $p_\theta(\boldsymbol{z}_T) = p_\theta(P\boldsymbol{z}_T)$ for all permutation matrices $P \in \{0,1\}^{N\times N}$ acting on $\boldsymbol{z}_T \in \mathbb{R}^{N\times(d+3)}$ by permuting its rows. This holds because all rows are i.i.d., as $p_\theta(\boldsymbol{z}_T)$ has a diagonal covariance matrix.

Induction step: Assume $p_\theta(\boldsymbol{z}_t)$ is permutation-invariant. We show that if $p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ is permutation-equivariant, i.e. $p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = p_\theta(P\boldsymbol{z}_{t-1}|P\boldsymbol{z}_t)$ (which holds when using a permutation-equivariant architecture $\boldsymbol{\epsilon}_\theta$), then $p_\theta(\boldsymbol{z}_{t-1})$ will be permutation-invariant.

$$
\begin{aligned}
p_\theta(P\boldsymbol{z}_{t-1}) &= \int_{\boldsymbol{z}_t} p_\theta(P\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)p_\theta(\boldsymbol{z}_t) \qquad \text{(Chain rule of probability)}\\
&= \int_{\boldsymbol{z}_t} p_\theta(P\boldsymbol{z}_{t-1}|PP^{-1}\boldsymbol{z}_t)p_\theta(PP^{-1}\boldsymbol{z}_t) \qquad (PP^{-1} = \boldsymbol{I})\\
&= \int_{\boldsymbol{z}_t} p_\theta(\boldsymbol{z}_{t-1}|P^{-1}\boldsymbol{z}_t)p_\theta(P^{-1}\boldsymbol{z}_t) \qquad \text{(Equivariance and invariance)}\\
&= \int_{\boldsymbol{u}} p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{u})p_\theta(\boldsymbol{u})\underbrace{|\det P|}_{=1} \qquad \text{(Change of variables } \boldsymbol{u} = P^{-1}\boldsymbol{z}_t \text{ and } P \text{ is orhtogonal, so} \det P = \pm 1)\\
&= p_\theta(\boldsymbol{z}_{t-1})
\end{aligned}
$$

By induction, $p_\theta(\boldsymbol{z}_T), \ldots, p_\theta(\boldsymbol{z}_0)$ are all permutation-invariant.

Finally, since the decoder $p_\xi(\mathcal{G}|\boldsymbol{z}_0)$ is permutation-equivariant (as it applies the same operation on each node and each pair of nodes, thus not depending on the nodes ordering), with the same derivation, we also conclude that the final induced distribution $p_{\theta,\xi}(\mathcal{G})$ is permutation-invariant. □

Note: A similar derivation can also be applied to show the invariance of $p_\theta(\boldsymbol{z}_0)$ to rotations and reflections of the coordinates, as the only requirement on P for the derivation to hold is to be orthogonal, which is the case for rotation and reflection matrices R.

E. Implementation Details

E.1. Architecture Details

We train two sets of models with the same architectures on the two used datasets, ZINC250K and GuacaMol. The EDM, together with the autoencoder, has 9.2M total parameters, while the regressor has 4.2M parameters.

Encoder. The first part of the encoder $\mathcal{E}$, introduced in Section 3, is the conformer generation method, while the second part is an EGNN with 1 layer and 128 hidden features (See Appendix A.2). We found that having at least 1 EGNN layer is crucial to achieve high reconstruction accuracy for the autoencoder, and as this allows us to reach more than 99% accuracy, we did not use more layers. This EGNN takes as input a point cloud with the one-hot encodings of the atom types and formal charges as features $\boldsymbol{h}$ and the synthetic coordinates computed by the conformer generation part as coordinates $\boldsymbol{x}$. The output is a processed point cloud with continuous node embeddings of dimension $d = 2$ and updated coordinates.

Decoder. The decoder $\mathcal{D}$, also introduced in Section 3, consists of 2 fully connected networks MLP_{node} and MLP_{edge}, both having 256 hidden dimensions.

EDM. The denoising network of EDM is parametrized with an EGNN with 9 layers and 256 hidden features. The EDM has $T = 1000$ diffusion steps.

Regressor. The regressor used for the conditional generation and optimization experiments (see Section 5) is also an EGNN with 4 layers and 256 hidden features, followed by a sum pooling operation over the node features and then a fully connected network with 2 layers and a hidden dimension of 256 that maps the pooled graph embedding to the target values of the properties.

E.2. Training Details

We use the original train/validation/test splits of the used datasets (Irwin et al., 2012; Brown et al., 2019). The autoencoder is trained in the first stage to minimize the cross-entropy loss between the ground truth and predicted graphs for a maximum of 100 epochs, with early stopping if the validation accuracy does not improve for 10 epochs. To deal with the class imbalance problem caused by the sparsity of the graph adjacency matrices and the dominance of some atom types and formal charge values, we scale each term of the cross-entropy loss with a class-specific value based on the statistics of the training set (King & Zeng, 2001). In the second training stage, the EDM model is trained for 1000 epochs on ZINC250 and approximately 300 epochs on GuacaMol.

The regressor is trained with the L_1 loss to predict the molecular properties on the noisy latent point cloud representations of the molecules by applying the same noise schedule of the EDM. The regressor is trained for 500 epochs, with early stopping if the MAE on the validation set does not improve after 50 epochs.

All models are trained with a batch size of 64 and using the recent Prodigy optimizer (Mishchenko & Defazio, 2023) with $d_{coef} = 0.1$, which we found to be a very important hyperparameter for the stability of training.

We train all models on ZINC250K on a single Nvidia A100 GPU, and on GuacaMol, we use multi-GPU training on 4 Nvidia A100 GPUs.

E.3. Dataset Details

We use two datasets in our experiments: ZINC250K (Irwin et al., 2012) and GuacaMol (Brown et al., 2019). For both datasets, we represent a molecule as a graph. The nodes are atoms with the one-hot encoding of their atom type and their formal charge as features. In addition to modeling all heavy atoms, we also model explicit H atoms, which we found to increase the validity of the generated molecules as it reduces some kekulization errors produced by rdkit [2]. The edges are chemical bonds with the one-hot encoding of the bond type as features. We model the following bond types: no-bond, single, double, triple, and aromatic. We model the absence of a bond as a separate bond type to allow the autoencoder to

[2]https://github.com/rdkit/rdkit/wiki/FrequentlyAskedQuestionscant-kekulize-mol

reconstruct the full bond information. These graphs present very unbalanced scales for the atom types, formal charges, and bond types. We use class-specific weights to train the autoencoder, as outlined in Section 3.

ZINC250K has a total of 250K molecules and 220,011 training molecules. It has 10 atom types (including H) and 3 formal charge values (-1, 0, 1), and the largest molecule has 40 atoms.

GuacaMol has a total of $\approx$1.5M molecules, of which we use 1,273,104 for training. It has 13 atom types (including H) and 5 formal charge values (-1, 0, 1, 2, 3), and the largest molecule has 88 atoms.

We compute the synthetic coordinates needed for our model in a preprocessing step for both datasets and use the computed coordinates throughout the training.

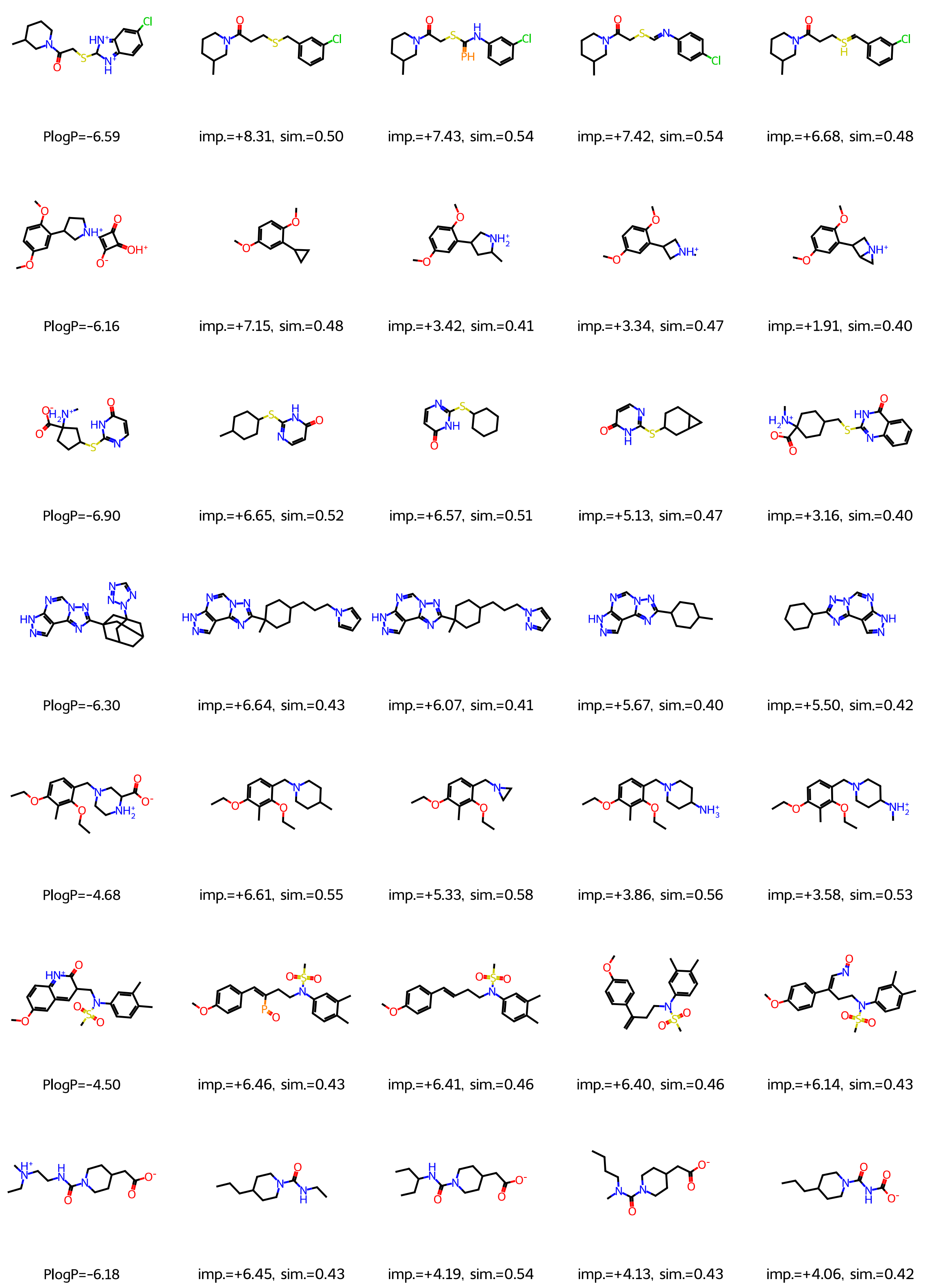

Figure 6. Molecules with the highest improvement in the LogP-constrained optimization task. Each row corresponds to one test molecule and 4 successful optimization results. We show the LogP value of the initial molecules, and for the optimized molecules, the achieved improvement (imp.) and the Tanimoto similarity to the initial molecule (sim.).

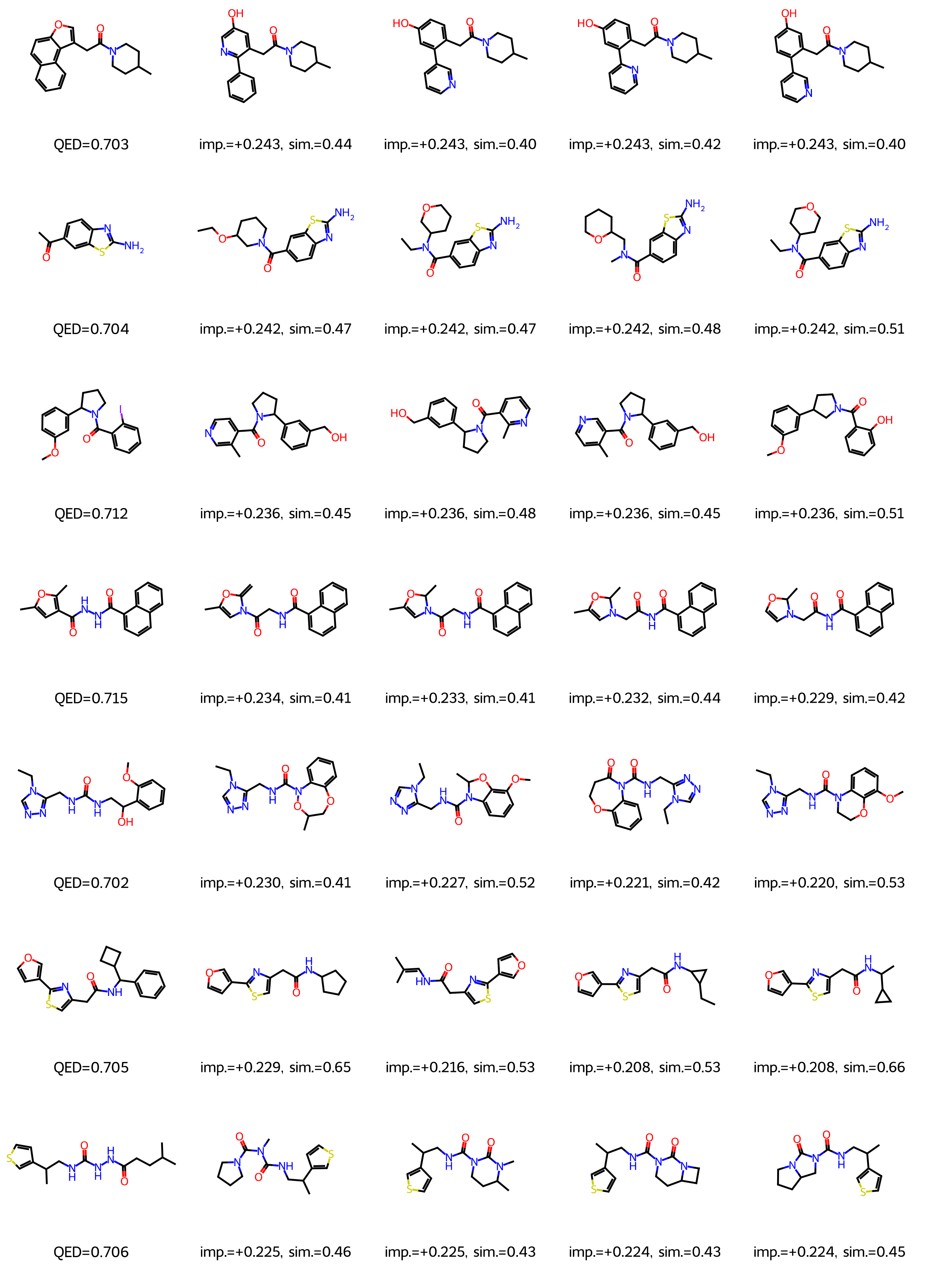

Figure 7. Molecules with highest improvement in the QED constrained optimization task. Each row corresponds to one test molecule and 4 successful optimization results. We show the QED value of the initial molecules, and for the optimized molecules, the achieved improvement (imp.) and the Tanimoto similarity to the initial molecule (sim.).