Any-Property-Conditional Molecule Generation with Self-Criticism using Spanning Trees

Anonymous authors
Paper under double-blind review

Abstract

Generating novel molecules is challenging, with most representations of molecules leading to generative models producing many invalid molecules. Spanning Tree-based Graph Generation (STGG) (Ahn et al., 2021) is a promising approach to ensure the generation of valid molecules, outperforming state-of-the-art generative models for unconditional generation. In practice, it is desirable to generate molecules conditional on one or multiple target properties rather than unconditionally. Thus, we extend STGG to multi-property conditional generation. Our approach, **STGG+**, incorporates a modern Transformer architecture, random masking of properties during training (enabling conditioning on *any* subset of properties and classifier-free guidance), an auxiliary property-prediction loss (allowing the model to *self-criticize* molecules and select the best ones), and other improvements. We show that **STGG+** achieves state-of-the-art performance on in-distribution and out-of-distribution conditional generation, as well as reward maximization.

1 Introduction

Generating novel molecules is challenging, and the choice of molecular representation significantly impacts the performance of generative models. Recent methods generate molecules in 2D (as graphs) (Jo et al., 2022; Vignac et al., 2022; Thompson et al., 2022; Jo et al., 2023) or 3D (Hoogeboom et al., 2022; Bao et al., 2022; Xu et al., 2023; Huang et al., 2024). However, 1D representations, such as SMILES (Weininger, 1988), remain the standard representation in practical molecule design (Segler et al., 2018; Kwon et al., 2023; M. Bran et al., 2024; Wu et al., 2024).

A key issue with molecule representations is that a single error by the generative model can result in invalid molecules, which is especially challenging as the molecule size increases. Spanning Tree-based Graph Generation (STGG) (Ahn et al., 2021) tackle this issues by masking invalid tokens during sampling, preventing the generation of invalid structures (e.g., atoms without bonds between them, branch end before branch start) and ensuring proper valency. The masking employed by STGG not only prevents invalid molecules, but also leads to higher-quality and more diverse generated molecules, outperforming or matching more recent state-of-the-art generative models for unconditional generation (Jang et al., 2023). However, its application in multi-property conditional settings has not been explored. We address this setting along with a few additional challenges, as discussed below.

Any-property-conditioning. In real-world applications, it is desirable to generate molecules *conditional on one or multiple target properties* rather than unconditionally Jain et al. (2023). Furthermore, we want to condition on *different* subsets of desirable properties without needing to retrain the model for each subset.

Self-criticism. A critical issue in molecule discovery is synthesis time, which can take weeks or months. To avoid extreme synthesis costs, we need to filter the molecules that we provide to chemists. Some properties can be verified through simulations, but this is slow, and not all properties can be simulated. Another option is to rely on external property predictor models, but training, validating, and managing multiple property predictors can be challenging. To address this issue, we propose giving the model the ability to predict properties and thus *self-criticize* its own generated molecules, allowing it to filter out those with undesirable properties. Such self-criticism mechanisms are common in language models Madaan et al. (2024); however, they are not used in STGG.

Out-of-distribution properties. We often seek to generate novel molecules with out-of-distribution (OOD) properties to discover new structures. Classifier-Free Guidance (CFG) (Ho & Salimans, 2022) is a technique to improve

conditioning fidelity; we found CFG useful for in-distribution properties, but problematic for some OOD conditioning values (especially extreme values), resulting in poor generated molecules. Since guidance can be beneficial to some conditioning values but not others, we propose *random guidance* with best-of-k self-filtering (described in Section 3.5).

To summarize, we tackle any-property-conditional molecule generation in a practical, real-world setup. In doing so, we make the following contributions:

1. We make multiple improvements compared to STGG (Figure 1; ablation in Section 4.5):
   (a) **Any-property-conditioning** enabling conditioning on multiple target properties and any subset of them without retraining (Section 3.2);
   (b) **Improved architecture** based on recent advances in Transformers (Section 3.1);
   (c) **Improved Spanning-Tree** with enhanced tokenization and masking to improve validity and generalization (Section 3.3);
   (d) **Auxiliary property prediction objective** enabling the self-criticism ability and improving conditioning fidelity (Section 3.6);
   (e) **Random guidance for extreme value conditioning** to improve classifier-free guidance on extreme OOD conditioning fidelity (Section 3.5).

2. By comprehensive evaluation, we demonstrate excellent performance in terms of:
   (a) high quality and diversity for unconditional generation on QM9 and Zinc (Appendix A.7);
   (b) high quality and conditioning fidelity for in-distribution (Section 4.1) and OOD (Sections 4.2 and 4.4) conditional generation on the HIV, BACE, and BBBP datasets, as well as the challenging Chromophore DB with larger and more complex molecules;
   (c) high reward and diversity for multi-objective reward maximization on QM9 (Section 4.3).

Our evaluation spans diverse molecule generation scenarios, including realistic (yet underexplored in machine learning) cases like the Chromophore DB dataset and HOMO-LUMO property optimization. This underscores the significance of our proposed approach and its exploration.

2 Background and Related Work

2.1 1D vs 2D vs 3D representations

There are many ways of representing molecules in the context of molecular generation. Some of the most popular methods are (1) autoregressive models on 1D strings (Segler et al., 2018; Ahn et al., 2021; Kwon et al., 2023) and (2) diffusion (Song & Ermon, 2019; Ho et al., 2020; Song et al., 2020) models on 2D graphs. While both approaches have similar sample complexity, 1D strings offer a more compressed representation, requiring less space and fewer parameters and, thus, increased potential for scalability. We provide a detailed comparison between different representations in Appendix A.1.

Furthermore, recent results indicate that 1D strings are as competitive as 2D molecular graph methods for both unconditional molecule generation (Jang et al., 2023; Fang et al., 2023) and property prediction (Yüksel et al., 2023). Both 1D and 2D representations encapsulate the same amount of information, making the choice largely a matter of preference. We advocate for 1D string representations due to their scalability and effective utilization by Transformer models, and thus, we focus on this type of representation in our work.

Molecules are inherently 3D objects and can be generated in 3D (Hoogeboom et al., 2022; Bao et al., 2022; Xu et al., 2023; Huang et al., 2024; Song et al., 2024; Luo et al., 2024; Verma et al., 2022; Mercatali et al., 2024). Generating 3D molecules is challenging and often results in fewer valid molecules. Since 3D conformations can be inferred from 1D or 2D molecular representations using popular chem-informatics tool Sun et al. (2020); Landrum et al. (2024), 1D strings like SMILES remain the standard representation in molecule design (M. Bran et al., 2024; Wu et al., 2024).

The most popular choice of 1D string-based representation is SMILES (Weininger, 1988), an extremely versatile method capable of representing any molecule. However, when used in generative models, generated SMILES strings often correspond to invalid molecules. A single incorrectly placed token often leads to an invalid molecule. Graph-based

diffusion methods also face a similar issue (Vignac et al., 2022). Recent methods such as STGG(Ahn et al., 2021) and SELFIES (Krenn et al., 2022) were developed to tackle this issue by preventing the generation of invalid molecules. For a detailed comparison of SMILES, SELFIES, and STGG, see Appendix A.2.

SELFIES has been shown to perform worse than SMILES on property-conditional molecule generation (Gao et al., 2022; Ghugare et al., 2023). Meanwhile, SMILES has been shown to perform worse than STGG on unconditional generation using the same architecture (Ahn et al., 2021). Furthermore, STGG performs on-par or better than state-of-the-art unconditional generative models(Ahn et al., 2021; Jang et al., 2023). Therefore, our work focuses exclusively on extending and improving STGG.

2.2 STGG

STGG (Ahn et al., 2021) uses a SMILES-like vocabulary with begin "`(`" and end "`)`" branch tokens, ring start "`[bor]`" and i-th ring end "`[eor-`i`]`" tokens. Contrary to SELFIES, STGG was made from the ground up for molecule generation. STGG leverages a Transformer architecture (Vaswani et al., 2017) to sample the next tokens conditional on the tokens of the current unfinished molecule. To predict the ring end tokens, STGG uses a similarity-based output layer distinct from the linear output layer used to predict other tokens. STGG also uses an input embedding to track the number of open rings. Invalid next tokens are prevented through masking of next tokens that would lead to impossible valencies (e.g., atoms, ring-start, and branch-start when insufficient valency remains) and structurally invalid tokens (e.g., ring-i end when fewer than i ring start tokens are present).

In the next section, we show how to improve STGG's architecture, vocabulary, and masking, adapt STGG for any-property conditional generation, and improve conditioning fidelity through classifier-free guidance, self-criticism, and random classifier-free guidance for extreme conditioning.

3 Method

3.1 Architecture

We enhance the architecture used in STGG, which is a regular Transformer (Vaswani et al., 2017) directly from PyTorch main libraries. To improve it, we leverage recent improvements in Large Language Models following GPT-3 (Radford et al., 2019), Mistral (Jiang et al., 2023), and Llama (Touvron et al., 2023). The improvements include: 1) RMSNorm (Zhang & Sennrich, 2019) replacing LayerNorm (Ba et al., 2016); 2) residual-path weight initialization (Radford et al., 2019); 3) bias-free architecture (Chowdhery et al., 2023); 4) rotary embeddings (Su et al., 2024) instead of relative positional embedding; 5) lower-memory and faster attention with Flash-Attention-2 (Dao et al., 2022; Dao, 2023); 5) SwiGLU activation function (Hendrycks & Gimpel, 2016; Shazeer, 2020); 6) changes in hyperparameters following GPT-3 (Radford et al., 2019) (i.e., AdamW (Loshchilov & Hutter, 2017; Kingma & Ba, 2014) $\beta_2 = 0.95$, cosine annealing schedule (Loshchilov & Hutter, 2016), more attention heads, no dropout). These modifications aim to enhance the model's efficiency, scalability, and overall performance. We also considered more efficient architectures (Appendix A.5).

3.2 Any-property conditioning

We first preprocess continuous properties by applying a simple z-score standardization. To condition the model on any subset of desired properties without retraining we need to be able to "turn off" properties by masking them. For continuous variables, we handle such missing values through a binary indicator variable: if a property is missing, we set the property value to 0 and the missing indicator to 1. It is important to include these missing indicators because we cannot assume the plausible values for the missing features (e.g., if A is 1.0, B is missing, maybe the only possible range for B is around 3 and 4, so if we leave B at 0 without missing indicator, it may confuse the model). For categorical variables, we add an extra category for missing values. During training, we mask a random subset of t properties, where t is chosen uniformly between 0 and the total number of properties (Appendix A.3.2). This allows us to condition the model on any subset of desired properties at test time while ignoring the rest.

Our model takes the standardized continuous features (continuous properties concatenated with their binary missing indicators) as input and processes them using a 2-layer multilayer perceptron (MLP) with the Swish activation

(Hendrycks & Gimpel, 2016; Ramachandran et al., 2017). Each categorical feature is then processed individually using a linear embedding. These processed outputs are added directly to the embedding of all tokens (Figure 1).

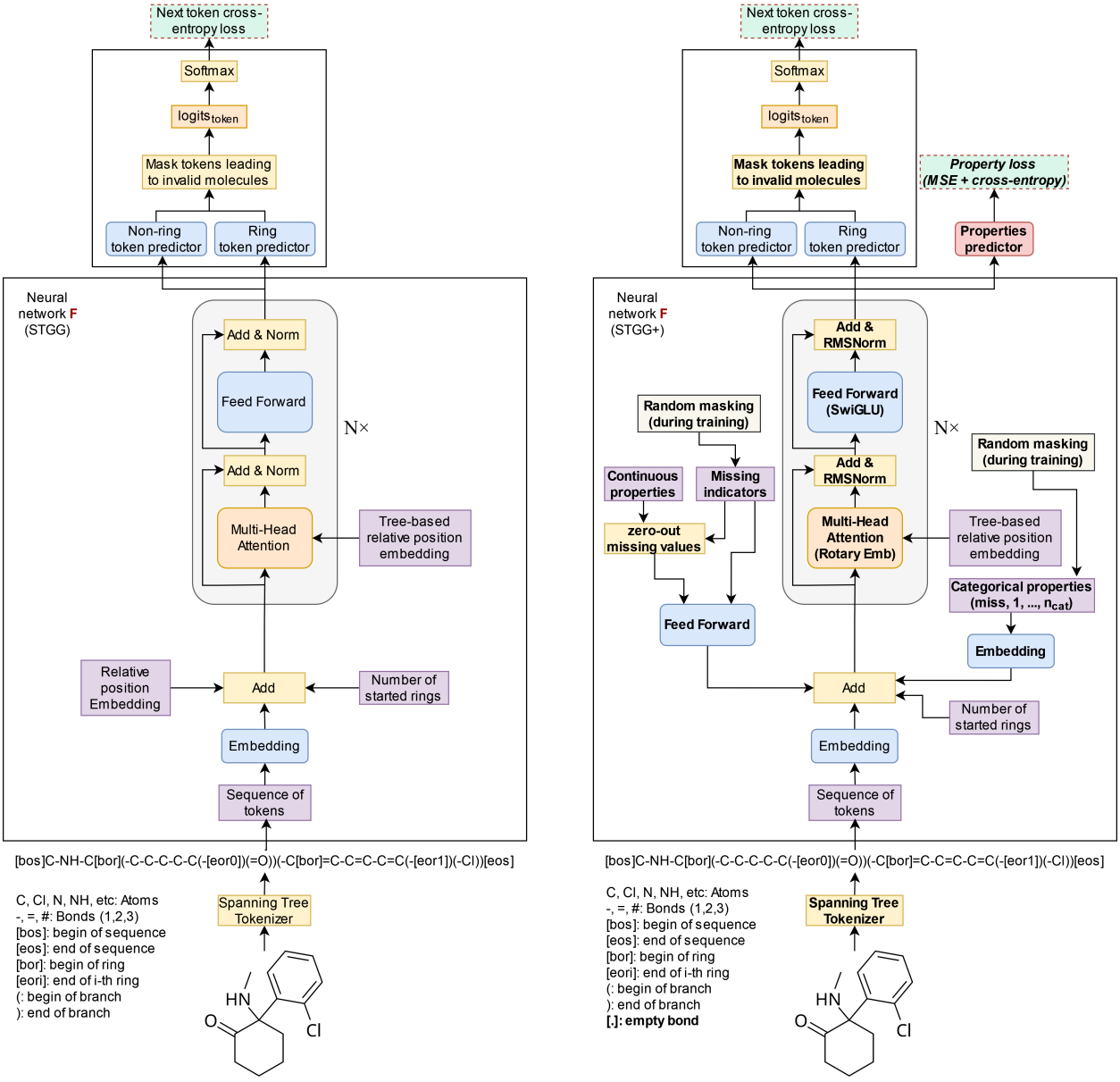

Figure 1: Left: STGG architecture, Right: Our **STGG+** architecture. The molecule is tokenized and embedded. Properties embeddings are added. The output produces the property and next-token predictions (masked to prevent invalid tokens). Novel components compared to STGG are in **bold**. See Section 3 for explanations on the components. See Section 4.5 for an ablation showing the importance of each component.

3.3 Improvements to Spanning-Tree

Starting from STGG as base, we implement several improvements. Firstly, we extend the vocabulary to allow for the generation of molecular compounds that are composed of multiple unconnected graphs (e.g., salt is represented as [Na+].[Cl-], where [Na+] and [Cl-] are single-atom molecules connected through an ionic bond), enabling the model to solve a broader range of problems. STGG uses a fixed vocabulary and a fixed set of maximum valencies that determines

how many valence bonds each atom can form. Instead of requiring a predefined vocabulary, we automate the process of building a vocabulary based on the atoms found in the dataset and their maximum valency, again derived from the dataset. This data-centric approach allows us to represent complex structures, including non-molecular compounds containing metals.

We observe that STGG can occasionally generate incomplete samples by creating too many branches without closing them within the allowed maximum length, particularly when conditioning on extreme out-of-distribution properties. To address this, we modify the token masking process to ensure the model closes its branches when the number of open branches approaches the number of tokens left to reach the maximum length. This additional masking step prevents the rare but problematic situation of incompletely-generated samples. Additionally, for larger molecules, it is possible for the model to produce more rings than the maximum number of rings (100); we now mask the creation of rings when the maximum number is reached. With these additional masks, we generally maintain 100% validity, even when generating molecules with out-of-distribution properties. We describe the masking algorithms in details (before and after the changes) in Appendix A.12.

Contrary to STGG, we do not canonicalize molecules and instead use a random ordering of molecule components (a different random ordering is sampled for each molecule during training). This improves generalization as it can be seen as data augmentation (see ablations in Section 4.5).

3.4 Classifier-free guidance

To enforce better conditioning of the properties, we use classifier-free guidance (CFG), originally designed for diffusion models (Ho & Salimans, 2022), which was found to be beneficial for autoregressive language models too (Sanchez et al., 2023). This technique involves directing the model more toward the conditional model's direction while pushing it away from the unconditional model's direction by an equal amount (Figure 2).

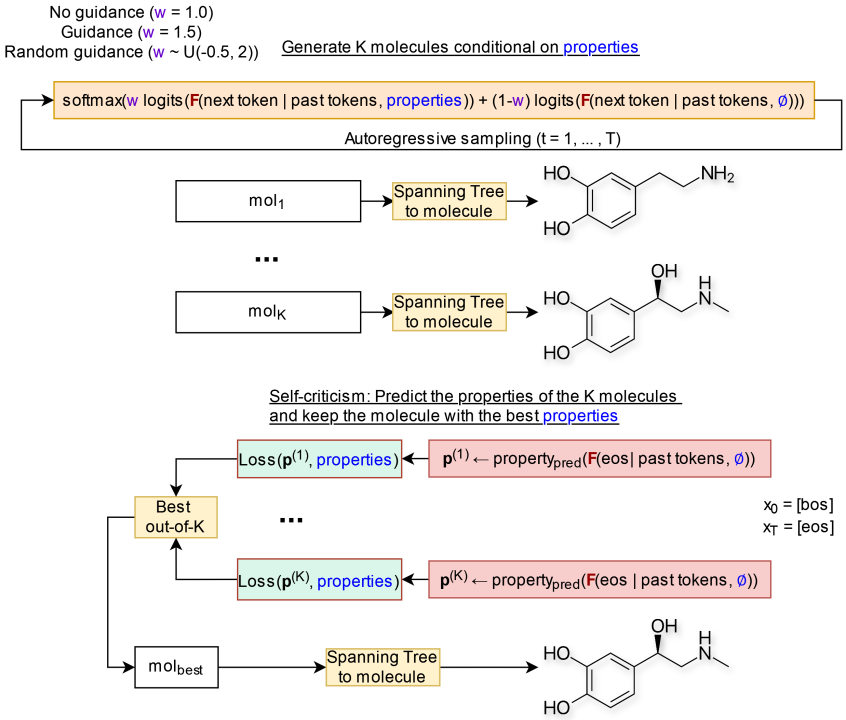

Figure 2: Generation and self-prediction using **STGG+**. Generate K molecules conditional on properties using classifier-free guidance. The unconditional model predicts the properties of the K molecules; the molecule assumed closest to the desired properties is returned.

3.5 Random guidance for extreme conditioning

Guidance (Ho & Salimans, 2022) can be problematic for extreme (out-of-distribution) conditioning values, resulting in poor "generative efficiency" (% of valid, unique, and novel molecules) and conditioning fidelity. However, guidance can still be beneficial to some extreme conditioning values. To improve generative performance on extreme conditioning values, we propose to randomly sample a guidance $w \sim \mathcal{U}(-0.5, 2)$ for each sample, ensuring high diversity through a mix of low and high guidance. Then, using self-criticism, our method selects the best-out-of-k molecule from the molecules generated at different guidance levels, indirectly allowing the model to determine by itself which guidance is best for each sample. This is effectively a way to balance exploration and exploitation (higher guidance means less diversity and better property-alignment, while lower guidance means more diversity and less property-alignment).

3.6 Self-criticism

To make the model more powerful, we provide the model with the ability to self-criticise its own generated molecules. The purpose is improve the quality of generated samples by using a jointly-trained property predictor to rank and filter the generated samples. It works as follows: 1) the model generates k molecules for a given set of properties, 2) it evaluates the k molecules molecules properties based on its own property-predictor (see the paragraph below), and 3) it returns the molecule whose properties best match the conditioned properties. This best-out-of-k strategy significantly improves the quality of its generated molecules.

For the model to be able to predict properties of the molecules, we add a property-prediction loss to the training objective (Figure 1). During training, the model is tasked with predicting both the next token and the properties of the current unfinished molecule. During sampling, we generate molecules conditioned on desired properties with classifier-free guidance (Figure 2). Then, we mask out the properties (making them as fully missing) and reprocess the molecule until we reach the end-of-sequence (EOS) token. At this point, we extract the predicted property of this molecule.

4 Experiments

We run four sets of experiments. First, we show that our model can generate molecules conditioned on properties from the test set with high fidelity. Second, we show that our model can efficiently generate (high % of novel, unique, and valid) molecules with high fidelity on out-of-distribution (OoD) properties. Third, we show that our model can produce molecules that maximize a reward function, achieving similar or better performance compared to online learning methods using offline learning. Finally, we show that our model can generate high fidelity molecules conditioned on out-of-distribution (OoD) properties on a small dataset of larger and more complex molecules.

We experiment with six datasets: (1) QM9 (Ramakrishnan et al., 2014) with around 134k molecules and maximum SMILES length of 37; (2) Zinc250K (Sterling & Irwin, 2015) with 250k molecules and maximum length of 136; (3) BBBP (Wu et al., 2018) with 862 molecules and maximum length of 186; (4) BACE (Wu et al., 2018) with 1332 molecules and maximum length of 161; (5) HIV (Wu et al., 2018) with 2372 molecules and maximum length of 193; (6) Chromophore DB (Joung et al., 2020) with 6810 molecules and maximum length of 511. See Appendix A.3 for details on these datasets, A.4 for more information on the hyperparameters, A.6 for property prediction performance metrics of the self-critic. We also perform unconditional generation experiments on QM9 and Zinc250k in Appendix A.7. We rely on the following software: PyTorch (Paszke et al., 2019), Molecular Sets (MOSES) (Polykovskiy et al., 2020) and RDKit (Landrum et al., 2024). Unless otherwise specified, we use RDKit to evaluate the properties of the generated molecules.

We follow the same protocol as Liu et al. (2024). We train our model on HIV, BACE, and BBBP. We use the same train, valid, and test splits as Liu et al. (2024). Each dataset has an experimental categorical property related to HIV virus replication inhibition (HIV), blood-brain barrier permeability (BBBP), or human β-secretase 1 inhibition (BACE), respectively, and two continuous properties: synthetic accessibility (SAS) (Ertl & Schuffenhauer, 2009) and complexity scores (SCS) (Coley et al., 2018). We evaluate the models using metrics on distribution and fidelity of conditioning after generating molecules conditional on properties from the test set. The condition control metrics are the Mean Absolute Error (MAE) of SAS (evaluated by RDKit) and accuracy of the categorical property (evaluated by a Random Forest (Breiman, 2001) predictor using the Morgan Fingerprint (Morgan, 1965; Gao et al., 2022)). The distribution metrics are validity, atom coverage in the largest connected graph (how many unique atom types are produced), internal

diversity (average pairwise similarity of generated molecules), fragment-based similarity (Degen et al., 2008), Fréchet ChemNet Distance (FCD) (Preuer et al., 2018). We consider any atom coverage above the test set coverage to indicate good coverage. The Property accuracy metric depends on a RandomForest classifier, thus we consider any accuracy equal or above the test set to indicate good condition control.

4.1 In-distribution conditional generation

Baselines. Following Liu et al. (2024), we compare our method to strong recent baselines: GraphGA (Jensen, 2019), MARS (Xie et al., 2021), LSTM on SMILES with Hill Climbing (LSTM-HC) (Hochreiter & Schmidhuber, 1997; Brown et al., 2019), and powerful graph diffusion models: DiGress (Vignac et al., 2022), GDSS Jo et al. (2022), MOOD (Lee et al., 2023), and the state-of-the-art Graph DiT (Liu et al., 2024) (oral presentation at NeurIPS 2024).

Table 1: Conditional generation of 10K molecular compounds on HIV, BBBP, and BACE.

Tasks	Metric	Validity ↑	Distribution Learning: Coverage* ↑	Diversity ↑	Similarity ↑	FCD ↓	Condition Control: MAE ↓	Accuracy * ↑
	Model \ Property						SAS	BACE, BBBP, or HIV
SAS & BACE	MOOD	1.00	8/8	0.89	0.26	44.24	1.89	0.51
	Graph GA	1.00	8/8	0.86	0.98	7.41	0.96	0.47
	Graph DiT	0.87	8/8	0.82	0.88	7.05	0.40	0.91
	STGG**	1.00	8/8	0.82	0.98	3.82	0.45	0.95
	STGG+ ($k = 1$)	1.00	8/8	0.83	0.98	3.80	0.24	0.91
	STGG+ ($k = 5$)	1.00	8/8	0.83	0.98	3.80	0.18	0.93
	Test data	1.00	**7/8***	0.82	1.00	0.00	0.00†	**0.82***
SAS & BBBP	MOOD	0.80	9/10	0.93	0.17	34.25	2.03	0.49
	Graph GA	1.00	9/10	0.90	0.95	10.17	1.21	0.30
	Graph DiT	0.85	9/10	0.89	0.93	11.85	0.36	0.94
	STGG**	1.00	9/10	0.89	0.92	11.74	0.98	0.75
	STGG+ ($k = 1$)	1.00	10/10	0.89	0.94	9.86	0.47	0.87
	STGG+ ($k = 5$)	1.00	9/10	0.89	0.94	10.10	0.38	0.90
	Test data	1.00	**10/10***	0.88	1.00	0.00	0.02	**0.81***
SAS & HIV	MOOD	0.29	29/29	0.93	0.14	32.35	2.31	0.51
	Graph GA	1.00	28/29	0.90	0.97	4.44	0.98	0.60
	Graph DiT	0.77	28/29	0.90	0.96	6.02	0.31	0.98
	STGG**	1.00	27/29	0.90	0.96	4.56	0.44	0.95
	STGG+ ($k = 1$)	1.00	27/29	0.90	0.97	4.08	0.31	0.88
	STGG+ ($k = 5$)	1.00	24/29	0.90	0.97	4.32	0.23	0.91
	Test data	1.00	**21/29***	0.90	1.00	0.07	0.02	**0.73***

*The classifier from Liu et al. (2024) (used in the last column) has limited accuracy on the test set; thus, any *Property Acc.* above the **test data accuracy** is not indicative of better quality. Similarly, atom coverage is not 100% on test data; thus, any coverage above the **test set coverage** does not indicate better performance.

**STGG with categorical embedding, missing indicators, random masking, and extra symbol for compounds.

†The dataset properties are rounded to two decimals hence MAE is not exactly zero.

Results. The experiment results are shown in Table 1 (for the full table with more baselines, see Appendix A.8). We find that **STGG+** obtains near-perfect validity, coverage consistently higher than the test set, high diversity, and high test-set similarity. Notably, we attain the best FCD; in fact, we are the only method that matches the training data's performance, indicating that we have reached the performance cap. Regarding condition control, we achieve the best MAE on BACE and HIV, and the second-best on BBBP (very close to Graph DiT). We also obtain better performance than base STGG (with random masking and the extra symbol for compounds) on FCD and MAE, which shows that our improvements lead to lower distance in distribution and better property conditioning. We further observe that self-criticism ($k > 1$) improves property fidelity (lower MAE) while sacrificing a bit of diversity (lower coverage).

4.2 Out-of-distribution conditional generation

We follow the same protocol as Kwon et al. (2023). Our model is trained on Zinc250K using exact molecule weight, logP, and Quantitative Estimate of Druglikeness (QED) (Bickerton et al., 2012) as properties. For evaluation, we generate 2K candidate molecules and calculate two metrics: 1) generative efficiency, defined as the probability that the following three conditions are satisfied at the same time: validity, uniqueness (not a duplicate), and novelty (not in train data)), and 2) the Minimum Mean Absolute Error (MinMAE) between the generated and conditioned properties (at ± 4 standard-deviation). Note that for QED, the high condition value is at an impossible value of 1.2861 (the possible range is 0 to 0.948). Conditioning on impossible values is not ideal, but we choose to follow the protocol of Kwon et al. (2023) for better comparability, and it lets us verify whether our model behaves reasonably when conditioning values are difficult or impossible to achieve. We use the same train, valid, and test splits as Jo et al. (2022). Following Shao et al. (2020), we compare our model to vanilla VAE with k-annealing (BaseVAE) (Kingma & Welling, 2013; Bowman et al., 2015), ControlVAE (Shao et al., 2020), and various single-decoder (SD) and multi-decoders (MD) methods proposed by Shao et al. (2020).

Table 2: Out-of-distribution ($\mu \pm 4\sigma$) property-conditional generation of 2K molecules on Zinc250K. Generative efficiency (% of valid, novel, and unique molecules) and Minimum MAE. See Section A.10 for Mean MAE results over the top-100 molecules.

	Generative Efficiency			Properties - MinMAE					
	molWt	logP	QED	molWt		logP		QED	
Condition				84	580	-3.2810	8.1940	0.1778	1.2861*
MD	0.49	0.42	0.47	9.8e−2	1.7e−1	2.0e−2	3.0e−4	1.5e−3	1.0e−1
MD_{dif}	0.46	0.43	0.47	7.4e−3	4.7e−2	3.0e−4	5.1e−3	2.0e−4	2.6e−2
$MD_{\text{dif,col}}$	0.46	0.54	0.44	1.1e−1	6.2e−2	1.3e−3	5.0e−4	6.0e−4	8.6e−2
STGG**	0.99	0.99	0.99	5.8e−2	7.5e−2	7.9e−3	1.9e−1	1.5e−2	8.0e−4
STGG+ $(k = 1)$	0.82	0.82	0.54	8.6e−3	9.1e−3	1.0e−4	1.6e−3	1.0e−5	5.1e−1
STGG+ $(k = 5)$	0.88	0.74	0.50	1.1e−3	1.7e−2	1.0e−4	1.6e+0	1.0e−4	5.2e−1
STGG+ $(w \sim \mathcal{U}(-0.5, 2), k = 1)$	0.94	0.92	0.82	2.1e−2	2.4e−2	1.0e−4	7.0e−4	7.0e−6	5.8e−3
STGG+ $(w \sim \mathcal{U}(-0.5, 2), k = 5)$	0.90	0.77	0.79	1.0e−3	6.1e−3	2.0e−7	2.8e−2	1.0e−4	1.2e−3
Train data (closest sample)	-	-	-	5.7e+1	7.3e+1	1.5e−1	2.0e−3	1.8e−2	8.2e−4

*The value is improper; we condition on 1.2861 but calculate the MAE with respect to the max QED (0.948).
**STGG with missing indicators, and random masking.

Results. We see in Table 2 that base STGG (with random masking) reaches the best generative efficiency (% of valid, novel, and unique molecules), but performs much worse than **STGG+** in terms of property conditioning. Our method sacrifices a small amount of generative efficiency (when compared to base STGG) in order to obtain much better property-conditioning; we see that our method generally obtains the smallest MAE. However, while the model performs optimally when using random guidance, it struggles with high guidance values when generating molecules for the impossible QED value of 1.2861. Additionally, we observe that the model performs worse with the best-of-5 when generating molecules with high logP, suggesting that the property predictor of STGG+ makes incorrect predictions for high out-of-distribution logP values. We also report the average of the top-100 molecules MAE (Top-100 MAE) instead of the top-1 MAE (MinMAE) for our STGG+ and base STGG (Table 10 in Appendix).

For Top-100 MAE, STGG+ performs much better than STGG in all cases except for high QED, where STGG is slightly better. Random guidance is helpful for high QED and logP.

4.3 Reward maximization

Jain et al. (2023) use reinforcement learning or GFlowNet (Bengio et al., 2023) to solve a task on the QM9 (Ramakrishnan et al., 2014) dataset. They seek to produce QM9-like molecules that maximize a reward composed of four properties: HOMO-LUMO gap (Griffith & Orgel, 1957), SAS (Ertl & Schuffenhauer, 2009), QED (Bickerton et al., 2012), and molecular weight. This reward is maximized when the HOMO-LUMO gap is as large as possible, and SAS, QED, and weight are 2.5, 1.0, and 105, respectively. We compare to Envelope QL (Yang et al., 2019), MOReinforce (Lin et al., 2022), MOA2C (Mnih et al., 2016), GFlowNet (MOGFN-PC) (Bengio et al., 2023). The HOMO-LUMO gap is evaluated with MXMNet (Zhang et al., 2020).

Instead of using the reward, we train a STGG+ model conditioned on the four properties directly. This shows a benefit of STGG+ our approach, since we do not need to scalarize the multi-objective reward. Since the HOMO-LUMO gap needs to be maximized there is no appropriate conditioning value. We arbitrarily set it to 0.5, which corresponds to approximately five standard deviations (a limitation of our conditioning method, as we cannot maximize a property, only set a fixed value). The other properties are set to their optimal values: 2.5, 1.0, and 105.

Table 3: Reward maximization on QM9.

	Type	Data	Reward (↑)	Diversity (↑)
Envelope QL	Online	1M molecules	0.65	0.15
MOReinforce			0.57	0.47
MOA2C			0.61	0.61
MOGFN-PC			0.76	0.07
STGG**	Offline	QM9 (∼115K molecules)	0.73	0.90
STGG+$(k = 1)$			0.78	0.24

**STGG with missing indicators, and random masking.

Results. Our results are shown in Table 3. Our approach yields the highest average rewards and we obtain higher diversity than GflowNet, using ∼11.5% of the molecules. This makes our approach significantly more efficient. However, note that solving this task with online methods is a different setting and can be considered more difficult.

4.4 Hard: Small dataset of large molecules (Chromophore DB)

As a more challenging example, we explore the generation of molecules with out-of-distribution properties on Chromophore DB (Joung et al., 2020), a small dataset of around 6K molecules with an average of 35 atoms per molecule (compared to 23 atoms for Zinc250K and 9 atoms for QM9). To make the problem more realistic, we only sample 100 molecules (in the real world, chemists would decide which of those 100 molecules to synthesize based on their expert knowledge). We want to know if one of those 100 molecules has the desired out-of-distribution properties. Given the small size of the dataset, it can be useful to first pre-train on a large set of small molecules (Zinc250K) and then fine-tune on the smaller dataset of large molecules (Chromophore DB). We try this strategy (pre-train and fine-tune) in addition to training only on Chromophore DB.

Table 4: Out-of-distribution ($\mu \pm 4\sigma$) property-conditional generation of 100 molecules on Chromophore DB. Generative Efficiency (% of valid, novel, and unique molecules) and Minimum MAE. See Section A.10 for Mean MAE results.

	Generative Efficiency				Properties - MinMAE			
	molWt	logP		QED	molWt	logP		QED
Condition	1538.00	-13.63	28.69	1.24*	1538.00	-13.63	28.69	1.24*
Trained on Chromophore DB (1000 epochs)								
STGG+$(k = 1)$	0.97	0.33	0.98	0.59	9.02	3.30	0.03	0.30
STGG+$(k = 100)$	0.88	0.25	0.82	0.81	5.24	6.02	8.02	0.25
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 1)$	0.91	0.71	0.92	0.75	0.41	8.10	0.12	0.05
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 100)$	0.89	0.71	0.94	0.83	0.74	0.89	7.03	0.01
Pre-trained on Zinc250K (50 epochs) and fine-tuned on Chromophore DB (100 epochs)								
STGG+$(k = 1)$	0.99	0.96	0.99	0.98	0.94	0.38	0.41	0.15
STGG+$(k = 100)$	1.00	0.96	0.93	1.00	2.37	0.35	0.42	0.09
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 1)$	1.00	0.95	0.97	1.00	0.47	0.66	0.01	0.02
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 100)$	1.00	0.92	0.98	0.99	13.19	0.45	0.18	0.01
Train data (closest sample)	-	-	-	-	1.40	9.62	0.17	0.01

†We removed low molWt and QED which are both impossible negative values.
*The value of 1.24 is improper; we calculate the MAE with respect to the max QED (0.948).

Results. The experiment results are shown in Table 4 (see Table 11 for the top-100 MAE). Pre-training on Zinc250K generally improves performance (Efficiency and MinMAE). For most properties, random guidance with filtering ($k > 1$) leads to the closest properties. However, for high logP, we obtain better property fidelity with no filtering ($k = 1$), indicating that the model struggles with property prediction on large out-of-distribution logP values.

4.5 Ablations and analyses

We provide ablations for the various **STGG+** components on OOD properties for Zinc (Table 5). Since we have 6 conditions (2 per property), results can be mixed; thus, we must average over the 6 conditions. Since every condition has different scaling, to average correctly, we use as metric the relative MAE (the difference divided by absolute value of the true value; averaged over all samples and over the 6 conditions). We compare Top-100, Top-100, Top-1 Relative MAEs. We train with 3 different seeds. We show that all the components improve the performance, except randomizing the node ordering (versus using a fixed node ordering) which has no significant impact. Variance across different seeds is low.

We provide visualizations of generated molecules in Appendix A.9.1, A.9.2, A.9.3. The visualizations show that STGG+ can generate molecules that satisfy OOD conditions while maintaining validity.

Table 5: Ablation for the out-of-distribution ($\mu \pm 4\sigma$) property-conditional generation on Zinc. Average Relative MAE over the 6 conditions (low/high molWt, logP, QED). The numbers correspond to Mean (Standard-deviation) over 3 seeds.

	Top-100 MAE	Top-10 MAE	Top-1 MAE (MinMAE)
STGG+ ($w \sim \mathcal{U}(-0.5, 2), k = 1$)	0.0247 (0.0029)	0.0040 (0.0005)	0.0010 (0.0001)
without randomize-order	0.0252 (0.0021)	0.0040 (0.0009)	0.0009 (0.0004)
without MLP (1-layer)	0.0260 (0.0029)	0.0046 (0.0007)	0.0012 (0.0005)
without the property-prediction loss	0.0264 (0.0017)	0.0052 (0.0004)	0.0012 (0.0001)
without improved architecture	0.0348 (0.0009)	0.0074 (0.0003)	0.0014 (0.0003)
different guidance ($w \sim \mathcal{U}(0.5, 2)$)	0.0974 (0.0081)	0.0567 (0.0169)	0.0332 (0.0186)
without guidance	0.1010 (0.0082)	0.0834 (0.0192)	0.0719 (0.0272)
with fixed guidance ($w = 1.5$)	0.1085 (0.0022)	0.0930 (0.0123)	0.0867 (0.0192)
without standardized properties	0.1309 (0.0287)	0.0322 (0.0062)	0.0190 (0.0045)
with self-criticism ($k = 2$)	0.0207 (0.0022)	0.0029 (0.0002)	0.0004 (0.0004)
with self-criticism ($k = 5$)	0.0203 (0.0039)	0.0023 (0.0007)	0.0006 (0.0004)
with self-criticism ($k = 10$)	0.0269 (0.0113)	0.0035 (0.0018)	0.0007 (0.0004)

4.6 Limitations

Molecule validity as measured by RDKit is widespread in machine learning, which is why we use it, but it is not a full picture of chemical validity. A molecule could be unstable or implausible by physical laws not implemented in RDKit; the field of generating *synthesizable* molecules still has many open questions. Another limitation is that many properties (e.g., molecule weight, logP) are not especially interesting from a chemistry perspective. We use them because they are widespread in the literature due to their ease of use through existing tools. This is why we included datasets with more complex properties, such as HOMO-LUMO gap and the categorical features for HIV, BBBP, and BACE. These either require slow simulations or fast machine learning models that may be inaccurate, especially in OOD settings. We believe that establishing new benchmarks with more complex properties is essential to the progress of the field. The property predictor of our approach may not be as accurate as property predictors made explicitly for this task, meaning our method may not always select the best molecules out of k choices, particularly in OOD scenarios; we found this to be the case for large OOD logP conditioning values.

5 Conclusion

In this paper, we demonstrated that with specific techniques, optimization, and architectural improvements, spanning tree-based graph generation (STGG) can be leveraged to generate high-quality and diverse molecules conditioned on both *in-distribution* and *out-of-distribution* properties. Our method achieves equal or superior performance on validity, novelty, uniqueness, closeness in distribution, and conditioning fidelity compared to competing approaches while being extremely efficient and fast. Using fewer molecules than required by online methods (RL/GFlowNet), we also obtain high multi-property-reward molecules in a one-shot manner from a pre-trained model.

Broader Impact

Discovering new molecules could have beneficial impact through the development of better drugs and hardware (television, computer screens, etc.). However, harmful molecules could also be discovered.

References

Sungsoo Ahn, Binghong Chen, Tianzhe Wang, and Le Song. Spanning tree-based graph generation for molecules. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 16, 17, 18, 19

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

Fan Bao, Min Zhao, Zhongkai Hao, Peiyao Li, Chongxuan Li, and Jun Zhu. Equivariant energy-guided sde for inverse molecular design. In *The eleventh international conference on learning representations*, 2022. 1, 2

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023. 8

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012. 8, 24

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015. 8

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001. 6, 20

Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019. 7

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023. 3, 19

Connor W Coley, Luke Rogers, William H Green, and Klavs F Jensen. Scscore: synthetic complexity learned from a reaction corpus. *Journal of chemical information and modeling*, 58(2):252–261, 2018. 6

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. 3, 19

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022. 3, 19

Jorg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem*, 3(10):1503, 2008. 7

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1:1–11, 2009. 6, 8

Yin Fang, Ningyu Zhang, Zhuo Chen, Lingbing Guo, Xiaohui Fan, and Huajun Chen. Domain-agnostic molecular generation with self-feedback. *arXiv preprint arXiv:2301.11259*, 2023. 2, 16

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems*, 35:21342–21357, 2022. 3, 6, 17, 20

Raj Ghugare, Santiago Miret, Adriana Hugessen, Mariano Phielipp, and Glen Berseth. Searching for high-value molecules using reinforcement learning and transformers. *arXiv preprint arXiv:2310.02902*, 2023. 3, 17

JS Griffith and LE Orgel. Ligand-field theory. *Quarterly Reviews, Chemical Society*, 11(4):381–393, 1957. 8

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 19

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3, 4, 19

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 1, 5, 6

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 16

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 7

Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022. 1, 2

Han Huang, Leilei Sun, Bowen Du, and Weifeng Lv. Learning joint 2-d and 3-d graph diffusion models for complete molecule generation. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. 1, 2

Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-Garcıa, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. In *International conference on machine learning*, pp. 14631–14653. PMLR, 2023. 1, 8

Yunhui Jang, Seul Lee, and Sungsoo Ahn. A simple and scalable representation for graph generation. In *The Twelfth International Conference on Learning Representations*, 2023. 1, 2, 3, 16

Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019. 7

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. 3

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International Conference on Machine Learning*, pp. 10362–10383. PMLR, 2022. 1, 7, 8

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. *arXiv preprint arXiv:2302.03596*, 2023. 1

Joonyoung Francis Joung, Minhi Han, Minseok Jeong, and Sungnam Park. DB for chromophore. 4 2020. doi: 10.6084/m9.figshare.12045567.v2. URL `https://figshare.com/articles/dataset/DB_for_chromophore/12045567`. 6, 9, 18, 19

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3, 19

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 8

Mario Krenn, Qianxiang Ai, Senja Barthel, Nessa Carson, Angelo Frei, Nathan C Frey, Pascal Friederich, Théophile Gaudin, Alberto Alexander Gayle, Kevin Maik Jablonka, et al. Selfies and the future of molecular string representations. *Patterns*, 3(10), 2022. 3, 16

Kisoo Kwon, Kuhwan Jeong, Junghyun Park, Hwidong Na, and Jinwoo Shin. String-based molecule generation via multi-decoder vae. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023. 1, 2, 8

Greg Landrum, Paolo Tosco, Brian Kelley, Ricardo Rodriguez, David Cosgrove, Riccardo Vianello, sriniker, gedeck, Gareth Jones, Nadine Schneider, Eisuke Kawashima, Dan Nealschneider, Andrew Dalke, Matt Swain, Brian Cole, Samo Turk, Aleksandr Savelev, Alain Vaucher, Maciej Wójcikowski, and jasondbiggs. rdkit/rdkit, 2024. URL `https://doi.org/10.5281/zenodo.11396708`. 2, 6, 18

Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, pp. 18872–18892. PMLR, 2023. 7

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. *arXiv preprint arXiv:2203.15386*, 2022. 8

Gang Liu, Jiaxin Xu, Tengfei Luo, and Meng Jiang. Graph diffusion transformers for multi-conditional molecular generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=cfrDLD1wfO`. 6, 7, 19, 23

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 3

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 3, 19

Yanchen Luo, Sihang Li, Zhiyuan Liu, Jiancan Wu, Zhengyi Yang, Xiangnan He, Xiang Wang, and Qi Tian. Text-guided diffusion model for 3d molecule generation. 2024. 2

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, pp. 1–11, 2024. 1, 2

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024. 1

Giangiacomo Mercatali, Yogesh Verma, Andre Freitas, and Vikas Garg. Diffusion twigs with loop guidance for conditional graph generation. *arXiv preprint arXiv:2410.24012*, 2024. 2

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016. 8

Harry L Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of chemical documentation*, 5(2):107–113, 1965. 6, 20

Mohit Pandey, Gopeshh Subbaraj, and Emmanuel Bengio. Gflownet pretraining with inexpensive rewards. *arXiv preprint arXiv:2409.09702*, 2024. 17

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023. 19

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023. 19

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alan Aspuru-Guzik, and Alex Zhavoronkov. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Frontiers in Pharmacology*, 2020. 6

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58 (9):1736–1741, 2018. 7

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 3, 19

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 4

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014. 6, 8, 18, 19

Guillaume Sanchez, Honglu Fan, Alexander Spangher, Elad Levi, Pawan Sasanka Ammanamanchi, and Stella Biderman. Stay on topic with classifier-free guidance. *arXiv preprint arXiv:2306.17806*, 2023. 5

Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131, 2018. 1, 2

Huajie Shao, Shuochao Yao, Dachun Sun, Aston Zhang, Shengzhong Liu, Dongxin Liu, Jun Wang, and Tarek Abdelzaher. Controlvae: Controllable variational autoencoder. In *International conference on machine learning*, pp. 8655–8664. PMLR, 2020. 8

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. 3, 19

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 2, 16

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 2, 16

Yuxuan Song, Jingjing Gong, Yanru Qu, Hao Zhou, Mingyue Zheng, Jingjing Liu, and Wei-Ying Ma. Unified generative modeling of 3d molecules via bayesian flow networks. *arXiv preprint arXiv:2403.15441*, 2024. 2

Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015. 6, 18, 19

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 3, 19

Qiming Sun, Xing Zhang, Samragni Banerjee, Peng Bao, Marc Barbry, Nick S Blunt, Nikolay A Bogdanov, George H Booth, Jia Chen, Zhi-Hao Cui, et al. Recent developments in the pyscf program package. *The Journal of chemical physics*, 153(2), 2020. 2

Rylee Thompson, Boris Knyazev, Elahe Ghalebi, Jungtaek Kim, and Graham W Taylor. On evaluation metrics for graph generative models. *arXiv preprint arXiv:2201.09871*, 2022. 1

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 3

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3, 19

Yogesh Verma, Samuel Kaski, Markus Heinonen, and Vikas Garg. Modular flows: Differential molecular generation. *Advances in neural information processing systems*, 35:12409–12421, 2022. 2

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734*, 2022. 1, 3, 7

Conghao Wang, Hiok Hian Ong, Shunsuke Chiba, and Jagath C Rajapakse. Gldm: hit molecule generation with constrained graph latent diffusion model. *Briefings in Bioinformatics*, 25(3):bbae142, 2024. 17

David Weininger. 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. 1, 2, 16

Juan-Ni Wu, Tong Wang, Yue Chen, Li-Juan Tang, Hai-Long Wu, and Ru-Qin Yu. t-smiles: a fragment-based molecular representation framework for de novo ligand design. *Nature Communications*, 15(1):4993, 2024. 1, 2

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018. 6, 18, 19

Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. Mars: Markov molecular sampling for multi-objective drug discovery. *arXiv preprint arXiv:2103.10432*, 2021. 7

Minkai Xu, Alexander S Powers, Ron O Dror, Stefano Ermon, and Jure Leskovec. Geometric latent diffusion models for 3d molecule generation. In *International Conference on Machine Learning*, pp. 38592–38610. PMLR, 2023. 1, 2

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in neural information processing systems*, 32, 2019. 8

Atakan Yüksel, Erva Ulusoy, Atabey Ünlü, and Tunca Doğan. Selformer: molecular representation learning via selfies language models. *Machine Learning: Science and Technology*, 4(2):025035, 2023. 2, 16

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019. 3, 19

Shuo Zhang, Yang Liu, and Lei Xie. Molecular mechanics-driven graph neural network with multiplex graph for molecular structures. *arXiv preprint arXiv:2011.07457*, 2020. 8

A Appendix

A.1 1D vs 2D representations

There are many ways to represent molecules in the context of molecular generation. The most popular methods are autoregressive models on 1D strings and diffusion (Song & Ermon, 2019; Ho et al., 2020; Song et al., 2020) models on 2D graphs. We highlight the main distinction between the two representations below in the context.

Let D be the size of the training dataset, n be the number of atoms in a given molecule, d is the embedding size, and b is the number of bond types.

Diffusion models on 2D graphs:

- $G = (X, A)$ where the vertices X contains the list of atoms (size: $[n, d]$) and A is the adjacency matrix of the edges (size: $[n, n, b]$) for each bond type.
- A is an extremely sparse matrix with many zero elements
- Input space is $\mathcal{O}(nd + bn^2)$; unless using low-rank projections, the number of parameters must scale proportionally to this amount
- Typically use diffusion models (or related methods) given the large number of steps it would take to generate X and A autoregressively
- Equivariant Graph Neural Networks (E-GNNs) are generally used to ensure a unique representation for a given molecule
- Although it has a single representation per molecule, multiple random noises per graph are needed due to diffusion; thus, sample complexity is $\mathcal{O}(Dn_{noise})$

Autoregressive models on 1D strings:

- X (size: $[L, d]$) is a string containing the molecule where L is proportional to n
- The string starts from a random atom and traverses the 2D molecular graph
- Input space is $\mathcal{O}(nd)$; this makes it efficient to process
- Typically use autoregressive models (e.g., Transformers) as it scales well
- We can either 1) fix the ordering in some way to make representation unique, or 2) use random orderings as data augmentation with a non-unique representation for a given molecule; thus, sample complexity is $\mathcal{O}(Dn_{augments})$

As can be seen, both methods have similar sample complexity, but 1D strings are much more compressed representations, leading to less space and parameters and, thus, increased potential for scalability. Furthermore, recent results show that 1D strings are as competitive as 2D molecular graph methods for unconditional molecule generation Jang et al. (2023); Fang et al. (2023) and property prediction Yüksel et al. (2023). In the end, both representations contain as much information. Thus, the choice is a matter of preference. 1D strings are easier to scale and can make good use of the power of Transformers; hence, we focus on this type of representation.

A.2 Spanning Tree compared to other 1D string-based representations

The most popular choice of string-based representation is SMILES (Weininger, 1988). SMILES is extremely versatile, allowing the representation of any molecule. However, for the purpose of generative models, trying to generate SMILES strings directly can quickly lead to many invalid molecules. Graph-based diffusion methods encounter the same issue. Recently, methods have been created to prevent the creation of invalid molecules: Spanning Tree (Ahn et al., 2021) and SELFIES (Krenn et al., 2022). Below, we describe in detail the differences between all three methods.

SMILES:

- Massive vocabulary allows the representation of every aspect of molecules
- There are many ways of representing a single molecule
- Begin-branch token "(" to deviate from the main path and close branch token ")"
- Pointer token i to indicate both the beginning and end of rings

SELFIES:

- Restricted SMILES vocabulary
- Prevent invalid molecules through a carefully designed context-free grammar:
  - Atoms and bonds are combined into single tokens (with other aspects such as charge and number of hydrogen atoms) so that we cannot have an atom without a bond and a bond without an atom
  - Hard-designed rules for maximum valencies of specific elements (slightly more permissible than octet rule, but cannot handle every case)
  - Keep track of valencies; ignore future tokens in the current branch if there is not enough valency left and reduce bond order if needed
  - There is an open-branch token Branch-i and close-ring token ring-i where i specifies the number of future tokens in the branch and how many backward steps (in tokens) are needed to reach the ring closure; this ensures that all branches and tokens are not left opened

Spanning-Tree:

- Restricted SMILES vocabulary
- Begin and end branch tokens, with ring start and i-th ring end tokens
- Similarity-based output layer to determine the probabilities of ring ends and input embedding injection for how many rings are opened
- Prevent invalid molecules through masking of tokens before softmax:
  - Masking of invalid tokens due to impossible valencies (atoms, ring-start, and branch-start when not enough valency is left) based on the valencies of the training data
  - Masking of invalid next tokens (atom after atom, bond after bond, ring-i end when there are less than i ring start tokens)
  - Force branch ending through masking when getting too close to maximum sequence length to prevent unfinished molecules (new)

As can be seen, SMILES has such a large vocabulary that each molecule can be represented in completely different ways, and its main problem for generative models is that many token choices lead to invalid molecules (e.g., two bonds, incorrect valencies, unfinished branches, or rings, etc.).

SELFIES prevents invalid molecules through its smart, context-free grammar. However, work by Gao et al. (2022) and Ghugare et al. (2023) found that while SELFIES prevent invalid molecules, it makes exploration more difficult and reduces the performance of generative models (in terms of obtaining high-reward samples, i.e., molecules with desired properties). A significant challenge for the generative models based on SELFIES is the need to pre-define the number of tokens contained in a branch (a deviation from the main path in a 1D string) and count backward the number of tokens required to reach the beginning of the ring (starting from the end). This requires extensive planning and counting by the generative model.

On the other hand, Spanning-tree use clever masking of incorrect tokens to prevent invalid molecules and doing so does not require the model to do significant planning-in-advance and counting when selecting the next token (including the i-th ring end tokens which require no counting due to the similarity-based prediction). Note that Wang et al. (2024) and Pandey et al. (2024) also use a similar masking to the one devised by Ahn et al. (2021) to improve validity, thus it can be applied to other representations.

A.3 Datasets details and canonicalization

QM9 (Ramakrishnan et al., 2014) has 21 atom tokens: CH3, C, O, CH2, CH, NH, N, N-, NH+, OH, NH2, F, NH3+, O-, NH2+, N+, C-, CH-, NH3, OH2, CH4. The maximum length is 37. The dataset has 133886 molecules with around 10% of the molecules in the test set and 5% in the validation set.

Zinc250K (Sterling & Irwin, 2015) has 34 atom tokens: CH3, C, CH, N, S, CH2, O, NH, NH+, NH2, NH2+, NH3+, OH, Cl, O-, N-, F, Br, N+, S-, I, SH, P, NH-, O+, OH+, S+, CH2-, CH-, SH+, PH, PH+, P+, PH2. The maximum length is 136. The dataset has 250k molecules with around 10% of the molecules in the test set and 5% in the validation set.

BBBP (Wu et al., 2018) has 31 atom tokens: CH, C, F, CH2, N, S, CH3, O, OH, NH2, Cl, NH, OH2, Br, O-, N+, Na, Cl-, H+, C-, Na+, NH+, NH3+, Br-, P, N-, SH, CH2-, CH-, I, B. The maximum length is 186. The dataset has 862 molecules with around 20% of the molecules in the test set and 20% in the validation set.

BACE (Wu et al., 2018) has 20 atom tokens: F, C, N, CH, NH+, CH2, NH2, O, Cl, S, CH3, NH, OH, NH2+, Br, O-, NH3+, N+, N-, I. The maximum length is 161. The dataset has 1332 molecules with around 20% of the molecules in the test set and 20% in the validation set.

HIV (Wu et al., 2018) has 76 atom tokens: CH3, C, O, CH2, N, NH2, CH, N+, NH2+, I, NH, Br, Se, OH, S, O-, Br-, SH, Cl, I-, S+, Zn-2, OH+, N-, NaH, PH, Ir-3, Cl-, NH3, F, P, BrH, C-, Co-2, Cu-4, As, B-2, Sn, ClH, Rh-4, O+, S-, Pt, Fe-2, B, U+2, Pd-2, Fe-3, Pt-2, Pt+2, Si, P+, IH2, Fe, SiH, Cl+3, Ge, NH+, Zr, K+, AlH3-, IH, KH, Mn+, Fe-4, Cu-3, Ni-4, LiH, Co-3, Pd-3, Fe+2, Ga-3, CH2-, U, Mn, Co-4. The maximum length is 193. The dataset has 2372 molecules with around 20% of the molecules in the test set and 20% in the validation set.

Chromophore DB (Joung et al., 2020) has 46 atom tokens: CH, C, N, CH3, CH2, O, N+, B-, F, S, OH, NH, Cl, NH2, P, O+, Si, O-, Se, C+, B, Br, I, NH+, NH2+, N-, S+, SiH, C-, Na, Sn, NH3+, S-, Si-, P-, Cl+3, I-, BH3-, P+, BH, CH4, NH-, SH, Ge, Te, Na+. The maximum length is 511. The dataset has 6810 molecules with around 5% of the molecules in the test set and 5% in the validation set.

Note that we base the maximum length on the largest SMILES string after being transformed with the Spanning tree tokenizer.

A.3.1 Canonicalization

Similar to STGG (Ahn et al., 2021), we use explicit Hydrogen atoms (with no implicit Hydrogen atom) in the tokens. This is an arbitrary choice. After generation, we always transform back to canonical SMILES using RDKit (Landrum et al., 2024). Note that RDKit may change the number of Hydrogen atoms based on its own rule-set. All our molecule figures are based on RDKit so they reflect the molecules after SMILES canonicalization by RDKit.

Here is an example below.

Training SMILES: C[C@@]12C=CC(=O)C=C1CC[C@@H]
1[C@@H]3CC[C@](O)(C(=O)COP(=O)([O-])[O-])[C@]3(C)C[C@@H](O)[C@H]12

STGG tokenized: [bos]C-C[bor][bor]-C=C-C(=O)(-C=C(-[eor0])(-C-C-CH[bor]-CH[bor]-C-C-C(-O)(-C(=O)(-C-O-P(=O)(-O-)(-O-)))(-C(-[eor3])(-C)(-C-CH(-O)(-CH(-[eor1])(-[eor2])))))))[eos]

Canonical SMILES: CC12C=CC(=O)C=C1CCC1C2C(O)
CC2(C)C1CCC2(O)C(=O)COP(=O)([O-])[O-]

A.3.2 Any-property masking

For any-property masking, we show an example below. Assume that there are $T = 5$ properties. Each time we sample a training molecule, we choose a random number t of properties to mask uniformly between 0 and $T = 5$. Assuming that $t = 3$, we create the masking vector: [1, 1, 1, 0, 0, 0]. Then, we randomly shuffle the masking vector, leading to: [1, 0, 0, 1, 0, 1]. Then, we mask the properties with a masking value of 1.

A.4 Hyperparameters

The original STGG (Ahn et al., 2021) used the AdamW optimizer (Loshchilov & Hutter, 2017; Kingma & Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, no weight decay, a fixed learning rate of 2e-4 and batch-size 128. The Transformer architecture had 3 layers, dropout 0.1, 8 attention heads, and embedding size 1024. They processed only one property with a single linear layer.

STGG+ uses the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 0.1. The Transformer architecture has 3 layers, no dropout, 16 attention heads, SwiGLU (Hendrycks & Gimpel, 2016; Shazeer, 2020) with expansion scale of 2, no bias term (Chowdhery et al., 2023), Flash Attention (Dao et al., 2022; Dao, 2023), RMSNorm (Zhang & Sennrich, 2019), Rotary embeddings (Su et al., 2024), residual-path weight initialization (Radford et al., 2019). When not using random guidance, we use classifier-free guidance with a guidance parameter $w = 1.5$, where $w = 1$ means no guidance.

For QM9 (Ramakrishnan et al., 2014), we train for 50 epochs with batch size 512, learning rate 1e-3, max length 150. For Zinc250K (Sterling & Irwin, 2015), we train for 50 epochs with batch size 512, learning rate 1e-3, max length 250. For HIV, BACE, and BBBP (Wu et al., 2018), we train for 10K epochs (same as done by Liu et al. (2024)), since these are small datasets, with batch size 128, learning rate 2.5e-4, max length 300.

For Chromophore DB (Joung et al., 2020), we train for 1000 epochs with batch size 128, learning rate 2.5e-4, max length 600. For the pre-training on Zinc250K and fine-tuning on Chromophore-DB: we pre-train with batch size 512, learning rate 1e-3, and max length 600 for 50 epochs and fine-tune with batch size 128, learning rate 2.5e-4, and max length 600 for 100 epochs.

We generally use 1 to 4 A-100 GPUs to train the models. Training takes less than a day. Note that we use a higher max length than the data max length (generally around 25-50%) to ensure that we can adequately generate molecules with out-of-distribution properties that could be bigger than usual.

For pretraining and then fine-tune, there are two ways to preprocess the properties: we can either standardize them with respect to the pre-training or the fine-tuning datasets. Standardizing with respect to the pre-training dataset can lead to extreme values in the fine-tuning (e.g., 4 standard deviation in Chromophore's MolWt is 15 standard-deviation in Zinc250K's MolWt). Hereby, to reduce the gap between pre-trained and fine-tuned conditioning values, we preprocess the properties by standardizing with respect to the fine-tune dataset properties during both pre-training and fine-tuning.

A.5 Alternative architectures considered

In this work, we enhance the Transformer architecture used by Ahn et al. (2021) using recent developments in Large Language Models (LLMs). Although powerful, the Transformer architecture with self-attention (Vaswani et al., 2017) is quadratic in context length, which means that the time and memory increase significantly when dealing with long-context length.

In addition to improvements on Transformer, new architectures such as Mamba (Gu & Dao, 2023), Hyena, (Poli et al., 2023) or RWKV (Peng et al., 2023) have appeared, which are sub-quadratic with respect to context-length, allowing them to handle long-context length better. We initially considered some of these architectures to improve inference speed. However, it is hard to synthesize and manufacture molecules of substantial sizes. Thus, the context length is generally quite limited (e.g., the largest molecule on Chromophore has 511 tokens, while modern LLMs have a context length of at least 4096). As long as the context length is less or equal to 2048, FlashAttention (Dao et al., 2022) is fast enough that there is no inference speed benefit for using Mamba (Gu & Dao, 2023).

A.6 Property prediction

Table 6: Property prediction on the test set using **STGG+** or a Random Forest (Breiman, 2001) predictor/classifier using the Morgan Fingerprint (Morgan, 1965) as done by Gao et al. (2022).

		Accuracy	Mean squared error (MSE)					
Task	Method	HIV	QED	MolWt	logP	SAS	SCS	Gap
QM9	STGG+	-	0.0010	0.0018	0.0012	-	-	-
QM9	Random Forest	-	0.2665	0.6124	0.2014	-	-	-
Zinc250K	STGG+	-	0.0008	0.0005	0.0005	-	-	-
Zinc250K	Random Forest	-	0.4077	0.4209	0.3907	-	-	-
HIV	STGG+	0.8463	-	-	-	0.0268	0.0216	-
HIV	Random Forest	0.7263	-	-	-	0.3605	0.4672	-
BACE	STGG+	0.9551	-	-	-	0.0126	0.0070	-
BACE	Random Forest	0.8165	-	-	-	0.1773	0.3948	-
BBBP	STGG+	0.8743	-	-	-	0.0354	0.0314	-
BBBP	Random Forest	0.8057	-	-	-	0.3152	0.4740	-
QM9 (Reward)	STGG+	-	-	0.0009	0.0005	0.0021	-	0.0032
QM9 (Reward)	Random Forest	-	-	0.6122	0.2015	0.2114	-	0.1459

A.7 Unconditional Generation

Table 7: Molecular graph generation performance on QM9.

Method	Valid (%) (↑)	Unique (%) (↑)	Novel (%) (↑)	FCD (↓)	Scaf. (↑)	SNN (↑)	Frag. (↑)
Domain-agnostic graph generative models							
EDP-GNN	47.52	**99.25**	86.58	2.680	0.3270	0.5265	0.8313
GraphAF	74.43	88.64	86.59	5.625	0.3046	0.4040	0.8319
GraphDF	93.88	98.58	**98.54**	10.928	0.0978	0.2948	0.4370
GDSS	95.72	98.46	86.27	2.900	0.6983	0.3951	0.9224
DiGress	98.19	96.67	25.58	0.095	0.9353	0.5263	0.0023
DruM	99.69	96.90	24.15	0.108	**0.9449**	0.5272	0.9867
GraphARM	90.20	-	-	1.220	-	-	-
GEEL	**100.0**	96.08	22.30	0.089	0.9386	0.5161	0.9891
Molecule-specific generative models							
CharRNN	99.57	-	-	**0.087**	0.9313	0.5162	0.9887
CG-VAE	**100.0**	-	-	1.852	0.6628	0.3940	0.9484
MoFlow	91.36	98.65	94.72	4.467	0.1447	0.3152	0.6991
STGG	**100.0**	96.76	72.73	0.585	0.9416	**0.9998**	**0.9984**
Unconditional (masking all properties)							
STGG+	**100.0**	97.17	74.41	0.089	0.9265	0.5179	0.9877
Conditional (using test properties)							
STGG+ (k=1)	**100.0**	97.63	75.99	0.134	0.8906	0.5004	0.9799
STGG+ (k=5)	**100.0**	96.86	74.18	0.166	0.9050	0.5039	0.9860

Table 8: Molecular graph generation performance on Zinc250K.

Method	Valid (%) (↑)	Unique (%) (↑)	Novel (%) (↑)	FCD (↓)	Scaf. (↑)	SNN (↑)	Frag. (↑)
Domain-agnostic graph generative models							
EDP-GNN	63.11	99.79	**100.00**	16.737	0.0000	0.0815	0.0000
GraphAF	68.47	98.64	99.99	16.023	0.0672	0.2422	0.5348
GraphDF	90.61	99.63	**100.00**	33.546	0.0000	0.1722	0.2049
GDSS	97.01	99.64	**100.00**	14.656	0.0467	0.2789	0.8138
DiGress	94.99	99.97	99.99	3.482	0.4163	0.3457	0.9679
DruM	98.65	99.97	99.98	2.257	0.5299	0.3650	0.9777
GraphARM	88.23	-	-	16.260	-	-	-
GEEL	99.31	99.97	99.89	0.401	0.5565	0.4473	0.9920
Molecule-specific generative models							
CharRNN	96.95	-	-	0.474	0.4024	0.3965	**0.9988**
CG-VAE	**100.0**	-	-	11.335	0.2411	0.2656	0.8118
MoFlow	63.11	99.99	**100.00**	20.931	0.0133	0.2352	0.7508
STGG	**100.0**	99.99	99.89	**0.278**	**0.7192**	**0.4664**	0.9932
Unconditional (masking all properties)							
STGG+	**100.0**	99.99	99.94	0.395	0.5657	0.4316	0.9925
Conditional (using test properties)							
STGG+ (k=1)	**100.0**	**100.0**	99.98	0.514	0.5302	0.4099	0.9917
STGG+ (k=5)	**100.0**	**100.0**	**100.0**	0.562	0.5491	0.4176	0.9909

A.8 Full Table of conditional generation on HIV, BBBP, and BACE

Table 9: Full table: Conditional generation of 10K molecular compounds on HIV, BBBP, and BACE.

Tasks	Model	Distribution Learning					Condition Control	
		Validity ↑	Coverage* ↑	Diversity ↑	Similarity ↑	Distance ↓	Synthe. MAE ↓	Property Acc.* ↑
Synth. & BACE	DiGress	0.351	8/8	0.886	0.694	24.656	2.068	0.506
	DiGress v2	0.355	8/8	0.881	0.703	25.327	2.337	0.511
	GDSS	0.288	4/8	0.876	0.271	46.754	1.642	0.504
	MOOD	0.995	8/8	0.890	0.259	44.239	1.885	0.506
	Graph DiT	0.867	8/8	0.824	0.875	7.046	0.400	0.913
	Graph GA	1.000	8/8	0.859	0.981	7.410	0.963	0.469
	MARS	1.000	8/8	0.834	0.883	6.792	1.012	0.518
	LSTM-HC	0.997	8/8	0.815	0.798	17.559	0.921	0.582
	JTVAE-BO	1.000	6/8	0.668	0.728	30.470	0.992	0.463
	STGG**	1.000	8/8	0.824	0.979	3.824	0.453	0.949
	STGG+ $(k=1)$	1.000	8/8	0.829	0.979	3.796	0.238	0.912
	STGG+ $(k=5)$	1.000	8/8	0.826	0.979	3.802	0.178	0.926
	Train data	1.000	8/8	0.819	0.981	3.837	0.003†	0.991
	Test data	1.000	**7/8***	0.824	1.000	0.000	0.002†	**0.817***
Synth. & BBBP	DiGress	0.696	9/10	0.910	0.681	18.692	2.366	0.654
	DiGress v2	0.689	9/10	0.911	0.634	19.450	2.269	0.653
	GDSS	0.622	3/10	0.842	0.267	39.944	1.379	0.504
	MOOD	0.801	9/10	0.927	0.172	34.251	2.028	0.490
	Graph DiT	0.847	9/10	0.886	0.933	11.851	0.355	0.942
	Graph GA	1.000	9/10	0.895	0.951	10.166	1.208	0.302
	MARS	1.000	8/10	0.864	0.770	10.979	1.225	0.519
	LSTM-HC	0.999	8/10	0.888	0.893	16.390	0.997	0.559
	JTVAE-BO	1.000	5/10	0.746	0.582	33.575	1.162	0.496
	STGG**	1.000	9/10	0.891	0.916	11.736	0.982	0.754
	STGG+ $(k=1)$	1.000	10/10	0.888	0.937	9.859	0.466	0.867
	STGG+ $(k=5)$	1.000	9/10	0.887	0.936	10.101	0.381	0.900
	Train data	1.000	8/10	0.883	0.957	9.890	0.017†	0.996
	Test data	1.000	**10/10***	0.880	0.998	0.000	0.018†	**0.806***
Synth. & HIV	DiGress	0.438	22/29	0.919	0.856	13.041	1.922	0.534
	DiGress v2	0.505	24/29	0.919	0.848	13.400	1.593	0.533
	GDSS	0.693	4/29	0.782	0.103	45.342	1.252	0.483
	MOOD	0.288	29/29	0.928	0.136	32.352	2.314	0.511
	Graph DiT	0.766	28/29	0.897	0.958	6.022	0.309	0.978
	Graph GA	1.000	28/29	0.899	0.966	4.442	0.984	0.604
	MARS	1.000	26/29	0.876	0.652	7.289	0.969	0.646
	LSTM-HC	0.999	13/29	0.909	0.915	7.466	0.948	0.674
	JTVAE-BO	1.000	3/29	0.806	0.417	41.977	1.236	0.485
	STGG**	1.000	27/10	0.899	0.961	4.558	0.442	0.950
	STGG+ $(k=1)$	1.000	27/29	0.896	0.970	4.075	0.314	0.876
	STGG+ $(k=5)$	1.000	24/29	0.897	0.9700	4.317	0.229	0.905
	Train data	1.000	27/29	0.895	0.970	4.019	0.018†	0.999
	Test data	1.000	**21/29***	0.895	0.998	0.074	0.015†	**0.726***

*The classifier from Liu et al. (2024) (used in the last column) has limited accuracy on the test set; thus, any *Property Acc.* above the **test data accuracy** is not indicative of better quality. Similarly, atom coverage is not 100% on test data; thus, any coverage above the **test set coverage** does not indicate better performance.
**STGG with categorical embedding, missing indicators, random masking, and extra symbol for compounds.
†The dataset properties are rounded to two decimals hence MAE is not exactly zero.

A.9 Min-MAE or Max-Reward molecules generated by STGG+

A.9.1 Zinc OOD

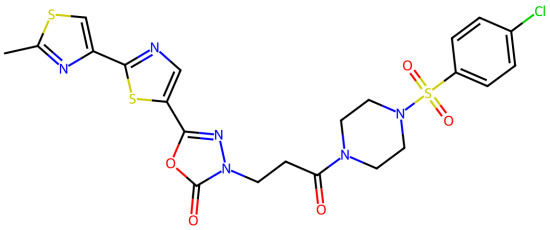

Figure 3: Conditioning on molWt=580.00

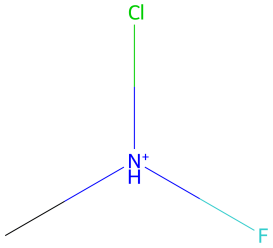

Figure 4: Conditioning on molWt=84.0008

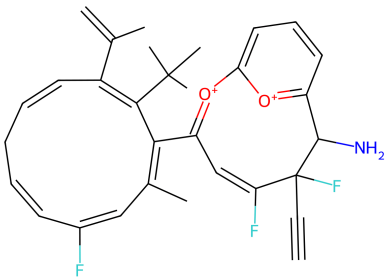

Figure 5: Conditioning on logP=8.1940

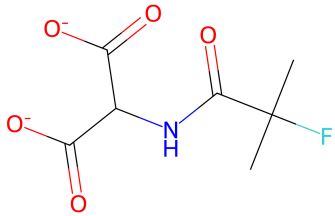

Figure 6: Conditioning on logP=-3.2810

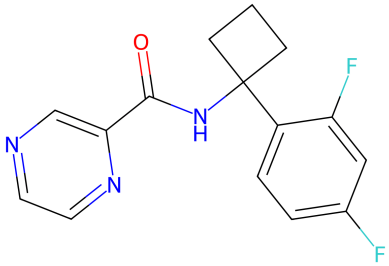

Figure 7: Conditioning on QED=1.2861

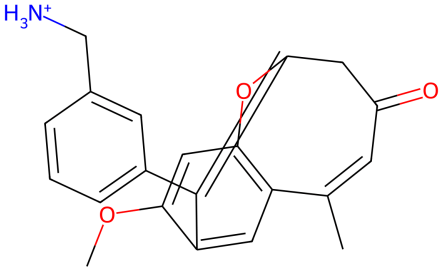

Figure 8: Conditioning on QED=0.1778 (which means low drug-likeness and less "chemical beauty" (Bickerton et al., 2012))

A.9.2 QM9 Reward Maximization

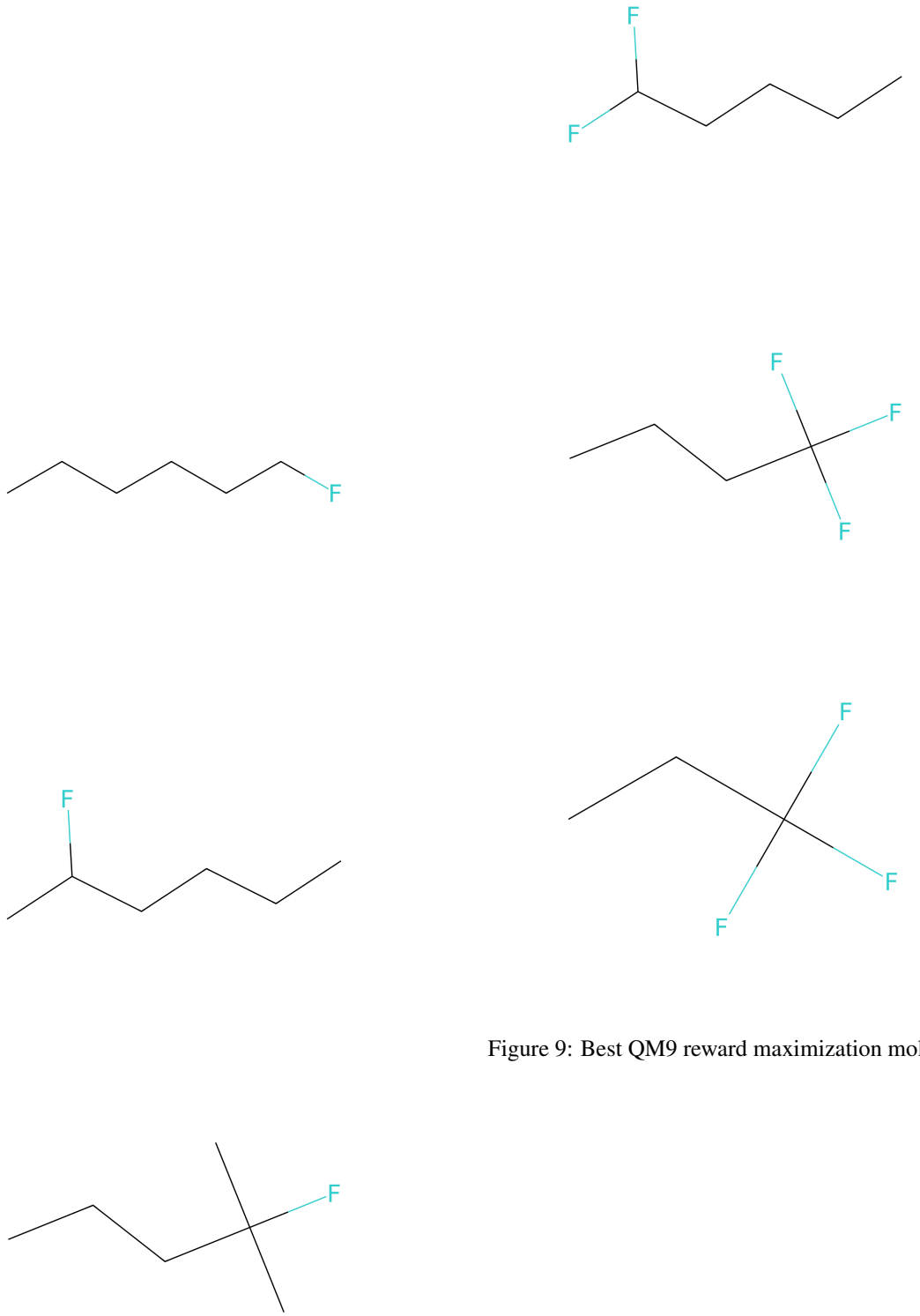

Figure 9: Best QM9 reward maximization molecules

A.9.3 Chromophore OOD

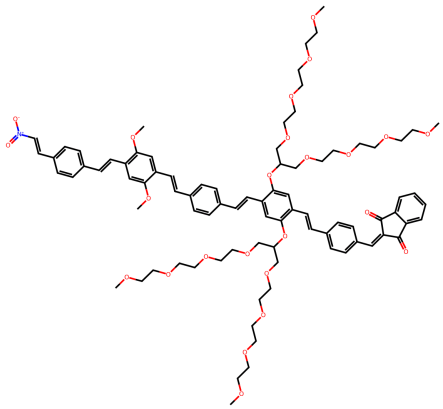

Figure 10: Conditioning on molWt=1538.00

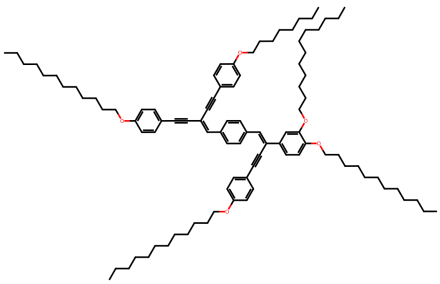

Figure 11: Conditioning on logP=28.6915

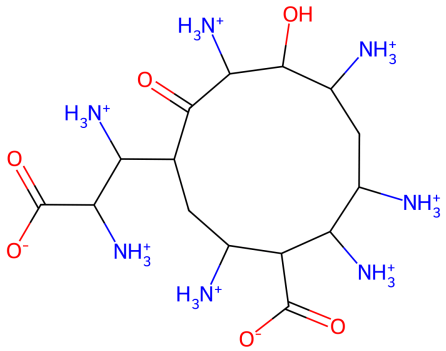

Figure 12: Conditioning on logP=-13.6292

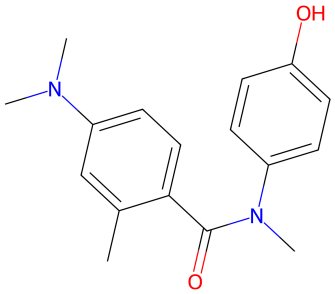

Figure 13: Conditioning on QED=1.2355

A.10 OOD Tables with Mean MAE over the top-100 molecules

Table 10: Out-of-distribution ($\mu \pm 4\sigma$) property-conditional generation of 2K molecules on Zinc250K.

	Properties - top-100 Mean MAE					
	molWt		logP		QED	
Condition	84	580	-3.2810	8.1940	0.1778	1.2861*
STGG**	18.248	5.559	1.204	1.548	0.206	0.022
STGG+$(k = 1)$	0.790	1.389	0.018	0.900	0.003	0.561
STGG+$(k = 5)$	1.289	1.503	0.021	3.710	0.003	0.571
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 1)$	1.533	2.088	0.040	0.285	0.005	0.060
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 5)$	1.285	1.104	0.022	0.803	0.004	0.042

*The value is improper; we condition on 1.2861 but calculate the MAE with respect to the maximum QED (0.948).

**STGG with missing indicators, and random masking.

Table 11: Out-of-distribution ($\mu \pm 4\sigma$) property-conditional generation of 100 molecules on Chromophore DB. We removed the low molWt and QED which are both impossible negative values.

	Properties - Mean MAE			
	molWt	logP		QED
Condition	1538.00	-13.63	28.69	1.24*
Trained on Chromophore DB (1000 epochs)				
STGG+$(k = 1)$	256.6	11.1	5.1	0.6
STGG+$(k = 100)$	562.3	11.0	16.3	0.5
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 1)$	805.6	15.4	11.1	0.5
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 100)$	609.5	8.8	14.3	0.2
Pre-trained on Zinc250K (50 epochs) and fine-tuned on Chromophore DB (100 epochs)				
STGG+$(k = 1)$	294.9	8.4	6.1	0.5
STGG+$(k = 100)$	401.9	5.6	13.1	0.4
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 1)$	543.0	14.6	12.7	0.5
STGG+$(w \sim \mathcal{U}(-0.5, 2), k = 100)$	416.5	6.1	13.0	0.2

*The value of 1.24 is improper; we calculate the MAE with respect to the maximum QED (0.948).

In Table 11, STGG+ with pre-training and fine-tuning generally performs slightly better than regular training. Random guidance is helpful for high QED.

A.11 Algorithms

Algorithm 1 STGG+ Training

Require: Dataset $\mathcal{D} = \{(x_i, y_i)\}$ where x_i is a molecule and $y_i \in \mathbb{R}^D$ are its properties
Require: Transformer model f_θ
while not converged **do**
  Sample batch $(x_1, \ldots, x_B)$ and properties $(y_1, \ldots, y_B)$
  for each molecule x_i in batch **do**
    Tokenize x_i into sequence $(t_1, \ldots, t_L)$
    Mask a random subset of m properties from $y_i = (y_{i1}, ..., y_{iD})$, where $m \sim \text{Uniform}(0, D)$
    Compute $(h_1, \ldots, h_L)$, where $h_j \leftarrow f_\theta(t_j | y_i, t_1, ..., t_{j-1})$
    Compute the cross-entropy loss $\mathcal{L}_{\text{CE}}$
    Compute the auxiliary property prediction loss $\mathcal{L}_{\text{prop}} = \frac{1}{2} \| f_\theta^{\text{pred}}(h_L) - y_i \|_2^2$
  Update θ using gradient descent on $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{prop}}$

Algorithm 2 STGG+ Sampling with Self-Criticism

Require: Target properties $y_{\text{target}} \in \mathbb{R}^T$
Require: Guidance strength w, number of candidates K, max length L_{max}
Require: Transformer model f_θ
Generate K candidate molecules:
for $k = 1$ to K **do**
  Initialize the sequence; $t_1 \leftarrow$ [BOS]
  Sample guidance scale $w \sim \mathcal{U}(0.5, 2)$ (optional; otherwise w=1 means no guidance)
  for $j = 2$ to L_{max} **do**
    Compute conditional logits $z_c = f_\theta(t_j | y_{\text{target}}, t_1, ..., t_{j-1})$
    Compute unconditional logits $z_u = f_\theta(t_j | \emptyset, t_1, ..., t_{j-1})$
    Apply Classifier-Free Guidance (CFG): $z = w z_c + (1 - w) z_u$
    Mask invalid tokens (valency violations, syntax errors, ring overflow, etc.)
    Sample the next token $t_j \sim \text{Softmax}(z)$
    if $t_j \leftarrow$ [EOS] **then**
      break
  Store the molecule s_k
Self-criticism:
for each candidate s_k **do**
  Process s_k through f_θ with empty properties to predict $\hat{y}_k$
  Compute distance $d_k = \| \hat{y}_k - y_{\text{target}} \|_2^2$
return s_j where $j \leftarrow \arg\min_k d_k$ {Select the best candidate}

A.12 Masking

Here we describe the original STGG masking algorithm and the improved **STGG+** masking algorithm.

Algorithm 3 STGG Masking algorithm

Require: maximum number of rings $ring_{max}$

if token is an atom **then**
  $allowed_{next} = [bonds, branch_{start}, ring_{start}]$
  if all branches are closed **then**
    $allowed_{next}.append([EOS])$
  else
    $allowed_{next}.append([branch_{end}])$
if token is a bond **then**
  $allowed_{next} = [atoms]$
  for $i \leftarrow 1$ to $ring_{max}$ **do**
    if $ring^i$ has not been closed **then**
      $allowed_{next}.append([ring^i_{end}])$
if token is a branch start **then**
  $allowed_{next} = [bonds]$
if token is a branch end **then**
  $allowed_{next} = [branch_{start}]$
  if all branches are closed **then**
    $allowed_{next}.append([EOS])$
  else
    $allowed_{next}.append([branch_{end}])$
if token is a ring start **then**
  $allowed_{next} = [bonds, branch_{start}, ring_{start}]$
  if all branches are closed **then**
    $allowed_{next}.append([EOS])$
  else
    $allowed_{next}.append([branch_{end}])$
if token is a ring end **then**
  $allowed_{next} = []$
  if all branches are closed **then**
    $allowed_{next}.append([EOS])$
  else
    $allowed_{next}.append([branch_{end}])$
if Beginning Of Sentence (BOS) **then**
  $allowed_{next} = [atoms]$
if End Of Sentence (EOS) **then**
  $allowed_{next} = []$
Apply the valency mask (remove illegal tokens from $allowed_{next}$ based on the valency of the atoms)
Mask every token not included in $allowed_{next}$ before sampling the next token

Algorithm 4 **STGG+** Masking algorithm

Require: maximum number of rings $ring_{max}$
Require: maximum length of a sequence MAX_{LEN}
if token is an atom **then**
  $allowed_{next} = [bonds, branch_{start}]$
  if this is the second double-bond in a row within a branch **then**
    $allowed_{next}.remove([bond_{double}])$
  if we have made less than $ring_{max}$ rings **then**
    $allowed_{next}.append([ring_{start}])$
  if all branches are closed **then**
    $allowed_{next}.append([EOS])$
    if empty bonds (.) are allowed (for compounds such as $[Na+].[Cl-]$) **then**
      $allowed_{next}.append([bond_{empty}])$
  else
    $allowed_{next}.append([branch_{end}])$
if token is a bond **then**
  $allowed_{next} = [atoms]$
  for $i \leftarrow 1$ to $ring_{max}$ **do**
    if $ring^i$ has not been closed **then**
      $allowed_{next}.append([ring^i_{end}])$
if token is a branch start **then**
  $allowed_{next} = [bonds]$
  if this is the second double-bond in a row within a branch **then**
    $allowed_{next}.remove([bond_{double}])$
if token is a branch end **then**
  $allowed_{next} = [branch_{start}]$
  if all branches are closed **then**
    $allowed_{next}.append([EOS])$
    if empty bonds (.) are allowed **then**
      $allowed_{next}.append([bond_{empty}])$
  else
    $allowed_{next}.append([branch_{end}])$
if token is a ring start **then**
  $allowed_{next} = [bonds, branch_{start}]$
  if this is the second double-bond in a row within a branch **then**
    $allowed_{next}.remove([bond_{double}])$
  if we have made less than $ring_{max}$ rings **then**
    $allowed_{next}.append([ring_{start}])$
  if all branches are closed **then**
    $allowed_{next}.append([EOS])$
    if empty bonds (.) are allowed **then**
      $allowed_{next}.append([bond_{empty}])$
  else
    $allowed_{next}.append([branch_{end}])$
if token is a ring end **then**
  $allowed_{next} = []$
  if all branches are closed **then**
    $allowed_{next}.append([EOS])$
    if empty bonds (.) are allowed **then**
      $allowed_{next}.append([bond_{empty}])$
  else
    $allowed_{next}.append([branch_{end}])$
if Beginning Of Sentence (BOS) or Empty bond (.) **then**
  $allowed_{next} = [atoms]$
if End Of Sentence (EOS) **then**
  $allowed_{next} = []$
When getting too close to MAX_{LEN}, if possible, remove atom, bonds, ring-start, branch-start tokens from $allowed_{next}$
Apply the valency mask (remove illegal tokens from $allowed_{next}$ based on the valency of the atoms)
Mask every token not included in $allowed_{next}$ before sampling the next token