Exploring graph traversal algorithms in graph-based molecular generation

Rocío Mercado,*,† Esben J. Bjerrum,† and Ola Engkvist*,†,‡

†*Molecular AI, Discovery Sciences, R&D, AstraZeneca Gothenburg*

Pepparedsleden 1, 431 50 Mölndal, Sweden

‡*Department of Computer Science and Engineering, Chalmers University of Technology*

Chalmersplatsen 4, 412 96 Gothenburg, Sweden

E-mail: rociomer@mit.edu; ola.engkvist@astrazeneca.com

Abstract

Here we explore the impact of different graph traversal algorithms on molecular graph generation. We do this by training a graph-based deep molecular generative model to build structures using a node order determined via either a breadth- or depth-first search algorithm. What we observe is that using a breadth-first traversal leads to better coverage of training data features compared to a depth-first traversal. We have quantified these differences using a variety of metrics on a dataset of natural products. These metrics include: percent validity, molecular coverage, and molecular shape. We also observe that using either a breadth- or depth-first traversal it is possible to over-train the generative models, at which point the results with the graph traversal algorithm are identical.

Introduction

Deep molecular generative models have gained a lot of attention recently for their promise to efficiently traverse the chemical space in search of molecules with desired properties.[1] These models harness tools from machine learning, such as deep neural networks and reinforcement learning, for the generation of new molecules which satisfy a set of desired criteria (e.g. activity against a known protein target, novelty, low toxicity, etc). As molecular design is a highly complex multi-objective optimization problem, where many inter-related properties need to be simultaneously optimized, the hypothesis is that deep learning methods are better equipped for identifying optimal molecules out of the vast chemical space than traditional molecular discovery paradigms, both in terms of speed and the quality of the solutions proposed. A wide range of deep molecular generative models have thus been proposed since 2016, including string-based[2–5] and graph-based models,[6–10] conformer generation methods,[11–14] and molecular generation approaches which take into account molecular synthesizability.[15–17]

Nonetheless, there have been limited studies on whether the order in which atoms are presented to a model during training/generation plays a role in the performance of these deep molecular generative models.[18,19] Here, we set out to answer whether a breadth- or a depth-first search would lead to better performance in graph-based deep molecular generation, as breadth- and depth-first search are two common graph traversal algorithms which are used to determine the artificial node order in molecular representations. We measure better performance via the generation of more desirable molecules, where 'desirability' is determined by comparing 1) the percentage of valid/unique molecules sampled, 2) chemical space coverage (including ring systems), and 3) the shape distributions of the sampled molecules. Molecular shape here is estimated from principle moment of inertia ratios.

Related Work

Graph traversal algorithms

Graph traversal algorithms (GTAs) determine the order in which nodes and edges are visited in a graph.[20] In cheminformatics, the most popular GTAs are breadth-first search (BFS), depth-first search (DFS), random search, and variations thereof.[21] Here, we focus on BFS and DFS, two algorithms which can be used to traverse a graph in opposing ways.

A BFS proceeds as follows: first all nearest neighbors of the pre-determined 'starting node' are sampled, followed by the nearest neighbors of those, and so on, until all nodes have been visited. Here, the term 'nearest neighbors' refers to the nodes directly connected to a given node via an edge. On the other hand, in a DFS, first a single branch stemming from the starting node is traversed entirely, followed by the next branch, and so on, until all branches have been explored. Whenever a new branching point is reached within each branch, the algorithm is repeated for each sub-branch, treating the branching node as the new 'starting node.' We illustrate the first few graph traversal steps using each of these algorithms in Figure 1 for Gilvsin D, a natural product randomly sampled from the dataset used in this work.

In both of these algorithms, ties between equivalent nodes can be broken by assigning a value to each node before/during the graph traversal and choosing, e.g., the node with the highest value when two otherwise equivalent nodes are reached. In this work, we use the canonical node order assigned by RDKit[22] for tie-breaking, to avoid re-implementing a separate tie-breaking algorithm. Additionally, we use slightly modified BFS and DFS algorithms when creating the training data; these modifications ensure that no disconnected fragments are generated during data preprocessing.[23] This was important due to the MPNN-component in our generative model, where having no disconnected fragments ensures that all nodes can pass 'messages' during the training and generation processes.

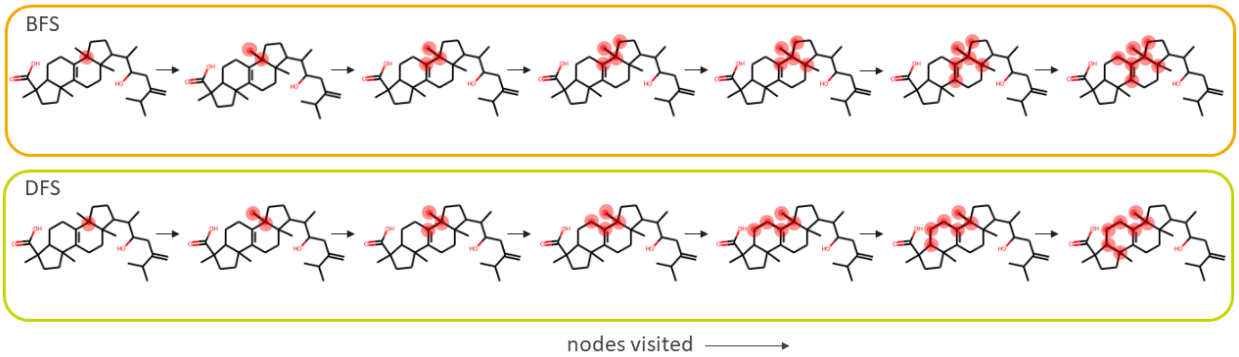

Figure 1: Illustration of the BFS and DFS node traversal for the first 7 nodes visited in Gilvsin D, a natural product found in the COCONUT dataset.[24] The red highlighted atoms indicate the atoms visited thus far in that step. During sampling, molecules will be generated via the sequential addition of nodes in the order in which they were traversed during training.

Graph-based molecular generation with GraphINVENT

In this work, we use GraphINVENT[23,25] as the baseline graph generative model for the GTA comparison. GraphINVENT is an auto-regressive, graph-based, molecular generative model which generates molecules one atom/bond at a time. GraphINVENT uses graph neural networks (GNNs) and a tiered deep neural network architecture to generate new molecular graphs sequentially by learning their action probability distributions (e.g. valid chemical actions) without any hard-coded chemical rules or masks. Models are trained by minimizing the Kullback-Leibler divergence between the 'true' and predicted action probability distributions for constructing a batch of graphs, where the action probability distributions are multi-dimensional tensors which encode all possible actions for growing (or terminating) a graph. GraphINVENT models can construct molecular graphs starting from either empty or incomplete graphs.

Contributions

In this work, we have analyzed the effect of using a breadth-first versus a depth-first node order when generating molecular graphs using GraphINVENT, an auto-regressive, GNN-based, deep molecular generative model. We find that using a breadth-first search leads

to better validity and chemical space coverage in molecules sampled from suitably trained models, and that models using breadth-first traversal are better at learning to generate 'rounder' molecules. Finally, we find that models using either GTA are able to over-train on the data, at which point their performance is nearly identical.

Methods

Molecular graph generation with GraphINVENT

Here we summarize the general structure of GraphINVENT models, and the algorithm for molecular graph generation, which is discussed in detail in a previous publication.[23]

Model structure

The generative models in GraphINVENT are deep neural networks, each consisting of two blocks: (1) a *GNN* block, and (2) a *global readout* block.

The *GNN block* takes as input the graph representation of a molecule (adjacency tensor, E, and node features matrix, X) and outputs the transformed node feature vectors, H^L, and the graph embedding, g. The *global readout block* then predicts an *action probability distribution* (APD) for each graph from the learned embeddings H^L and g. The APD is a vector property containing probabilities for all possible actions for growing a given graph; sampling it tells the model how to grow a graph.

Sampling

GraphINVENT models receive graphs as input and output APDs, from which possible actions can be sampled and applied to the graphs. The three possible actions are (1) *adding* a new node to the graph, (2) *connecting* the last appended node to another node in the graph, and (3) *terminating* the graph construction.

During sampling, new molecular graphs are created by inputing an empty graph into the model, and letting the model predict what action to sample next by outputing an APD (in the case of an empty graph, the only viable action is *adding* a new node). After an action is sampled from the resulting APD, it is applied to the input graph, and the resulting graph is fed back into the model to predict a new APD. This process of (1) generating an APD, (2) sampling and action, and (3) applying the action to the input graph, is repeated until the "terminate" action is sampled, or until an invalid action is sampled.

Datasets

The primary datasets used in this work were subsets from the COCONUT natural product database.[24] We explored natural products as they generally contain more complex ring systems and larger structures than typical drug-like datasets, which we believed would emphasize any differences between the two GTAs explored in this work. To narrow down the sets of molecules for this study, we filtered the full COCONUT by removing compounds with > 60 heavy atoms and the following atom types: {K, Na, Fe, As}. From the remaining structures, we then created 4 training sets, each with the same number of structures in the test set (10K or 5K) and 1K structures in the validation set:

- 10K randomly sampled molecules, referred to as the 'mixed' set herein,
- 10K randomly sampled 'long' molecules,
- 10K randomly sampled 'circular' molecules,
- 5K randomly sampled 'spherical' molecules.

We separated the molecules from the filtered COCONUT dataset into three separate classes ('long', 'circular', and 'spherical') based on their shape. To do this, a single conformer was generated for each molecule using RDKit's 'AllChem.EmbedMolecule()' method,[26] and the resulting conformers were classified according to their normalized principle moment ratios (NPRs) as follows:

- a conformer is classified as 'long' if it has NPR1 < 0.5 and NPR2 > 0.75,
- a conformer is classified as 'spherical' if it has NPR1 > 0.5 and NPR2 > 0.75,
- a conformer is classified as 'circular' if it has NPR2 < 0.75.

Here, we define NPR1 $= I_1/I_3$ and NPR2 $= I_2/I_3$, where I_1 is the first (smallest) principal moment of inertia (PMI), I_2 is the second PMI, and I_3 is the third (largest) PMI.[27] Most molecules in COCONUT are classified as 'long' under the above scheme, so we randomly sampled from within each class to get a similar number of molecules in each subset. However, not many molecules satisfy the 'spherical' criteria in the COCONUT subset (only ∼11K after applying our filters, which was fewer than for the other two shape classes) and that is why this subset is a bit smaller. The NPR distributions for each of the above subsets are illustrated using PMI plots in SI Figure 6.

In the SI we discuss two additional datasets, including a drug-like dataset.

Training GraphINVENT models

To train the generative models, we used the GGNN-based GraphINVENT implementation using default hyperparameters, with exceptions only for the following:

- *atom types*: B, C, N, O, F, Si, P, S, Cl, Se, Br, and I
- *formal charges*: -1, 0, 1, 2, and 3
- *max nodes*: 60
- *decoding route*: BFS or DFS.

All models were trained for 100 epochs, with sampling every 5 epochs.

Evaluation metrics

To evaluate the models, we considered the following metrics:

- Percent validity and uniqueness of sampled structures[a]

[a]Validity evaluated for 1M samples, while uniqueness and coverage evaluated only for the valid samples.

- Training and validation losses
- Chemical space coverage:[28,a]
  - Fraction molecules reproduced from reference set
  - Fraction ring systems reproduced from reference set
  - Fraction functional groups reproduced from reference set
- Shape analysis:[27,29,b]
  - NPR1 and NPR2 distributions (aka 'PMI plots')

Above, the term *reference set* refers to the combined training, testing, and validation sets for each COCONUT subset.

Sub-structure identification For the identification of ring systems and functional groups, we used an algorithm implemented in RDKit which is based on iterative marching through each molecule's atoms.[30]

Conformer generation For the shape analysis, we generated one low-energy conformer for each sampled molecular graph using RDKit's 'AllChem.EmbedMolecule()' method.[26] The method uses distance geometry, which requires a small amount of empirical information (e.g. ideal bond lengths, ideal bond angles, and ideal torsional angles) to generate conformers without the need for energy minimization using force fields. Experimental torsion angles are stored in a list and matched to specific sub-structures using SMARTS patterns during creation of an initial distance bounds matrix, from which a random distance matrix can be sampled and further refined. With this conformer generation algorithm, termed 'Experimental-Torsion Knowledge Distance Geometry' (ETKDG), the authors showed that 84% of a set of 1290 small-molecule crystal structures from the Cambridge Structural Database could be reproduced within an RMSD of 1.0 Å in the original work.[26] Then, the NPRs were used to

[b]Shape evaluated for 10K random, valid samples out of the 1M generated set.

quantify the shape of each generated conformer as in previous work.[27,29] New conformers were generated whenever we needed to compute a molecule's NPRs (we did not store the 3D conformers generated), and as there is an element of randomness in ETKDG, the conformers may be slightly different between runs. Nonetheless, the slight differences between conformers do not impact the results, as 10K structures are analyzed per PMI plot, and we verified that indeed PMI plots generated by different runs are statistically identical.

Evaluation The models were each evaluated at two epochs:

- E_{opt}: the epoch which minimizes the validation loss,
- E_{max}: the epoch which minimizes the training loss and maximizes the validation loss (i.e. the final training epoch).

For evaluation, 1M molecules were sampled from each trained model at the aforementioned epochs. No validity filters were used during sampling, meaning that if only 60% of the sampled molecules were 'valid', only 600K molecules would be written to disk after sampling.

Computational details

All the above models were trained using GraphINVENT 2.0 on an NVIDIA RTX-2080 Ti card using CUDA 10.1 and driver version 460.67.

Results

Analyzing the generation process

We can evaluate the generation process by looking at how structures are generated during sampling. In Figure 2, we illustrate the intermediates for two randomly sampled structures from the generated set. As expected, we see that using BFS, the generated molecule is built in a more 'circular' fashion and with ring formation occurring early on during generation, while

using the DFS leads to longer chains being built initially in the structure and ring closures coming later. Notably, the illustrated example was not 'cherry picked;' the differences are this striking for almost any randomly selected set of structures.

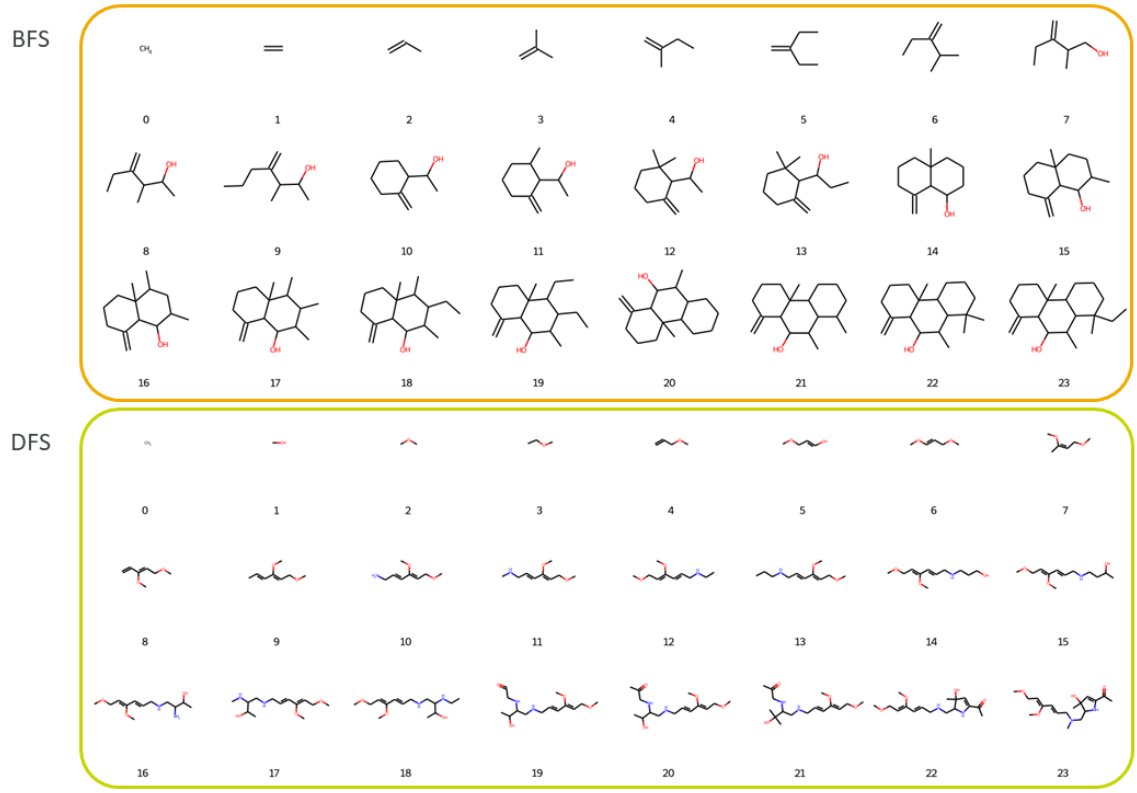

Figure 2: Example of two randomly selected molecules with an equal number of nodes being generated via a BFS (top) and DFS (bottom). These molecules were sampled from models trained on the 'mixed' COCONUT subset at E_{opt}.

File size and compute time

File size Interestingly, the HDF files created for the various training sets during preprocessing were 3-5% smaller for the DFS-processed data when compared to the BFS-processed data. During preprocessing, GraphINVENT combines identical subgraphs within mini-batches before saving processed graphs to disk.[23] This suggests that graphs processed using a DFS algorithm share more subgraphs in common than those processed using a BFS algorithm, as a greater overlap in subgraphs would mean fewer graphs are needed to represent the training data and thus lead to smaller files.

Compute time Sampling 1M molecules from each trained model took c.a. 2 hours per model on a single GPU. Training took a similar amount of time (a few hours per model).

Loss

For all the aforementioned COCONUT subsets, we found that $E_{opt} = 20$ and trained such that $E_{max} = 100$. For all models, we found that at E_{opt}, using a DFS led to both a lower training and validation loss for all datasets. At E_{max}, all models reached the same training loss, irrespective of GTA, while the BFS-trained model reached a higher validation loss. This suggests that using a BFS, over-fitting happens more quickly, likely due to the slightly greater number of subgraphs in the training data (see **File size** subsection above). By over-fitting here, we mean that the difference between the validation and training losses is large, implying the model is "over-fit" to the training set. This is often due to the training loss decreasing while the validation loss is increasing. Indeed, when we analyze the loss curves, we see that the validation loss increases much more rapidly when using a BFS than when using a DFS. Example loss curves for the COCONUT 10K subset ('mixed') are shown in the SI and were similar for all datasets.

Percent validity and uniqueness

Percent validity The first observation we made was that using the BFS algorithm led to a greater fraction of valid molecules for models trained using all except the 'circular' and 'spherical' datasets at E_{opt} (Table 1). The differences were small but significant (generally 5-10% higher validity using BFS). For the 'circular' and 'spherical' datasets, the percent validities were within one standard deviation for both algorithms at E_{opt}.

At the overfitting epoch E_{max}, the models performed equally in percent validity regardless of GTA.

Table 1: Comparison of the percent validity of sampled structures for models trained on each COCONUT subset and evaluated at both E_{opt} and E_{max}. Error bars are the standard deviation from 3 different runs.

	E_{opt}		E_{max}	
	BFS	DFS	BFS	DFS
'mixed'	62.67 ± 2.49	49.33 ± 2.62	96.67 ± 0.94	96.67 ± 0.94
'long'	53.33 ± 3.68	47.33 ± 3.30	96.00 ± 0.00	95.67 ± 0.47
'circular'	54.33 ± 4.78	50.67 ± 4.78	96.00 ± 0.00	96.00 ± 0.00
'spherical'	49.33 ± 3.30	52.67 ± 6.85	94.00 ± 0.00	95.00 ± 0.00

Percent uniqueness In general, the percent uniqueness was quite high across all models (>95%), and models performed equally on this metric for all datasets at the respective epochs analyzed. There was no observable difference between using BFS or DFS in the percent uniqueness. The only exception was the 'spherical' dataset, for which the BFS algorithm led to a lot more unique structures at E_{opt}. Results are shown in SI Figure 4.

Percent uniqueness is high across the board for these models because the action space is very large. Interestingly, all models showed a greater percent uniqueness at E_{max} compared to E_{opt}. The reason for this is that the percent uniqueness was evaluated only for the *valid* generated structures (out of 1M), and models generate a lot more valid structures at E_{max} (see Table 1).

No mode collapse was observed for any of these models.

Chemical space coverage analysis

Here we analyze the results for the chemical space coverage, breaking down results into 1) overall molecular coverage, 2) ring system coverage, and 3) functional group coverage.

Molecular coverage For all datasets, the molecular coverage, or the number of molecules reproduced from the reference set, was higher at E_{opt} for models using a BFS compared to models using a DFS. However, the opposite trend was observed at E_{max}, due to the BFS-

based models overfitting much more quickly to the training set and thus being less able to generate molecules from the testing and validation sets which also make up the reference set. In many cases, the differences between BFS and DFS models are often within a standard deviation of each other (Table 2).

Table 2: Comparison of the molecular coverage for models trained on each COCONUT subset and evaluated at both E_{opt} and E_{max}. The molecular coverage is the number of molecules (out of 1M samples) which are reproduced from the reference set (larger values mean better coverage). Each reference set consists of 21K molecules, except the 'spherical' reference set which consists of 11K molecules. Error bars are the standard deviation from 3 different runs.

	E_{opt}		E_{max}	
	BFS	DFS	BFS	DFS
'mixed'	71,547 ± 7059	51,192 ± 1594	879,640 ± 25,490	894,985 ± 25,084
'long'	58,523 ± 9672	44,741 ± 13,107	862,415 ± 1069	882,897 ± 5110
'circular'	53,867 ± 16,901	43,022 ± 3412	864,376 ± 1669	882,996 ± 1908
'spherical'	73,220 ± 21,632	67,269 ± 26,412	826,315 ± 3111	869,103 ± 888

Ring system coverage The number of ring systems found in sampled structures was generally slightly larger for models using a BFS (at both E_{opt} and E_{max}), although in many cases these results were also within one standard deviation of each other (SI Figure 5, top). However, when normalizing by the number of valid structures generated by each model, we find that there are no major differences between the number of ring systems using either algorithm at E_{opt} (at E_{max}, BFS still has a slight advantage).

Nonetheless, evaluating the models only in terms of the number of ring systems reproduced from each respective reference set, we find that at each respective epoch, and for each dataset, models trained on either GTA learn to reproduce the same number of ring systems from the reference set (not illustrated). We emphasize that this observation is true only when comparing BFS and DFS within each respective epoch, as at E_{max} the models indeed reproduce more ring systems from the reference set than at E_{opt}.

Functional group coverage Generally speaking, using a BFS led to greater number of identifiable functional groups in the generated sets at both E_{opt} and E_{max} (SI Figure 5, bottom). However, normalizing by the number of valid structures generated by each model at each respective epoch, we find that the functional group coverage was the same regardless of GTA, just as in the case of ring system coverage. In other words, there are simply more functional groups generated because there are a lot more valid, unique structures generated when using a BFS.

Molecular shape analysis

To visualize the differences in shape between the different sets of generated structures, we generated conformers from the sampled molecular graphs and computed their PMI ratios (NPR1 and NPR2) using RDKit. In Figure 3, we visualize these results using PMI plots, which visualize the NPRs on an upside-down triangle. Molecules with NPRs towards the top-left corner are considered more 'long', molecules with NPRs towards the top-right corner are considered more 'spherical', and molecules with NPRs towards the bottom corner are more 'circular.'[29]

At E_{opt}, there are clear differences in the shapes of the structures generated depending on the GTA used (Figure 3). Generally, what we observe is that structures generated using a BFS tend to be more circular/spherical, whereas structures generated using a DFS tend to be more long. The distributions in NPRs are also much sharper when using a DFS than when using a BFS, meaning that using a BFS leads to molecules which sample the NPR space more broadly. At E_{max}, the NPRs for the different sets of BFS- or DFS-generated structures are indistinguishable (SI Figure 7).

Other datasets

Larger dataset To investigate the effect of training set size, we repeated all calculations done on the 10K dataset on a larger 50K subset of COCONUT. Similar observations were

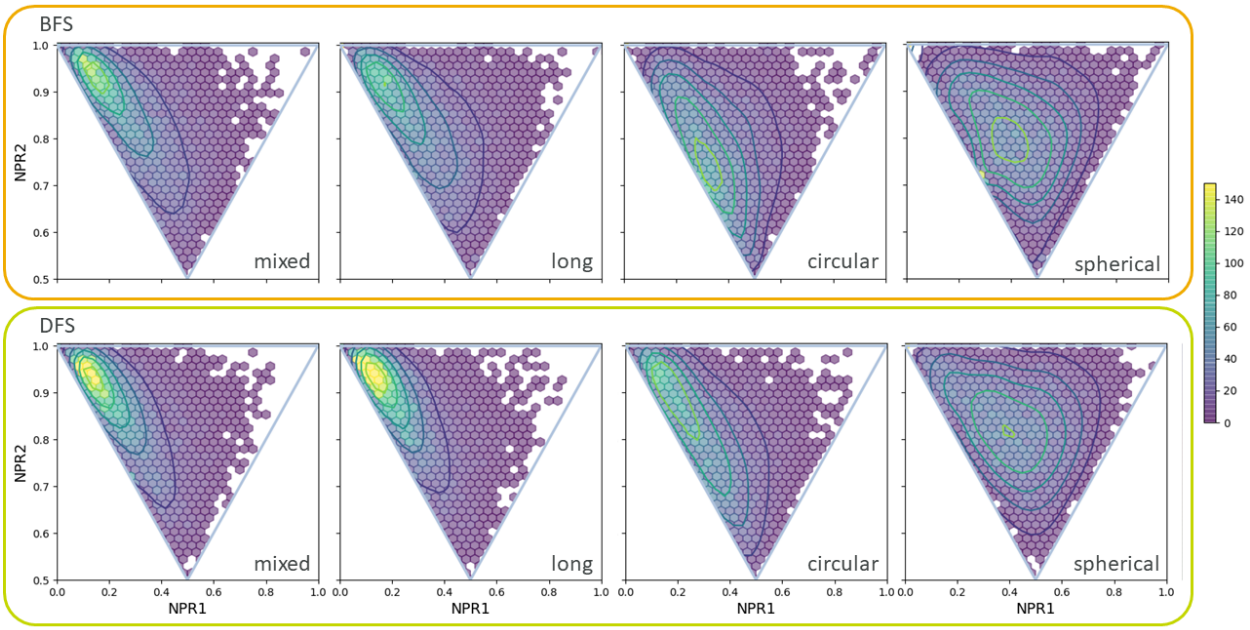

Figure 3: PMI plots for generated molecules for models trained on each COCONUT subset at E_{opt} (top: BFS, bottom: DFS). Although 1M molecules were sampled from each model, results are only plotted for 10K random samples in each case. The contours are plotted over the hexbin histogram, where a hexagon is present if at least one structure was generated in that regime; otherwise the colorbar indicates the number of structures in each bin (out of 10K random samples). Contours are generated from kernel density estimates on the data.

made for the 50K subset as for the 10K subset (see the SI), underlining that the training set size had little to no effect on the GTA comparison.

Drug-like molecules We also repeated the calculations and analysis on a dataset of 4311 dopamine receptor type 2 (DRD2) active molecules. Similar but more subtle observations were made as for the COCONUT subsets discussed above (see the SI), verifying that our results hold for drug-like molecules.

Discussion

The shape a molecule has plays an important role in medicinal chemistry. Although molecules are better described as ensembles of conformations rather than as rigid objects, a molecule's general shape can, for instance, affect whether a molecule is able to access and bind

to a specific protein binding pocket. As such, it can be important to select the appropriate GTA when generating molecules, and we have demonstrated that the GTA used can bias the types of structures generated by a molecular generative model. However, many molecular generation methods use packages (like RDKit) which by default assign a canonical node order that is based on a DFS. As such, we believe that the performance of deep molecular generative models could be improved by simply switching to a canonical node order based on a BFS, rather than a DFS, for molecular generation.

To support this claim, below we summarize our key findings:

- Using a BFS algorithm for molecular graph generation overall leads to higher percent validity in sampled structures than using a DFS (5-10% higher in most cases).
- Using a BFS leads to a larger number of molecules reproduced from the training set in the sampled structures (greater chemical space coverage).
- Using a BFS leads to molecules having more 'circular' or 'spherical' shapes (based on their PMI ratios), on average, whereas using a DFS leads to a bias in 'long' structures.

Additionally, we made the following observations:

- Ring system and functional group coverage does not depend on graph traversal algorithm.
- The findings above are for E_{opt}, or the epoch which minimizes the validation loss. At E_{max}, when models are over-trained, there are no discernible differences between models trained using either algorithm.

It is interesting that ring system and functional group coverage does not depend on the GTA used. We initially expected a greater ring system coverage using BFS, as it leads to ring formation earlier on in the molecular generation process. However, after normalizing for the greater percentage of valid structures generated using a BFS, there were no observable differences, and both algorithms were able to generate the same percentage of ring systems/functional groups from the reference set. In other words, while using a BFS does

lead to a greater number of sampled ring systems and functional groups in the generated structures than when using a DFS, this is a direct result of the greater number of valid and unique structures generated when using a BFS.

We note once again that for the purposes of training with GraphINVENT, we used slightly modified BFS and DFS algorithms which ensure that no disconnected fragments are formed during data preprocessing (due to the message passing neural network in the framework). It would be interesting to see if, in models which can afford to use 'pure' BFS/DFS algorithms, the choice of graph traversal algorithm would lead to even larger differences in the properties of the sampled structures at E_{opt}.

Finally, we emphasize that GraphINVENT uses a single decoding route for each molecular graph seen during training (i.e. BFS/DFS is used once per graph to return a node order). As such, it is unclear whether these observations would still hold for models which use significant amounts of data augmentation via augmenting the number of construction paths (and by extension, node orders) seen by the model for each training example.

Conclusion

In this work we examined what effect the two most common graph traversal algorithms have when used in molecular graph generation. By investigating the differences in sampled molecules from models trained using either a) a breadth-first or b) a depth-first node traversal algorithm, we found that a BFS algorithm demonstrates advantages to DFS for molecular generation when a model is not over-fit. In the limit of large epochs, both BFS- and DFS-based models are able to over-fit to the data and perform identically. We recommend for all future auto-regressive molecular graph generation tools to use a BFS when training, as we found it leads to an increase in the percent validity of the generated structures and better sampling of the reference set properties.

Code availability

The original GraphINVENT code has been updated to include a DFS option for graph traversal in preprocessing. This code is available at `https://www.github.com/MolecularAI/GraphINVENT`. The analysis scripts, dataset, models, and code necessary to reproduce the results presented in this work are available on Zenodo at `https://doi.org/10.5281/zenodo.5018415`.

Acknowledgement

We thank Dr. Samuel Genheden and Simon Johansson for their feedback on the manuscript. R.M. thanks the AZ Postdoc Program for funding.

References

(1) Jiménez-Luna, J.; Grisoni, F.; Weskamp, N.; Schneider, G. Artificial Intelligence in Drug Discovery: Recent Advances and Future Perspectives. *Expert Opin. Drug Discovery* **2021**, 1–11.

(2) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.* **2018**, *4*, 268–276.

(3) Sanchez-Lengeling, B.; Outeiral, C.; Guimaraes, G. L.; Aspuru-Guzik, A. Optimizing Distributions Over Molecular Space. An Objective-Reinforced Generative Adversarial Network for Inverse-Design Chemistry (ORGANIC). *ChemRxiv* **2017**, doi:10.26434/chemrxiv.5309668.v3.

(4) Segler, M. H.; Kogej, T.; Tyrchan, C.; Waller, M. P. Generating Focused Molecule

Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Cent. Sci.* **2018**, *4*, 120–131.

(5) Blaschke, T.; Arús-Pous, J.; Chen, H.; Margreitter, C.; Tyrchan, C.; Engkvist, O.; Papadopoulos, K.; Patronov, A. REINVENT 2.0: An AI Tool for De Novo Drug Design. *J. Chem. Inf. Model.* **2020**,

(6) Samanta, B.; De, A.; Jana, G.; Chattaraj, P. K.; Ganguly, N.; Gomez-Rodriguez, M. NeVAE: A Deep Generative Model for Molecular Graphs. *arXiv* **2018**, arXiv:1802.05283.

(7) Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; Battaglia, P. Learning Deep Generative Models of Graphs. *arXiv* **2018**, arXiv:1803.03324.

(8) You, J.; Liu, B.; Ying, R.; Pande, V.; Leskovec, J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. *arXiv* **2018**, arXiv:1806.02473.

(9) Zang, C.; Wang, F. MoFlow: An Invertible Flow Model for Generating Molecular Graphs. *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.* **2020**,

(10) Jin, W.; Barzilay, R.; Jaakkola, T. Hierarchical Generation of Molecular Graphs using Structural Motifs. *arXiv* **2020**, arXiv:2002.03230.

(11) Simm, G. N. C.; Hernández-Lobato, J. M. A Generative Model for Molecular Distance Geometry. *arXiv* **2020**, arXiv:1909.11459.

(12) Shi, C.; Luo, S.; Xu, M.; Tang, J. Learning Gradient Fields for Molecular Conformation Generation. *arXiv* **2021**, arXiv:2105.03902.

(13) Xu, M.; Luo, S.; Bengio, Y.; Peng, J.; Tang, J. Learning Neural Generative Dynamics for Molecular Conformation Generation. *arXiv* **2021**, arXiv:2102.10240.

(14) Satorras, V. G.; Hoogeboom, E.; Fuchs, F. B.; Posner, I.; Welling, M. E(n) Equivariant Normalizing Flows for Molecule Generation in 3D. *arXiv* **2021**, arXiv:2105.09016.

(15) Bradshaw, J.; Paige, B.; Kusner, M. J.; Segler, M.; Hernández-Lobato, J. M. A Model to Search for Synthesizable Molecules. Adv. Neural Inf. Process. Syst. 2019; pp 7937–7949.

(16) Bradshaw, J.; Paige, B.; Kusner, M. J.; Segler, M. H.; Hernández-Lobato, J. M. Barking Up the Right Tree: An Approach to Search Over Molecule Synthesis DAGs. *arXiv* **2020**, arXiv:2012.11522.

(17) Gottipati, S. K.; Sattarov, B.; Niu, S.; Pathak, Y.; Wei, H.; Liu, S.; Thomas, K. M.; Blackburn, S.; Coley, C. W.; Tang, J., et al. Learning To Navigate the Synthetically Accessible Chemical Space Using Reinforcement Learning. *arXiv* **2020**, arXiv:2004.12485.

(18) Maziarz, K.; Jackson-Flux, H.; Cameron, P.; Sirockin, F.; Schneider, N.; Stiefl, N.; Brockschmidt, M. Learning to Extend Molecular Scaffolds with Structural Motifs. *arXiv* **2021**, arXiv:2103.03864.

(19) Sacha, M.; Błaz, M.; Byrski, P.; Dabrowski-Tumanski, P.; Chrominski, M.; Loska, R.; Włodarczyk-Pruszynski, P.; Jastrzebski, S. Molecule Edit Graph Attention Network: Modeling Chemical Reactions as Sequences of Graph Edits. *J. Chem. Inf. Model.* **2021**, *61*, 3273–3284.

(20) Skiena, S. S. *The Algorithm Design Manual*; Springer London: London, 2008; pp 103–144.

(21) David, L.; Thakkar, A.; Mercado, R.; Engkvist, O. Molecular Representations in AI-Driven Drug Discovery: A Review and Practical Guide. *J. Cheminf.* **2020**, *12*, 1–22.

(22) RDKit: Open-Source Cheminformatics. `http://www.rdkit.org`, [Online; accessed 2020-05-18].

(23) Mercado, R.; Rastemo, T.; Lindelöf, E.; Klambauer, G.; Engkvist, O.; Chen, H.; Bjerrum, E. J. Graph Networks for Molecular Design. *Mach. Learn.: Sci. Technol.* **2020**,

(24) Sorokina, M.; Merseburger, P.; Rajan, K.; Yirik, M. A.; Steinbeck, C. COCONUT Online: Collection of Open Natural Products Database. *J. Cheminf.* **2021**, *13*, 1–13.

(25) Mercado, R.; Rastemo, T.; Lindelöf, E.; Klambauer, G.; Engkvist, O.; Chen, H.; Bjerrum, E. J. GraphINVENT. `https://github.com/MolecularAI/GraphINVENT`, [Online; accessed 2021-05-01].

(26) Riniker, S.; Landrum, G. A. Better informed distance geometry: using what we know to improve conformation generation. *J. Chem. Inf. Model.* **2015**, *55*, 2562–2574.

(27) Sauer, W. H.; Schwarz, M. K. Molecular Shape Diversity of Combinatorial Libraries: A Prerequisite for Broad Bioactivity. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 987–1003.

(28) Zhang, J.; Mercado, R.; Engkvist, O.; Chen, H. Comparative Study of Deep Generative Models on Chemical Space Coverage. *J. Chem. Inf. Model.* **0**, *0*, null, PMID: 34015916.

(29) Aldeghi, M.; Malhotra, S.; Selwood, D. L.; Chan, A. W. E. Two-and Three-Dimensional Rings in Drugs. *Chem. Biol. Drug Des.* **2014**, *83*, 450–461.

(30) Ertl, P. An Algorithm To Identify Functional Groups in Organic Molecules. *J. Cheminf.* **2017**, *9*, 1–7.

(31) Bertz, S. H. The First General Index of Molecular Complexity. *J. Am. Chem. Soc.* **1981**, *103*, 3599–3601.

(32) Kotsias, P.-C.; Arús-Pous, J.; Chen, H.; Engkvist, O.; Tyrchan, C.; Bjerrum, E. J. Direct Steering of De Novo Molecular Generation with Descriptor Conditional Recurrent Neural Networks. *Nature Mach. Int.* **2020**, *2*, 254–265.

(33) Kotsias, P.-C.; Bjerrum, E. J. DeepDrugCoder (DDC): Heteroencoder for Molecular Encoding and De Novo Generation. `https://github.com/pcko1/Deep-Drug-Coder`, [Online; accessed 2021-05-01].

Supporting Information Available

Abbreviations

- BFS : breadth-first search
- DFS : depth-first search
- GNN : graph neural network
- GTA : graph traversal algorithm
- MPNN : message-passing neural network
- NPR : normalized principle moment of inertia ratio
- PMI : principle moment of inertia

Additional COCONUT subset results

In this section we show some additional results for the aforementioned COCONUT subsets: percent uniqueness (Figure 4), and ring system and functional group coverage (Figure 5).

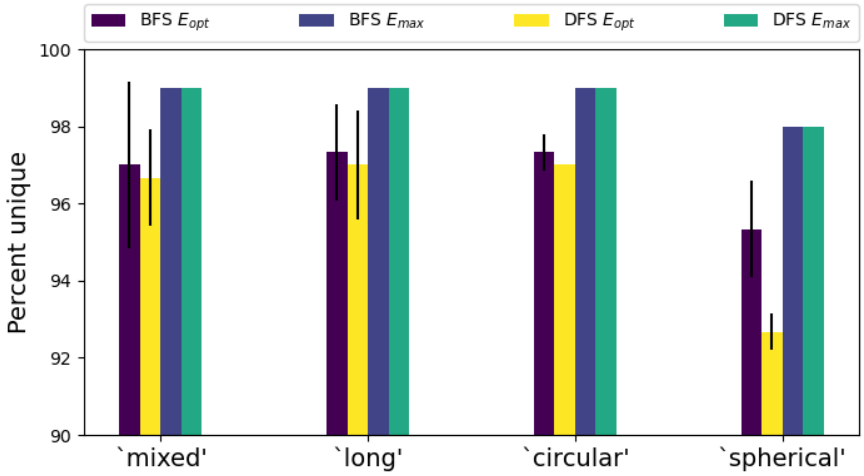

Figure 4: Comparison of the percent uniqueness of generated molecules for models trained on each COCONUT subset and evaluated at both E_{opt} and E_{max}. Error bars are the standard deviation from 3 different runs.

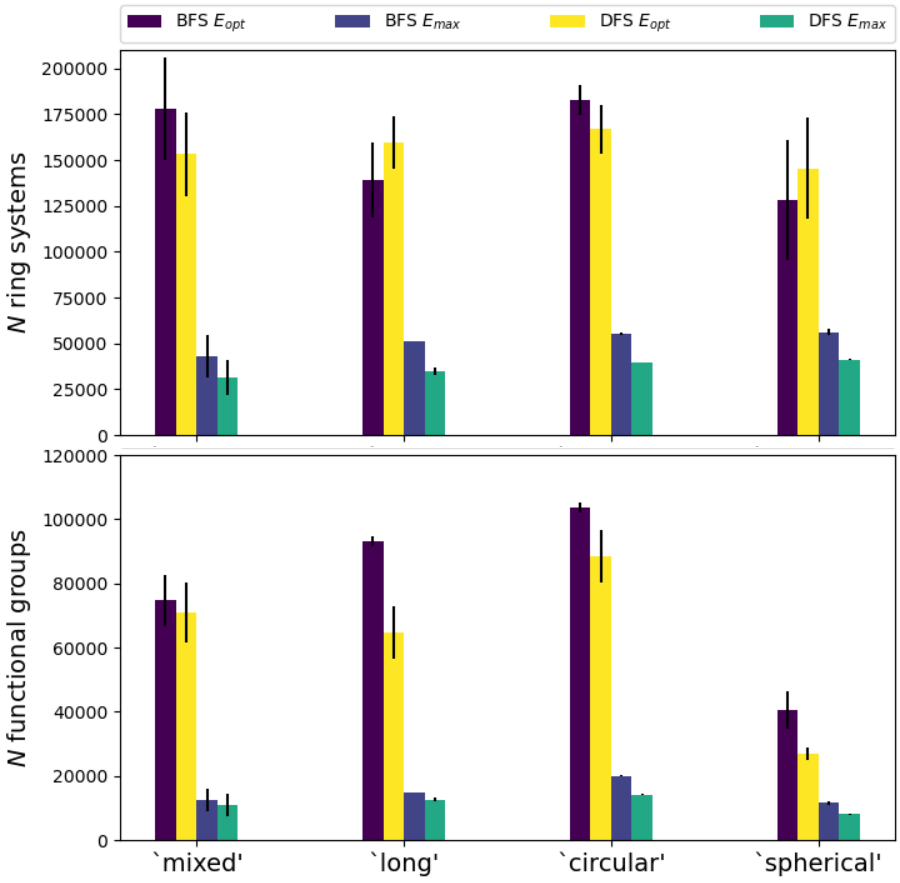

Figure 5: Comparison of the number of ring systems (top) and functional groups (bottom) found in the generated molecules for models trained on each COCONUT subset and evaluated at both E_{opt} and E_{max}. Error bars are the standard deviation from 3 different runs. Note the different y-axis scales.

Additionally, we show some analyses regarding the shape of the aforementioned COCONUT subsets (Figure 6) and the shape of the molecules generated by models trained on theses subsets after over-training (Figure 7). Note how, indeed, upon over-training models on each dataset using either a BFS or DFS (Figure 7), the PMI plots of generated molecules are almost identical to those of the reference datasets (Figure 6).

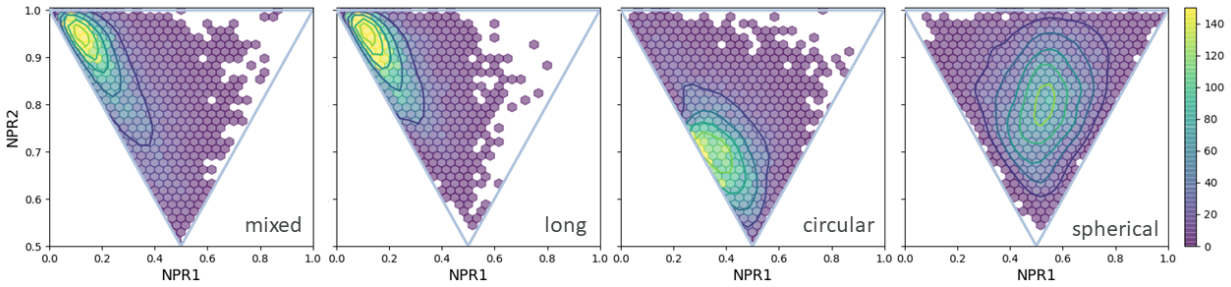

Figure 6: PMI plots for training set molecules for each COCONUT subset. Note that the differentiation between the different regimes isn't perfect as the NPRs depend on the specific conformers that were used to compute the PMIs, and as there is an element of randomness in the conformer generation algorithm, slightly different conformers may have been generated when creating the plot than were used for creating the different molecule classes (conformers were not saved). The contours are plotted over the hexbin histogram, where a hexagon is present if at least one structure was sampled in that regime; otherwise the colorbar indicates the number of structures in each bin (out of 10K random samples). Contours are generated from kernel density estimates on the data.

Finally, breaking down the molecules generated by molecular complexity led to some interesting results. As a measure of molecular complexity, we used the BertzCT.[31] In Figure 8, we show histograms of the BertzCT indices for the sampled and reference molecules when using the COCONUT 10K 'mixed' subset. What we find is that, at E_{opt} (left), the molecular complexity of sampled molecules using either BFS or DFS is much lower on average than the molecular complexity of the reference (i.e. training) set. However, at E_{max}, we find that the molecular complexity of the three distributions match. We found this general trend to hold for the 'spherical,' 'long,' and 'circular' 10K datasets. On average, the average complexity of the sampled structures was greater when using a BFS instead of a DFS, and only in the case of the 'circular' dataset was it the other way around.

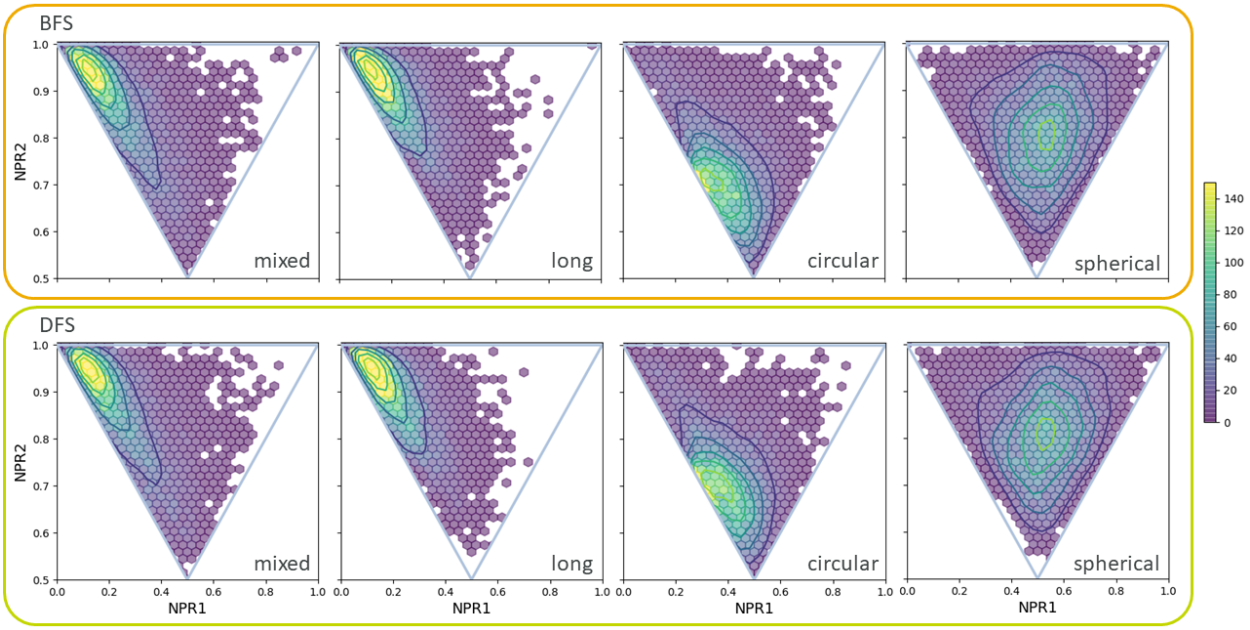

Figure 7: PMI plots for generated molecules for models trained on each COCONUT subset at E_{max} (top: BFS, bottom: DFS). Although 1M molecules were sampled from each model, results are only plotted for 10K random samples in each case. The contours are plotted over the hexbin histogram, where a hexagon is present if at least one structure was generated in that regime; otherwise the colorbar indicates the number of structures in each bin (out of 10K random samples). Contours are generated from kernel density estimates on the data.

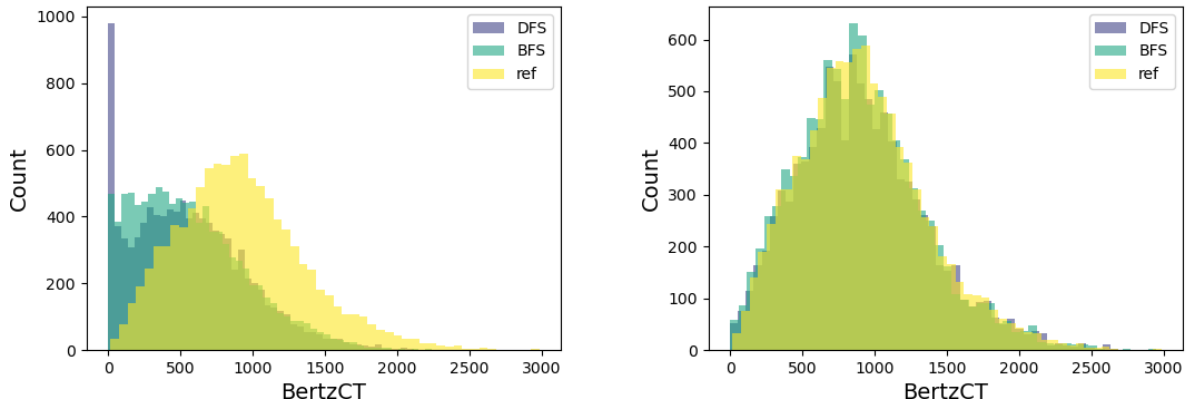

Figure 8: Histograms of the calculated molecular complexity, as approximated by BertzCT, for 10K sampled molecules using models trained on the COCONUT 10K 'mixed' subset at E_{opt} (left) and E_{max} (right). 'ref' refers to the training set.

Example loss curves

Example loss curves for the COCONUT 10K subset ('mixed') are shown in Figure 9. We generally observed that:

1. models over-train much more quickly when trained using a BFS than a DFS,
2. models reach the same final training loss whether using a BFS or a DFS,
3. and both the training and validation losses at E_{opt} are greater for models trained using a BFS than a DFS.

It is interesting that the BFS-based models perform better despite having a 'worse' loss curve for all datasets. However, when comparing BFS and DFS, the underlying data used for each model is not exactly the same due to the different molecular decoding routes generated by each GTA, and as such, it is not entirely meaningful to compare the absolute values of the loss achieved by models trained using different GTAs. For brevity, we do not show loss curves for the other datasets, as they all displayed the same trends.

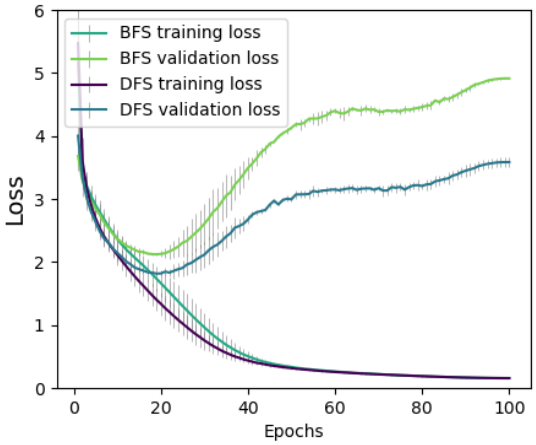

Figure 9: Loss curves for the 'mixed' COCONUT subset. The error bars are shown in gray for each curve and are the standard deviation from 3 different runs.

Larger dataset results: COCONUT 50K

In order to verify that our observations held for larger dataset, we re-ran the same experiments done on the 10K COCONUT subset on a larger subset. This larger subset consisted of 50K randomly sampled COCONUT molecules in the training set, 10K in the test set, and 1K in the validation set (sampled after applying the same filters as before). For this 50K subset, we found $E_{opt} = 15$ and $E_{max} = 50$.

Here we show the percent validity and molecular coverage (Figure 10), the ring system and functional group coverage (Figure 11), and the shape analysis (Figure 12). While not visualized, the percent uniqueness of 1M generated samples for all models trained on the COCONUT 50K subset was 99%, irrespective of evaluation epoch or GTA.

We note that in Figure 12, the differences between the two GTAs at E_{opt} (top two figures) are subtle; to draw the reader's attention to these differences, we highlight that in the PMI plots, we see a broader distribution of NPRs (less peaked) for the BFS-generated molecules (left) than the DFS-generated molecules (right). Additionally, the BFS-generated molecules cover more of the PMI space and have a slightly greater bias towards circular and spherical structures, which can be verified by seeing the greater density of hexbins in the top-right corner and right-most edge of the PMI plot.

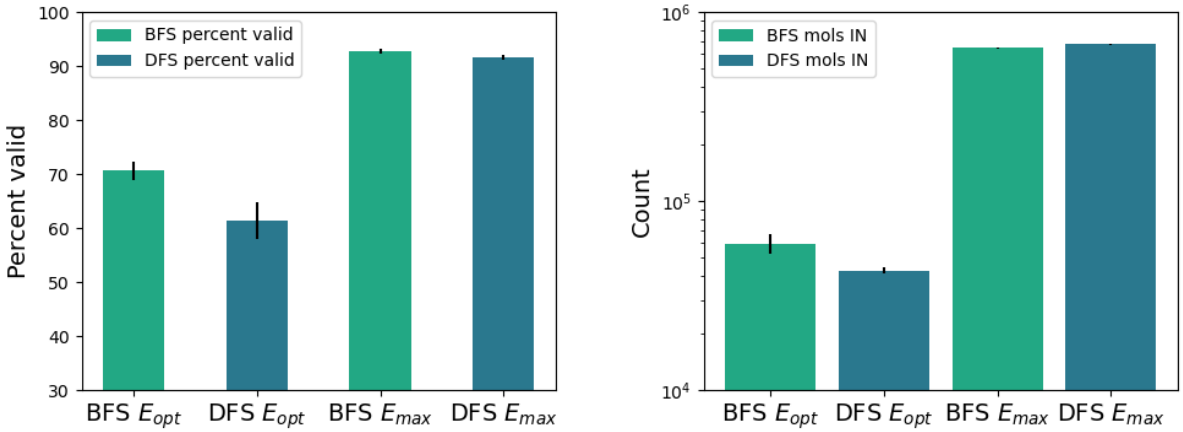

Figure 10: Comparison of the percent validity (left) and molecular coverage (right) of 1M generated molecules for models trained on the COCONUT 50K subset and evaluated at both E_{opt} and E_{max}. Error bars are the standard deviation from 3 different runs.

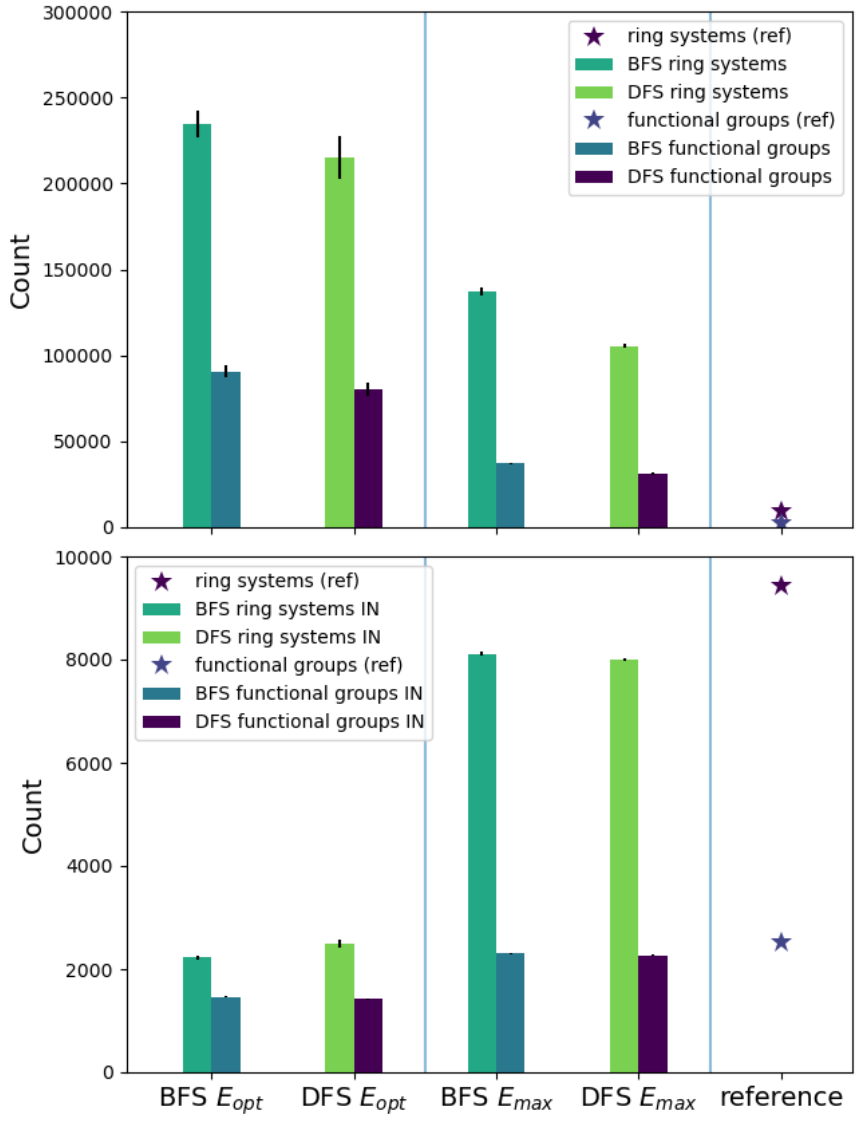

Figure 11: Comparison of the ring system and functional group coverage of generated molecules for models trained on the COCONUT 50K subset and evaluated at both E_{opt} and E_{max}. (top) Count of total number of ring systems and functional groups found in sampled structures at each epoch. (bottom) Count of total number of ring systems and functional groups in the generated structures which are *also* found in the reference (train + test + valid) set, hence why we refer to these as the 'IN' sets and the y-scale is much smaller than in the top figure. Error bars are the standard deviation from 3 different runs.

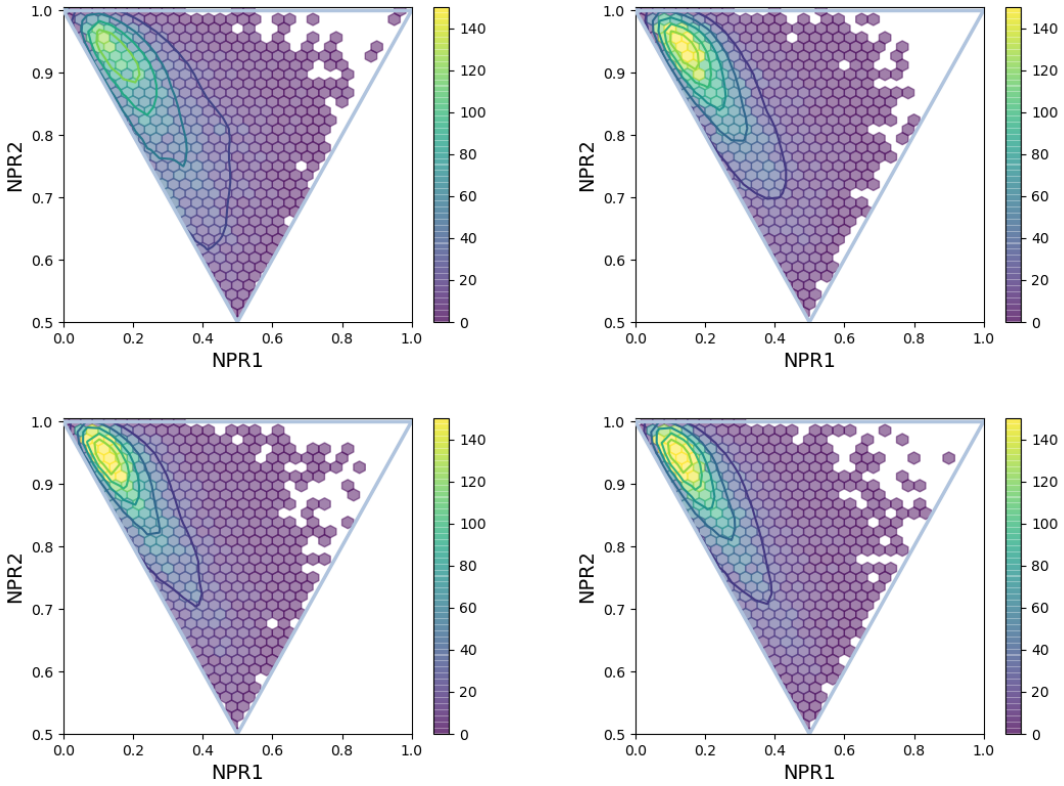

Figure 12: PMI plots for models trained on the COCONUT 50K subset at E_{opt} (top) and E_{max} (bottom), with (left) BFS results and (right) DFS results. Although 1M molecules were sampled from each model, results are only plotted for 10K random samples in each case. The contours are plotted over the hexbin histogram, where a hexagon is present if at least one structure was generated in that regime; otherwise the colorbar indicates the number of structures in each bin (out of 10K random samples). Contours are generated from kernel density estimates on the data.

Drug-like dataset results: DRD2 actives

In order to verify that our observations held for a drug-like dataset, we re-ran the same experiments done on the COCONUT subsets for a set of DRD2 actives.[32,33] This dataset consists of 4311 known DRD2 active molecules, split into 3448 molecules in the training set, 432 in the test set, and 431 in the validation set. For this dataset, we found $E_{opt} = 30$ and $E_{max} = 100$.

Molecules in this dataset consist of the following:

- *atom types*: C, N, O, F, S, Cl, and Br
- *formal charges*: -1, 0, and 1
- *max nodes*: 72.

Here we show the percent validity and molecular coverage (Figure 13), the ring system and functional group coverage (Figure 14), and the shape analysis (Figure 15).

As in Figure 12, we note that in Figure 15, the differences between the two GTAs at E_{opt} (top two figures) are subtle; to draw the reader's attention to these differences, we highlight that in the PMI plots, we see a slightly broader distribution of NPRs (less peaked) for the BFS-generated molecules (left) than the DFS-generated molecules (right). Additionally, the BFS-generated molecules cover more of the PMI space and have a slightly greater bias towards circular and spherical structures, which can be verified by seeing the greater density of hexbins in the top-right corner and right-most edge of the PMI plot.

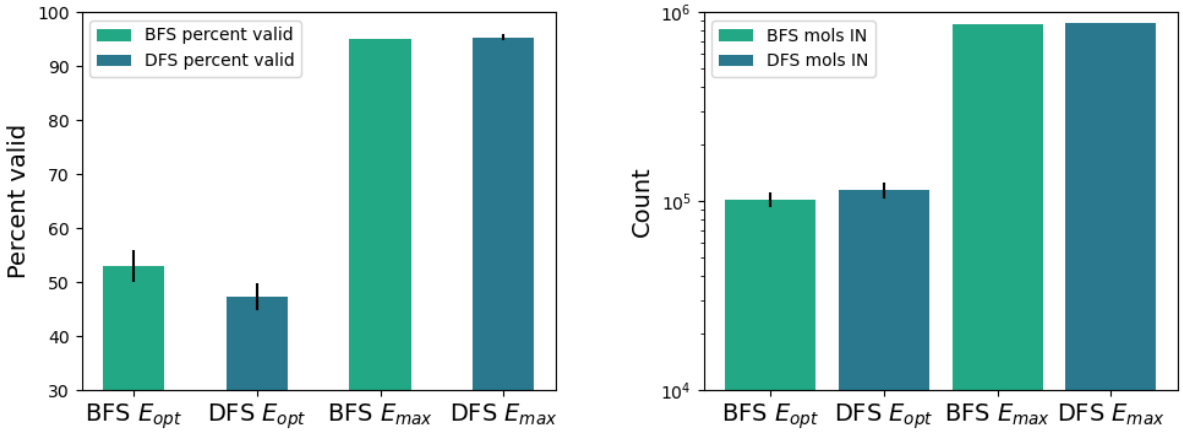

Figure 13: Comparison of the percent validity (left) and molecular coverage (right) of 1M generated molecules for models trained on the COCONUT 50K subset and evaluated at both E_{opt} and E_{max}. Error bars are the standard deviation from 3 different runs.

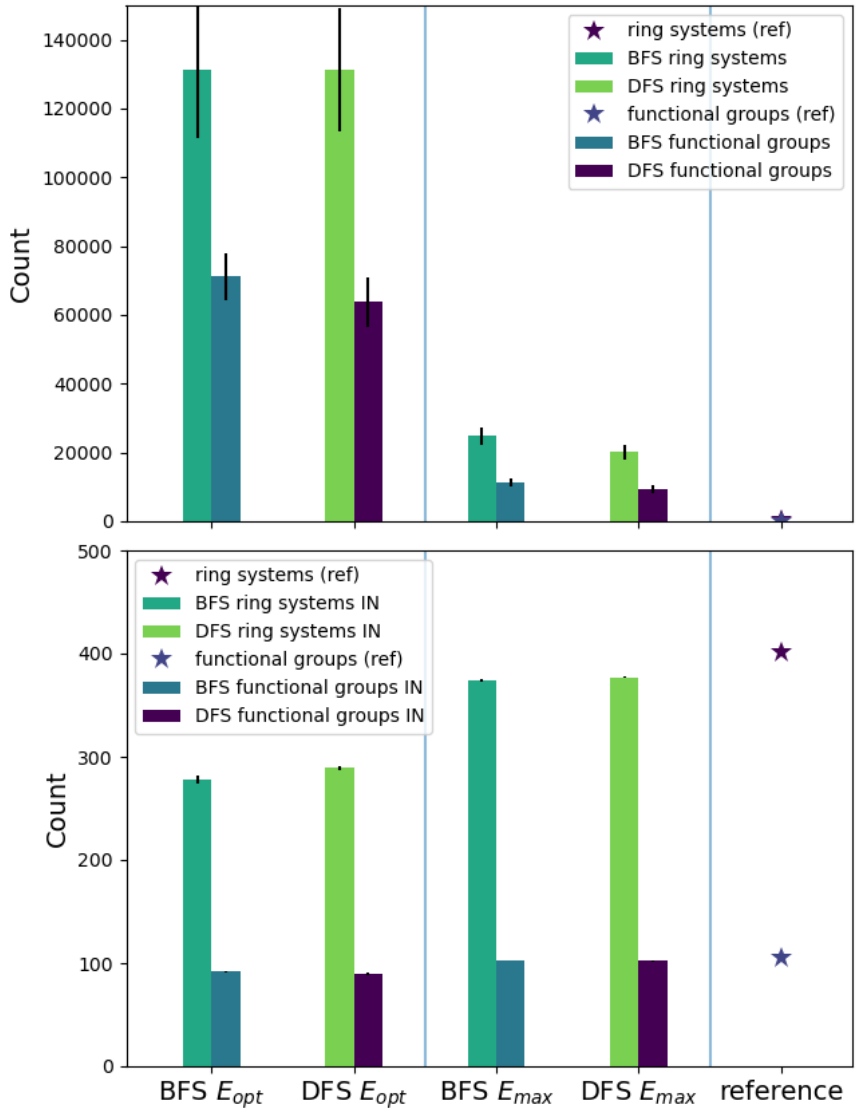

Figure 14: Comparison of the ring system and functional group coverage of generated molecules for models trained on the DRD2 actives dataset and evaluated at both E_{opt} and E_{max}. (top) Count of total number of ring systems and functional groups found in sampled structures at each epoch. (bottom) Count of total number of ring systems and functional groups in the generated structures which are *also* found in the reference (train + test + valid) set, hence why we refer to these as the 'IN' sets and the y-scale is much smaller than in the top figure. Error bars are the standard deviation from 3 different runs.

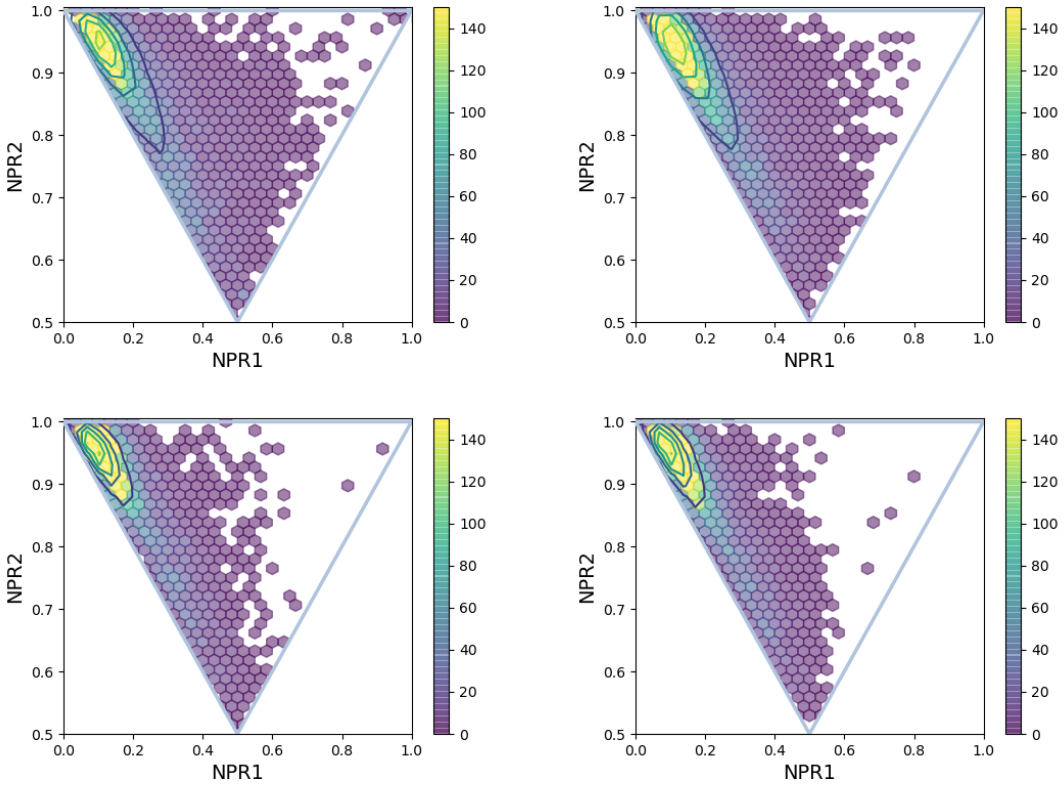

Figure 15: PMI plots for models trained on the DRD2 actives dataset at E_{opt} (top) and E_{max} (bottom), with (left) BFS results and (right) DFS results. Although 1M molecules were sampled from each model, results are only plotted for 10K random samples in each case. The contours are plotted over the hexbin histogram, where a hexagon is present if at least one structure was generated in that regime; otherwise the colorbar indicates the number of structures in each bin (out of 10K random samples). Contours are generated from kernel density estimates on the data.

Example molecules

Examples of randomly sampled molecules from the various COCONUT subsets are shown in Figure 16. Examples of molecules sampled from the models trained on the 'mixed' COCONUT subset using either (a) BFS or DFS, at (b) E_{opt} or E_{max}, are shown in 17. Note that as the structures are randomly sampled, that there can be 'ugly' yet chemically valid molecules in the mix, particularly in the case of E_{opt} in Figure 17.

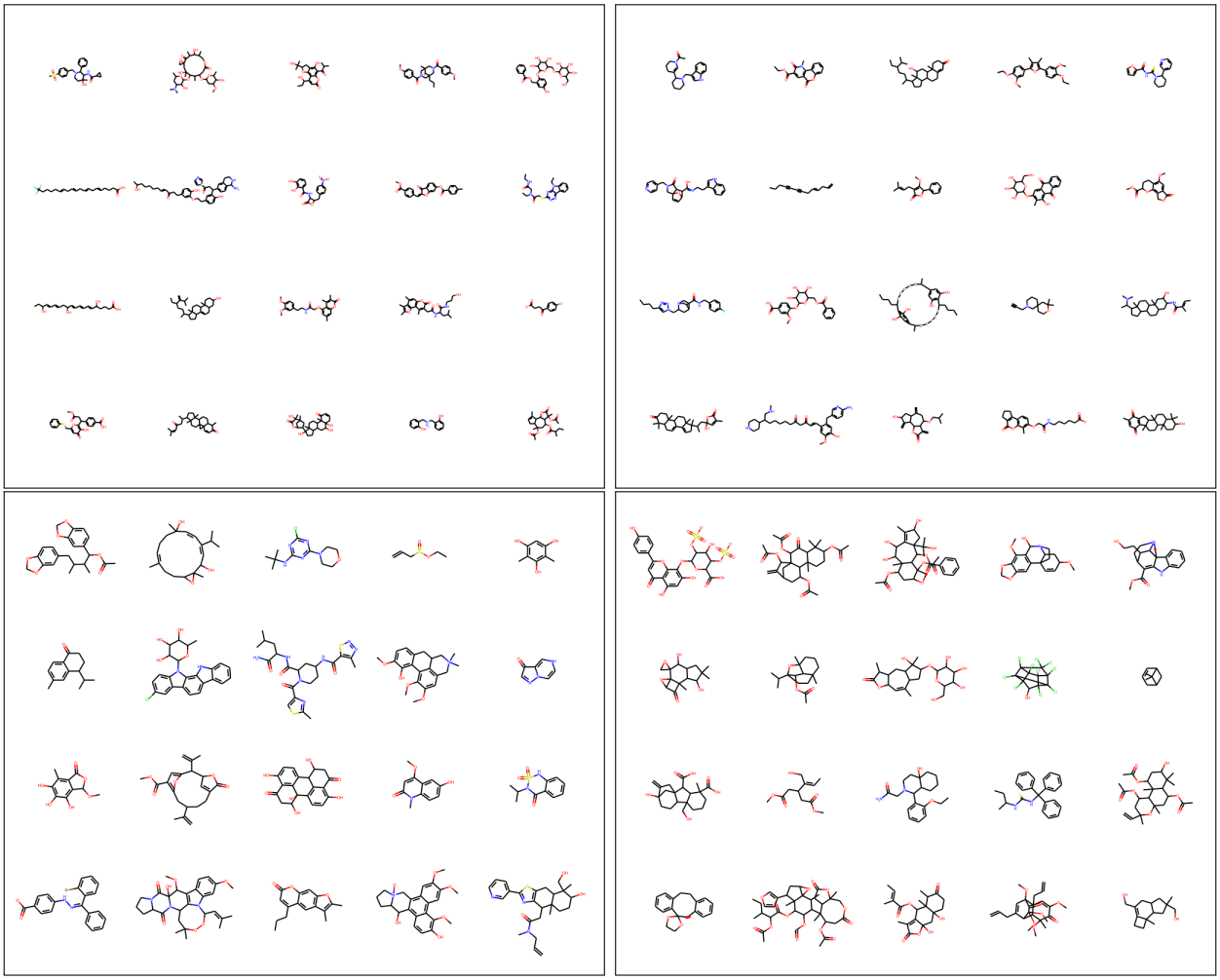

Figure 16: Randomly sampled molecules found in each COCONUT subset used for training. (top left) 'mixed' dataset; (top right) 'long' dataset; (bottom left) 'circular' dataset; (bottom right) 'spherical' dataset.

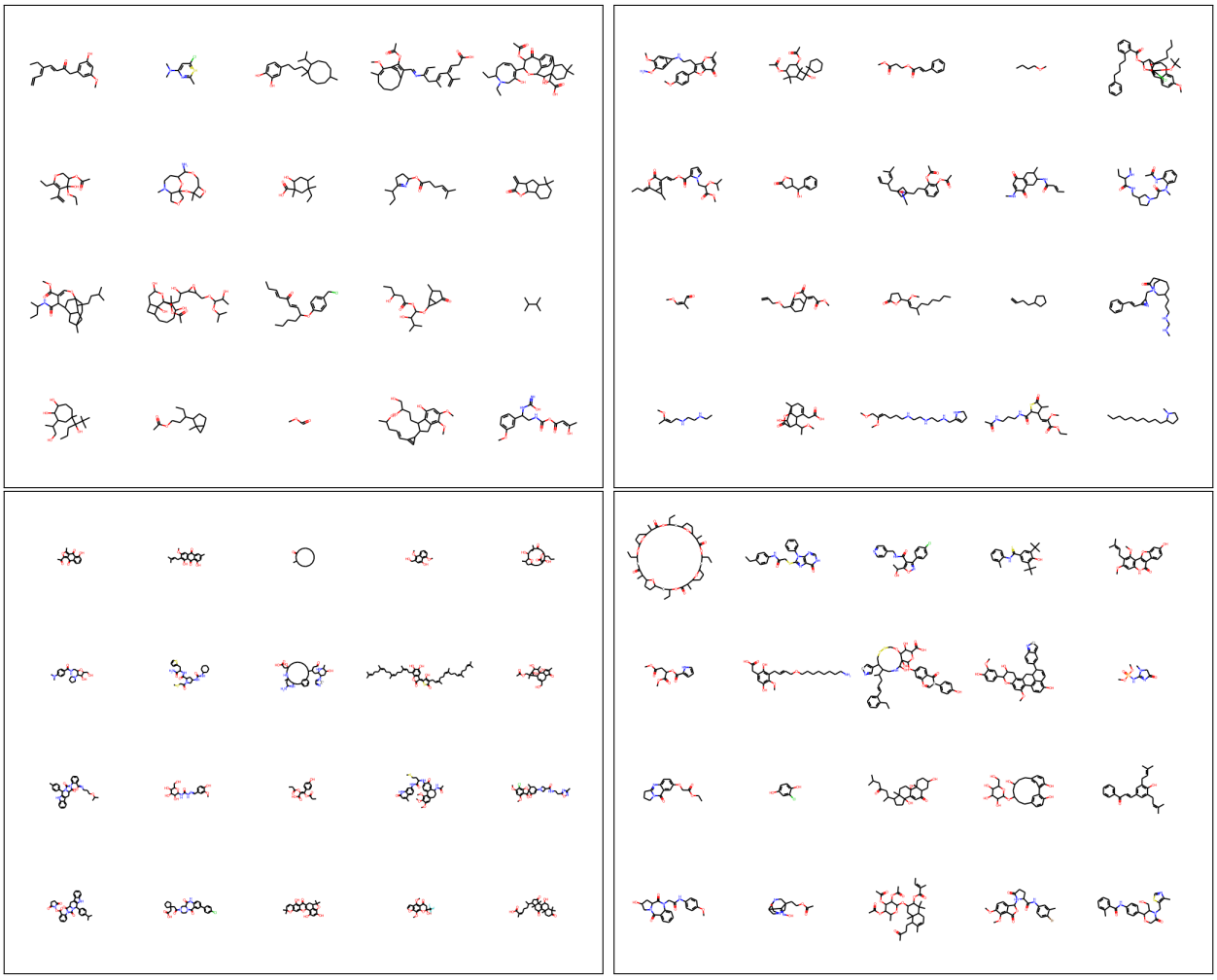

Figure 17: Randomly sampled molecules generated by models trained on the 'mixed' COCONUT 10K dataset using (top left) BFS @ E_{opt}; (top right) DFS @ E_{opt}; (bottom left) BFS @ E_{max}; (bottom right) using DFS @ E_{max}.

Graphical TOC Entry