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 (Weininger, 1988; Song & Ermon, 2019) for unconditional generation. In the real world, we want to be able to generate molecules conditional on one or multiple desired properties rather than unconditionally. Thus, in this work, 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. Traditional methods have mainly focused on SMILES (Weininger, 1988) 1D strings (Segler et al., 2018; Kwon et al., 2023), and 2D graphs (Jo et al., 2022; Vignac et al., 2022; Jo et al., 2023). A significant issue with these representations is that a single error by the generative model can result in invalid molecules, especially as molecule size increases.

Recently, Krenn et al. (2020) proposed Self-referencing Embedded Strings (SELFIES) (Krenn et al., 2020), a robust 1D string representation similar to SMILES that guarantees the generation of valid molecules through a carefully designed context-free grammar. However, recent 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, making the problem much more challenging for the model to solve.

An alternative approach, Spanning Tree-based Graph Generation (STGG) (Ahn et al., 2021), has recently emerged. Unlike SELFIES, STGG is designed explicitly for generative models and works 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. STGG uses a simple set of if/else conditions to mask out invalid tokens, which not only prevents invalid molecules but also leads to higher-quality and more diverse generated molecules (Ahn et al., 2021). Jang et al. (2023) shows that STGG generally performs equally or better than other state-of-the-art generative models (Song & Ermon, 2019; Ho et al., 2020; Song et al., 2020) for unconditional molecule generation. 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, we want to *generate molecules conditional on one or multiple desired properties* rather than unconditionally. Furthermore, we want to condition

on *any* subset of desirable properties without retraining the model each time that we condition on a different subset of properties.

Self-criticism Another critical issue is the synthesis time for molecules, which can take weeks, or months. Thus, we cannot expect chemists to synthesize all generated molecules. Ideally, we need a way to filter the molecules that we provide to chemists. Some properties can be verified through simulations, but this can be extremely 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 troublesome. *What if the generative model could predict the properties of its own generated molecules?* This is the idea we propose here: we give the model the ability to predict properties and thus *self-criticize* its own generated molecules, allowing it to automatically filter out those with undesirable properties.

Out-of-distribution properties We sometimes seek to generate novel molecules with out-of-distribution (OOD) properties in order to expand the range of our molecular knowledge. These OOD properties generally involve extreme range of values. 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 out-of-distribution conditioning values, especially for extreme values, resulting in poor generative efficiency (% of valid, unique, and novel molecules) and conditioning fidelity. Since guidance can be beneficial to some conditioning values, but not others, we propose a solution: *random guidance* with best-of-k self-filtering (described further below).

In this work, we tackle any-property-conditional molecule generation with self-criticism using an improved STGG. In doing so, we make the following contributions:

1. **Any-property-conditioning**: We use an MLP on standardized continuous features and embeddings on categorical features while randomly masking some properties during training, allowing conditional generation on any number of properties (0, 1, 2, or all) and the use of Classifier-Free Guidance (CFG) for improved performance (Section 3.2).
2. **Improved Transformer architecture**: We improve on the original Transformer architecture used in STGG by using: Flash-Attention, no bias terms, RMSProp, rotary embeddings, the SwiGLU activation, and better hyperparameters (Section 3.1).
3. **Improved Spanning-Tree**: We extend STGG to 1) allow compound structures with a new token and masking conditions, 2) prevent incomplete samples through special masking when there are too many opened branches, 3) prevent ring overflow, 4) randomize the order of the graph during training for better generalization, and 5) automatically calculate valency and adapt the token vocabulary based on the dataset (Section 3.3).
4. **Auxiliary property prediction objective**: The objective improves conditioning fidelity and enables out-of-the-box self-filtering of molecules with incorrect properties. (Section 3.5)
5. **Random guidance for extreme value conditioning**: Classifier-free guidance uses guidance $w > 1$ to improve performance (Section 3.4), but this can fail when conditioning on extreme values (which are needed to generate molecules with out-of-distribution properties). We propose using random guidance with best-of-k filtering as a solution (Section 3.6).
6. **Comprehensive performance evaluation**: We demonstrate excellent performance in terms of 1) distribution learning and diversity on unconditional generation (Section 4.1), 2) distribution learning and conditioning fidelity on in-distribution (Section 4.2) and out-of-distribution (Sections 4.3 and 4.5) conditional generation, and 3) diverse and high-reward samples on reward maximization (Section 4.4).

2 BACKGROUND

2.1 1D VS 2D REPRESENTATIONS

There are many ways of representing molecules in the context of molecular generation. Some of the most popular methods are autoregressive models on 1D strings (Segler et al., 2018; Ahn et al., 2021; Kwon et al., 2023) and 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

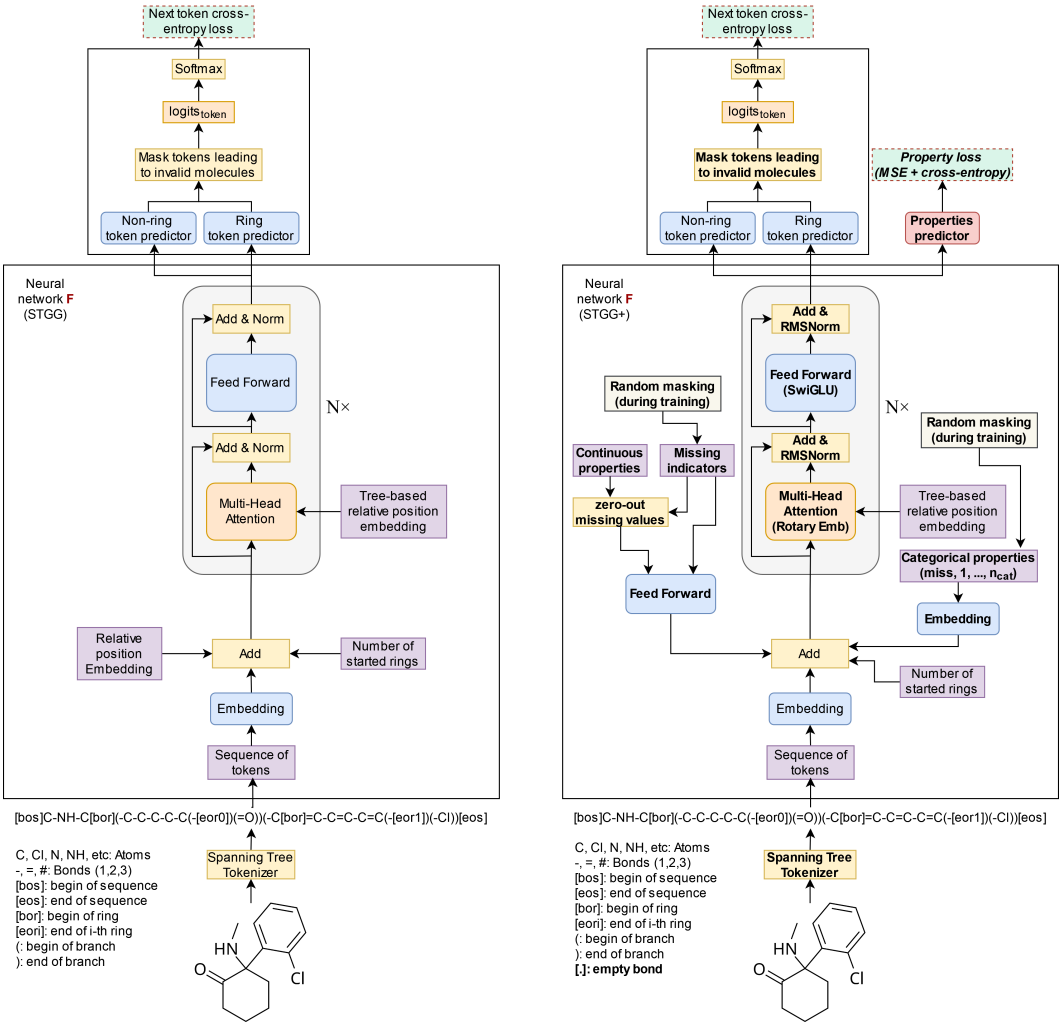

Figure 1: Left: STGG architecture, Right: Our **STGG+** architecture. The molecule is tokenized and embedded. The number of started rings and embeddings of continuous and categorical properties are added, and the output is passed to the Transformer. The Transformer output is then split to produce 1) the predicted property and 2) the token predictions (masked to prevent invalid tokens). Areas that changed or were added in **STGG+** are in **bold**. Please read the method section for more details.

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 of Transformer models, and thus, we focus on this type of representation in our work.

2.2 1D STRING REPRESENTATIONS

The most popular choice of 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. To

address this problem, recent methods like Spanning Tree-based Graph Generation (STGG) (Ahn et al., 2021) and SELFIES (Krenn et al., 2022) have been developed to prevent the generation of invalid molecules. For a detailed comparison of SMILES, SELFIES, and STGG, see Appendix A.2.

STGG has demonstrated performance on par with or better than state-of-the-art unconditional generative models (Ahn et al., 2021; Jang et al., 2023). Conversely, SELFIES has been shown to perform worse than SMILES on property-conditional molecule generation (Gao et al., 2022; Ghugare et al., 2023). Therefore, in this work, we focus on STGG for property-conditional molecule generation.

2.3 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 unconditional molecule generation. STGG leverages a Transformer (Vaswani et al., 2017) architecture 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., atom after atom, bond after bond, or ring-i end when fewer than i ring start tokens are present).

In the next section, we will show how to improve the STGG architecture, vocabulary, and masking, adapt STGG for any-property conditional generation, and improve fidelity on conditioned properties through several techniques (classifier-free guidance, self-criticism, random classifier-free guidance for extreme conditioning).

3 METHOD

We tackle the problem of any-property conditional generation with self-criticism using STGG.

3.1 ARCHITECTURE

We enhance the architecture used in STGG, 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, see Appendix A.5.

3.2 ANY-PROPERTY CONDITIONING

We preprocess continuous properties by applying a simple z-score standardization.

To condition the model on any subset of target properties without retraining the model every time we change the properties, we need to be able to turn off the conditioning of some properties. For continuous variables, we handle 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 will not make sense). 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 number of properties. See Appendix A.3.2 for details. This allows us to condition the model on any subset of desired properties at test time while ignoring the rest.

In the neural network, we process the standardized continuous features (continuous properties concatenated with their binary missing indicators) in a 2-layer multilayer perceptron (MLP) with 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. We also experimented with injecting these embeddings through adaptive normalization (Huang & Belongie, 2017), a method commonly used for conditioning on noise-level in diffusion models (Ho et al., 2020), but this approach massively increased the number of parameters without improving performance.

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 a 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 incomplete samples. Additionally, for massive molecules, it is possible for the model to rarely 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).

Contrary to STGG, we do not canonicalize molecules and instead use a random ordering of the molecules (a different random ordering is sampled for each molecule during training). Doing so improves generalization (see Appendix A.8).

3.4 Classifier-free guidance

To enforce better conditioning of the properties, we use classifier-free guidance, originally designed for diffusion models (Ho & Salimans, 2022), and found beneficial for autoregressive language models as well (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 illustrates this concept. The amount of guidance typically requires hyperparameter-tuning. However, for simplicity and generality, in all analyses, we arbitrarily set the guidance parameter w to 1.5, where $w = 1$ means no guidance.

3.5 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. During training, the model is tasked with predicting both the next token in

the sequence and the properties of the current unfinished molecule. During sampling, we generate molecules conditioned on the desired properties with classifier-free guidance. Then, we mask out the properties (making them 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.

The architecture with the described loss functions is illustrated in Figure 1. The methodology for generating molecules using classifier-free guidance and self-criticism is depicted in Figure 2.

3.6 Random guidance for extreme conditioning

Regular guidance can be problematic for some 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.

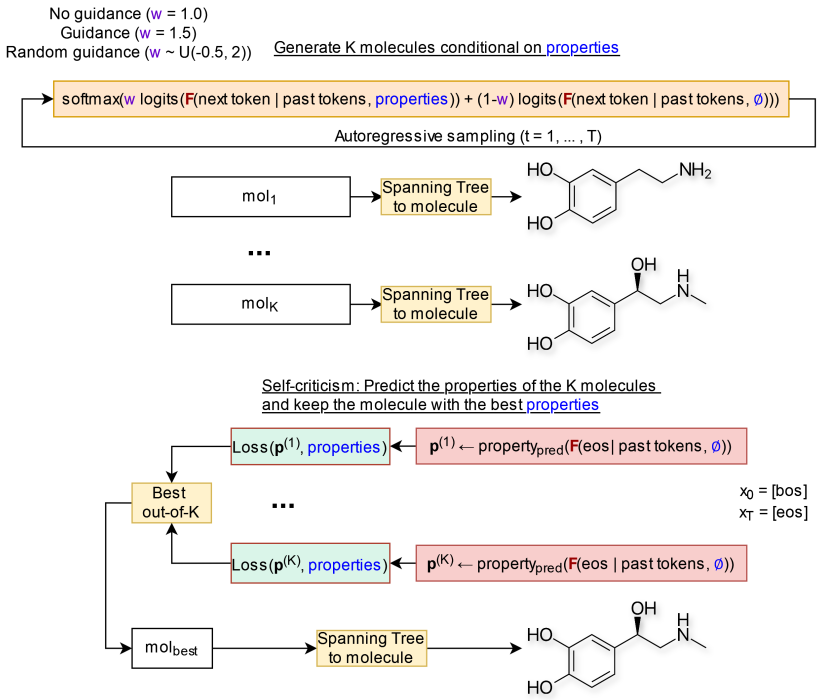

Figure 2: Generation and self-prediction using **STGG+**. We autoregressively generate K molecules conditional on desired properties using classifier-free guidance. The unconditional model predicts the properties of the K molecules and the molecule assumed closest to the desired properties is returned.

4 Experiments

We run four sets of experiments. First, we demonstrate that, after conditioning on in-distribution properties, our model can recover molecules close to those in the test set, achieving performance on par with state-of-the-art unconditional models. Second, we show that our model can generate molecules conditioned on properties from the test set with high fidelity on the specified properties. Third, we illustrate that our model can generate highly efficient (high % of novel, unique, and valid) molecules with high fidelity on out-of-distribution (OoD) properties. Fourth, 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, as a harder case, we show that our model can generate high fidelity molecules conditioned on out-of-distribution (OoD) properties on a small dataset of large molecules.

See Appendix A.3 for details on the datasets used, Appendix A.4 for more information on the hyperparameters, Appendix A.6 for property prediction performance metrics of the self-critic, and Appendix A.8 for an ablation on OOD properties for Zinc. Note that we rely on the following software: PyTorch (Paszke et al., 2019), Molecular Sets (MOSES) (Polykovskiy et al., 2020) and RDKit: Open-source cheminformatics (Landrum et al., 2024). Unless otherwise specified, we use RDKit to evaluate the properties of the generated molecules.

4.1 Unconditional generation

We train our model on QM9 (Ramakrishnan et al., 2014) and Zinc250K (Sterling & Irwin, 2015) using the molecule weight, logP, and Quantitative Estimate of Druglikeness (QED) (Bickerton et al., 2012) as properties. We test the similarity in the distribution of unconditionally generated molecules (masking the properties). We also test the same metrics on conditionally generated molecules, conditioned on properties from the test set. We use the same train, valid, and test splits as Jo et al. (2022). We compare to the strong recent baselines reported in Jang et al. (2023) which are: EDP-GNN (Niu et al., 2020), GraphAF (Shi et al., 2020), GraphDF (Luo et al., 2021), GDSS (Jo et al., 2022), DiGress (Vignac et al., 2022), DruM (Jo et al., 2023), GraphARM (Kong et al., 2023), GEEL (Jang et al., 2023), CharRNN (Segler et al., 2018), CG-VAE (Liu et al., 2018), MoFlow (Zang & Wang, 2020), and STGG (Ahn et al., 2021). The metrics are % Valid, % unique, % novel, Fréchet ChemNet Distance (FCD) (Preuer et al., 2018), scaffold similarity (Scaf.), similarity to nearest neighbor (SNN), and fragment similarity (Frag.).

Results The top performing methods (STGG, GEEL, STGG+) are shown in Table 1; they have similar performance. The full experiments can be found in Appendix A.7.

Table 1: Unconditional molecular graph generation performance.

Method	Valid (%) (↑)	Unique (%) (↑)	Novel (%) (↑)	FCD (↓)	Scaf. (↑)	SNN (↑)	Frag. (↑)
			QM9				
GEEL	**100.0**	96.08	22.30	**0.089**	0.9386	0.5161	0.9891
STGG	**100.0**	96.76	72.73	0.585	**0.9416**	**0.9998**	**0.9984**
STGG+	**100.0**	**97.17**	**74.41**	**0.089**	0.9265	0.5179	0.9877
			Zinc250K				
GEEL	99.31	99.97	99.89	0.401	0.5565	0.4473	0.9920
STGG	**100.0**	**99.99**	99.89	**0.278**	**0.7192**	**0.4664**	**0.9932**
STGG+	**100.0**	**99.99**	**99.94**	0.395	0.5657	0.4316	0.9925

4.2 Conditional generation

We follow the same protocol as Liu et al. (2024). We train our model on HIV, BACE, and BBBP (Wu et al., 2018). We use the same train, valid, and test splits as Liu et al. (2024). Each dataset has a 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 in the generated samples), 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.

Notably, the Fréchet Distance is one of the most popular and meaningful distance in generative models as it correlates well with both quality and diversity (Heusel et al., 2017); it corresponds to the Wasserstein distance on the hidden space of neural networks assuming normality. It has been used in many domains: images (Heusel et al., 2017), audio (Kilgour et al., 2018), videos (Unterthiner et al., 2019), and molecules (Preuer et al., 2018).

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), and MOOD (Lee et al., 2023), and Graph DiT (Liu et al., 2024).

Results The experiments are shown in Table 2 (for the full table with more baselines, see Appendix A.9). 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.

Table 2: 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.

4.3 Out-of-distribution conditional generation

We follow the same protocol as Kwon et al. (2023). Our model is trained on Zinc250K (Sterling & Irwin, 2015) 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: 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 must follow Kwon et al. (2023) protocol and its useful to test how the model behave in erroneous scenarios. 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).

Results The experiments are shown in Table 3. We see 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 high logP, suggesting that the property predictor of STGG+ makes incorrect predictions for out-of-distribution high logP values.

Table 3: 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 (MinMAE).

	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

*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.

4.4 Reward maximization

Jain et al. (2023) train reinforcement learning (RL) and GFlowNet (Bengio et al., 2023) agents to solve a task based on the QM9 (Ramakrishnan et al., 2014) dataset. They seek to produce QM9-like molecules which 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 giving the reward to our model, we train a STGG+ model conditioned on the four properties. 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.

Results Our experiments are shown in Table 4. Our approach yields slightly better molecules in terms of reward and diversity compared to online methods, using around 11.5% of the molecules. This makes our approach significantly more efficient. However, it is important to note that solving this task with online methods is a steep hill and can be considered more difficult.

4.5 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

Table 4: Reward maximization on QM9.

	Type	Data	Reward (↑)	Diversity (↑)
Envelope QL	Online	1M molecules	0.65	0.85
MOGFN-PC			0.76	0.93
STGG**	Offline	QM9 (∼115K molecules)	0.73	0.10
STGG+ ($k = 1$)			0.78	0.76
STGG+ ($k = 100$)			0.77	0.98

**STGG with missing indicators, and random masking.

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 might 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 training with this strategy (pre-train and fine-tune) in addition to only training on Chromophore DB.

Results The experiments are shown in Table 5. We find that pre-training on Zinc250K generally improves performance (Generative Efficiency and MinMAE) over training only on Chromophore DB. 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.

Table 5: 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 (MinMAE). We removed the low molWt and QED which are both impossible negative values.

	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

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

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 validty, 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.

While our method generates molecules with relatively good accuracy concerning the desired properties, it is still not perfect and can produce incorrect molecules, especially in out-of-distribution scenarios, which is a challenging task. Additionally, the property predictor of our approach may not be as optimal as property predictors engineered explicitly for this task, meaning our method may not always select the best molecules out of k choices, particularly in out-of-distribution scenarios; we found this to be the case for large out-of-distribution logP conditioning values. Currently, the method does not account for stereoisomers, although some properties can be dependent on stereoisomers.