DIFFERENTIABLE SCAFFOLDING TREE FOR MOLECULAR OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The structural design of functional molecules, also called molecular optimization, is an essential chemical science and engineering task with important applications, such as drug discovery. Deep generative models and combinatorial optimization methods achieve initial success but still struggle with directly modeling discrete chemical structures and often heavily rely on brute-force enumeration. The challenge comes from the discrete and non-differentiable nature of molecule structures. To address this, we propose differentiable scaffolding tree (`DST`) that utilizes a learned knowledge network to convert discrete chemical structures to locally differentiable ones. `DST` enables a gradient-based optimization on a chemical graph structure by back-propagating the derivatives from the target properties through a graph neural network (GNN). Our empirical studies show the gradient-based molecular optimizations are both effective and sample efficient. Furthermore, the learned graph parameters can also provide an explanation that helps domain experts understand the model output.

1 INTRODUCTION

The structural design of new functional molecules, also called molecular optimization, is the key to many scientific and engineering challenges, such as finding energy storage materials (Hachmann et al., 2011; Janet et al., 2020), small molecule pharmaceutics (Kuntz, 1992; Zhavoronkov et al., 2019), and environment-friendly material (Zimmerman et al., 2020). The objective is to identify novel molecular structures with desirable chemical or physical properties (Gómez-Bombarelli et al., 2018; Dai et al., 2018; Jin et al., 2018; You et al., 2018; Jin et al., 2019; Shi et al., 2020; Zhou et al., 2019; Jin et al., 2020; Zang & Wang, 2020; Xie et al., 2021). Recent advances in deep generative models (DGM) allow learning the distribution of molecules and optimizing the latent embedding vectors of molecules. Models in this category are exemplified by the variational autoencoder (VAE) (Gómez-Bombarelli et al., 2018; Dai et al., 2018; Jin et al., 2018; 2020) and generative adversarial network (GAN) (De Cao & Kipf, 2018). On the other hand, because of the discrete and not explicitly combinatorial nature of the enormous chemical space, applying combinatorial optimization algorithms with some structure enumeration has been the predominant approach (You et al., 2018; Jensen, 2019; Zhou et al., 2019; Nigam et al., 2020; Xie et al., 2021). Deep learning models have also been used to guide these combinatorial optimization algorithms. For example, You et al. (2018); Zhou et al. (2019); Jin et al. (2020); Gottipati et al. (2020) tried to solve the problem with deep reinforcement learning; Nigam et al. (2020) enhanced a genetic algorithm with a neural network as a discriminator; Xie et al. (2021); Fu et al. (2020) approached the problem with Markov Chain Monte Carlo (MCMC) to explore the target distribution guided by graph neural networks.

Despite the initial success of these previous attempts, the following limitations remain: (1) deep generative models optimize the molecular structures in a learned latent space, which requires the latent space to be smooth and discriminative. Training such models needs carefully designed networks and well-distributed datasets. (2) most combinatorial optimization algorithms, featured by evolutionary learning methods (Nigam et al., 2020; Jensen, 2019; Xie et al., 2021; Fu et al., 2020), exhibit random-walk behavior, and leverage trial-and-error strategies to explore the discrete chemical space. The recent deep reinforcement learning methods (You et al., 2018; Zhou et al., 2019; Jin et al., 2020; Gottipati et al., 2020) aim to remove random-walk search using a deep neural network to guide the searching. It is challenging to design the effective reward function into the objective (Jin et al., 2020). (3) Most existing methods require a great number of oracle calls (a property evaluator; see

Def. 1) to proceed with an efficient search. However, realistic oracle functions, evaluating with either experiments or high-fidelity computational simulations, are usually expensive. Examples include using biological assays to determine the potency of drug candidates (Wang et al., 2017), or conducting electronic structure calculation to determine photoelectric properties (Long et al., 2011).

We propose differentiable scaffolding tree (DST) to address these challenges, where we define a differentiable scaffolding tree for molecular structure and utilize a trained GNN to obtain the local derivative that enables continuous optimization. The main contributions are summarized as:

• We propose the differentiable scaffolding tree to define a local derivative of a chemical graph. This concept enables a gradient-based optimization of a discrete graph structure.

• We present a general molecular optimization strategy utilizing the local derivative defined by the differentiable scaffolding tree. This strategy leverages the property landscape's geometric structure and suppresses the random-walk behavior, exploring the chemical space more efficiently. We also incorporate a determinantal point process (DPP) based selection strategy to enhance the diversity of generated molecules.

• We demonstrate encouraging preliminary results on *de novo* molecular optimization with multiple computational objective functions. The local derivative shows consistency with chemical intuition, providing interpretability of the chemical structure-property relationship. Our method also requires less oracle calls, maintaining good performance in limited oracle settings.

2 Related Work

Existing molecular optimization methods can mainly be categorized as deep generative models and combinatorial optimization methods.

Deep generative models model a distribution of general molecular structure with a deep network model so that one can generate molecules by sampling from the learned distribution. Typical algorithms include variational autoencoder (VAE), generative adversarial network (GAN), energy-based models, flow-based model (Gómez-Bombarelli et al., 2018; Jin et al., 2018; De Cao & Kipf, 2018; Segler et al., 2018; Jin et al., 2019; Honda et al., 2019; Madhawa et al., 2019; Shi et al., 2020; Jin et al., 2020; Zang & Wang, 2020; Kotsias et al., 2020; Chen et al., 2021; Liu et al., 2021; Bagal et al., 2021). Shen et al. (2021) also leverages inverse learning based on SELFIES representation. However, its performance is not satisfactory, primarily due to the failure of training an adequate surrogate oracle. In addition, DGMs can leverage Bayesian optimization in latent spaces to optimize latent vectors and reconstruct to obtain the optimized molecules (Jin et al., 2018). However, such approaches usually require a smooth and discriminative latent space and thus an elaborate network architecture design and well-distributed data set. Also, as they learn the reference data distribution, their ability to explore diverse chemical space is relatively limited, evidenced by the recent molecular optimization benchmarks (Brown et al., 2019; Huang et al., 2021).

Combinatorial optimization methods mainly include deep reinforcement learning (DRL) (You et al., 2018; Zhou et al., 2019; Jin et al., 2020; Gottipati et al., 2020) and evolutionary learning methods (Nigam et al., 2020; Jensen, 2019; Xie et al., 2021; Fu et al., 2020). They both formulate molecule optimization as a discrete optimization task. Specifically, they modify molecule substructures (or tokens in a string representation (Weininger, 1988)) locally, with an oracle score or a policy/value network to tell if they keep it or not. Due to the discrete nature of the formulation, most of them conduct an undirected search (random-walk behavior), while some recent ones like reinforcement learning try to guide the searching with a deep neural network, aiming to rid the random-walk nature. However, it is challenging to incorporate the learning objective target into the guided search. Those algorithms still require massive numbers of oracle calls, which is computationally inefficient during the inference time (Korovina et al., 2020). Our method, DST, falls into this category, explicitly leverages the objective function landscape and conducts an efficient goal-oriented search. Instead of operating on molecular substructure or tokens, we define the search space as a set of binary and multinomial variables to indicate the existence and identity of nodes respectively, and make it locally differentiable with a learned GNN as a surrogate of the oracle. This problem formulation can find its root in conventional computer-aided molecular design algorithms with branch-and-bound algorithms as solutions (Sinha et al., 1999; Sahinidis & Tawarmalani, 2000).

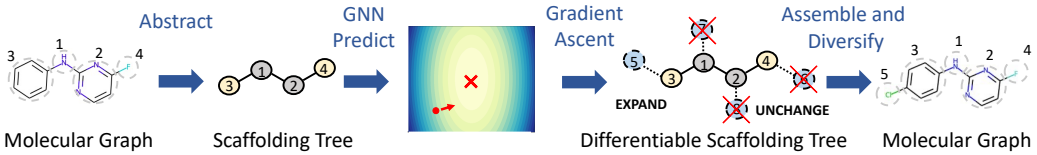


Figure 1: Illustration of the overall approach: During inference, we construct the corresponding scaffolding tree and differentiable scaffolding tree (DST) for each molecule. We optimize each DST along its gradient back-propagated from the GNN and sample scaffolding trees from the optimized DST. After that, we assemble trees into molecules and diversify them for the next iteration.

3 METHOD

We first introduce the formulation of molecular optimization and *differentiable scaffolding tree* (DST) in Section 3.1, illustrate the pipeline in Figure 1, then describe the key steps following the order:

- **Oracle GNN construction:** We leverage GNNs to imitate property oracles, which are targets of molecular optimization (Section 3.2). Oracle GNN is trained once and for all. The training is separately from optimizing DST below.
- **Optimizing differentiable scaffolding tree:** We formulate the discrete molecule optimization into *a locally differentiable* problem with a differentiable scaffolding tree (DST). Then a DST can be optimized by the gradient back-propagated from oracle GNN (Section 3.3).
- **Molecule Diversification** After that, we describe how we design a *determinantal point process (DPP)* based method to output diverse molecules for iterative learning (Section 3.4).

3.1 PROBLEM FORMULATION AND NOTATIONS

3.1.1 Molecular optimization problem Oracles are the objective functions for molecular optimization problems, e.g., QED quantifying a molecule's drug-likeness (Bickerton et al., 2012).

Definition 1 (Oracle $\mathcal{O}$)**.** *Oracle $\mathcal{O}$ is a black-box function that evaluates certain chemical or biological properties of a molecule X and returns the ground truth property $\mathcal{O}(X)$.*

In realistic discovery settings, the oracle acquisition cost is usually not negligible. Suppose we want to optimize P molecular properties specified by oracle $\mathcal{O}_1, \cdots, \mathcal{O}_P$, we can formulate a multi-objective molecule optimization problem through scalarization as represented in Eq. (1),

$$\arg\max_{X \in \mathcal{Q}} \; F(X; \mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_P) = f(\mathcal{O}_1(X), \cdots, \mathcal{O}_P(X)), \tag{1}$$

where X is a molecule, $\mathcal{Q}$ denotes the set of valid molecules; f is the composite objective combining all the oracle scores, e.g., the mean value of P oracle scores.

3.1.2 Scaffolding Tree The basic mathematical description of a molecule is molecular graph, which contains atoms as nodes and chemical bonds as edges. However, molecular graphs are not easy to generate explicitly as graphs due to the presence of rings, relatively large size, and chemical validity constraints. For ease of computation, we convert a molecular graph to a scaffolding tree as a higher-level representation, a tree of substructures, following Jin et al. (2018; 2019).

Definition 2 (**Substructure**)**.** *Substructures can be either an atom or a single ring. The substructure set is denoted $\mathcal{S}$ (vocabulary set), which covers frequent atoms and single rings in drug-like molecules.*

Definition 3 (**Scaffolding Tree $\mathcal{T}$**)**.** *A scaffolding tree, $\mathcal{T}_X$, is a spanning tree whose nodes are substructures. It is higher-level representation of molecular graph X.*

$\mathcal{T}_X$ is represented by (i) node indicator matrix, (ii) adjacency matrix, and (iii) node weight vector. We distinguish leaf and non-leaf nodes in $\mathcal{T}_X$. Among the K [1] nodes in $\mathcal{T}_X$, there are K_{leaf} leaf nodes (nodes connecting to only one edge) and $K - K_{\text{leaf}}$ non-leaf nodes (otherwise). The sets of leaf nodes and non-leaf nodes are denoted $\mathcal{V}_{\text{leaf}}$ and $\mathcal{V}_{\text{nonleaf}}$ correspondingly.

[1] K depends on molecular graph. During optimization (Section 3.3 and 3.4), after molecular structure changes, K is updated.

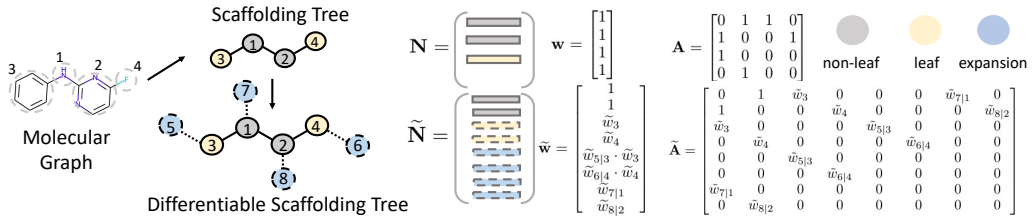


Figure 2: Example of differentiable scaffolding tree. We show non-leaf nodes (grey), leaf nodes (yellow), expansion nodes (blue). The dashed nodes and edges are learnable, corresponding to nodes' identity and existence, respectively. $\widetilde{\mathbf{w}}$ and $\widetilde{\mathbf{A}}$ share the learnable parameters $\{\widehat{\mathbf{w}}_3, \widehat{\mathbf{w}}_4, \widehat{\mathbf{w}}_{5|3}, \widehat{\mathbf{w}}_{6|4}, \widehat{\mathbf{w}}_{7|1}, \widehat{\mathbf{w}}_{8|2}\}$.

Definition 4. ***Node indicator matrix*** $\mathbf{N}$ *is decomposed as* $\mathbf{N} = \begin{pmatrix} \mathbf{N}_{nonleaf} \\ \mathbf{N}_{leaf} \end{pmatrix} \in \{0,1\}^{K \times |\mathcal{S}|}$, *where* $\mathbf{N}_{nonleaf} \in \{0,1\}^{(K-K_{leaf}) \times |\mathcal{S}|}$ *corresponds to non-leaf nodes while* $\mathbf{N}_{leaf} \in \{0,1\}^{K_{leaf} \times |\mathcal{S}|}$ *corresponds to leaf nodes. Each row of* $\mathbf{N}$ *is a one-hot vector, indicating which substructure the node belongs to.*

Definition 5. ***Adjacency matrix*** *is denoted* $\mathbf{A} \in \{0,1\}^{K \times K}$. $A_{ij} = 1$ *indicates the* i*-th node and* j*-th node are connected while 0 indicates unconnected.*

Definition 6. ***Node weight vector***, $\mathbf{w} = [1, \cdots, 1]^\top \in \mathbb{R}^K$, *indicates the* K *nodes in scaffolding tree are equally weighted.*

3.1.3 Differentiable scaffolding tree Similar to a scaffolding tree, a differentiable scaffolding tree (DST) also contains (i) node indicator matrix, (ii) adjacency matrix, and (iii) node weight vector, but with additional expansion nodes. Specifically, while inheriting leaf node set $\mathcal{V}_{\text{leaf}}$ and non-leaf node set $\mathcal{V}_{\text{nonleaf}}$ from the original scaffolding tree, we add expansion nodes and form *expansion node set*, $\mathcal{V}_{\text{expand}} = \{u_v | v \in \mathcal{V}_{\text{leaf}} \cup \mathcal{V}_{\text{nonleaf}}\}, |\mathcal{V}_{\text{expand}}| = K_{\text{expand}} = K$, where u_v is connected to v in the original scaffolding tree. We also define *differentiable edge set*, $\Lambda = \{(v, v') \mid v \in \mathcal{V}_{\text{leaf}} \text{ OR } v' \in \mathcal{V}_{\text{expand}};\ v, v' \text{ are connected}\}$ to incorporate all the edges involving leaf-nonleaf node and leaf/nonleaf-expansion node connections. To make it locally differentiable, we modify the tree parameters from two aspects: (A) *node identity* and (B) *node existence*. Figure 2 shows an example to illustrate DST.

(A) We enable optimization on *node identity* by allowing the corresponding node indicator matrix learnable:

Definition 7. ***Differentiable node indicator matrix*** $\widetilde{\mathbf{N}}$ *takes the form:*

$$\widetilde{\mathbf{N}} = \begin{pmatrix} \widetilde{\mathbf{N}}_{nonleaf} \\ \widetilde{\mathbf{N}}_{leaf} \\ \widetilde{\mathbf{N}}_{expand} \end{pmatrix} \in \mathbb{R}_+^{(K+K_{expand}) \times |\mathcal{S}|}, \quad \sum_{j=1}^{|\mathcal{S}|} \widetilde{\mathbf{N}}_{ij} = 1, \quad K = K_{expand}. \tag{2}$$

$\widetilde{\mathbf{N}}_{nonleaf} = \mathbf{N}_{nonleaf} \in \{0,1\}^{(K-K_{leaf}) \times |\mathcal{S}|}$ *are fixed, equal to the part in the original scaffolding tree, each row is a one-hot vector, indicating that we fix all the non-leaf nodes. In contrast, both* $\widetilde{\mathbf{N}}_{expand}$ *and* $\widetilde{\mathbf{N}}_{leaf}$ *are learnable, we use softmax activation to implicitly encode the constraint* $\sum_j \widetilde{\mathbf{N}}_{ij} = 1$ *i.e.,* $\widetilde{\mathbf{N}}_{ij} = \frac{\exp(\widehat{\mathbf{N}_{ij}})}{\sum_{j'=1}^{|\mathcal{S}|} \exp(\widehat{\mathbf{N}_{i,j'}})}$, $\widehat{\mathbf{N}}$ *are the parameters to learn. This constraint guarantee that each row of* $\widetilde{\mathbf{N}}$ *is a valid substructures' distribution.*

(B) We enable optimization on *node existence* by assigning learnable weights for the leaf and expansion nodes, construct adjacency matrix and node weight vector based on those values:

Definition 8. ***Differentiable adjacency matrix*** $\widetilde{\mathbf{A}} \in \mathbb{R}^{(K+K_{expand}) \times (K+K_{expand})}$ *takes the form:*

$$\widetilde{\mathbf{A}}_{ij} = \widetilde{\mathbf{A}}_{ji} = \begin{cases} 1/0, & (i,j) \notin \Lambda \ \ \textit{0:disconnected, 1:connected} \\ \sigma(\widehat{\mathbf{w}}_i), & (i,j) \in \Lambda, i \in \mathcal{V}_{leaf},\ j \in \mathcal{V}_{nonleaf} \\ \sigma(\widehat{\mathbf{w}}_{i|j}), & (i,j) \in \Lambda, i \in \mathcal{V}_{expand},\ j \in \mathcal{V}_{leaf} \cup \mathcal{V}_{nonleaf} \end{cases} \tag{3}$$

where Λ is the differentiable edge set defined above, Sigmoid function $\sigma(\cdot)$ imposes the constraint $0 \leq \widetilde{\mathbf{A}}_{ij} \leq 1$. $\widehat{\mathbf{w}} \in \mathbb{R}^{K_{leaf}+K_{expand}}$ are the parameters, each leaf node and expansion node has one learnable parameter. For connected i and j, when $i \in \mathcal{V}_{leaf},\ j \in \mathcal{V}_{nonleaf}$, $\widetilde{\mathbf{A}}_{ij} = \sigma(\widehat{\mathbf{w}}_i)$ measures the existence probability of leaf node i; when $i \in \mathcal{V}_{expand}, j \in \mathcal{V}_{leaf} \cup \mathcal{V}_{nonleaf}$, $\widetilde{\mathbf{A}}_{ij} = \sigma(\widehat{\mathbf{w}}_{i|j})$ measures the conditional probability of the existence of expand node i given the original node j. When j is a leaf node, it naturally embeds the inheritance relationship between the leaf node and the corresponding expansion node.

Definition 9. ***Differentiable node weight vector*** $\widetilde{\mathbf{w}} \in \mathbb{R}^{K+K_{expand}}$ *takes the form:*

$$\widetilde{\mathbf{w}}_i = \begin{cases} 1, & i \in \mathcal{V}_{nonleaf} \\ \sigma(\widehat{\mathbf{w}}_i), & i \in \mathcal{V}_{leaf} \\ \sigma(\widehat{\mathbf{w}}_{i|j})\sigma(\widehat{\mathbf{w}}_j), & i \in \mathcal{V}_{expand}, j \in \mathcal{V}_{leaf}, \quad (i,j) \in \Lambda, \\ \sigma(\widehat{\mathbf{w}}_{i|j})\tilde{\mathbf{w}}_j = \sigma(\widehat{\mathbf{w}}_{i|j}), & i \in \mathcal{V}_{expand}, j \in \mathcal{V}_{nonleaf}, \quad (i,j) \in \Lambda, \end{cases} \tag{4}$$

where all the weights range from 0 to 1. The weight of expansion node connecting to leaf node relies on the weight of corresponding leaf node. $\widetilde{\mathbf{w}}$ and $\widetilde{\mathbf{A}}$ (Def. 8) shares the learnable parameter $\widehat{\mathbf{w}}$. Figure 2 shows an example to illustrate DST.

3.2 Training Oracle Graph Neural Network

This section constructs a differentiable surrogate model to capture the knowledge from any oracle function. We choose graph neural network architecture for its state-of-the-art performance in modeling structure-property relationships. In particular, we imitate the objective function F with GNN:

$$\widehat{y} = \text{GNN}(X;\Theta) \approx F(X;\mathcal{O}_1,\mathcal{O}_2,\cdots,\mathcal{O}_P) = y, \tag{5}$$

where Θ represents the GNN's parameters. Concretely, we use a graph convolutional network (GCN) (Kipf & Welling, 2016). Other GNN variants, such as Graph Attention Network (GAT) (Veličković et al., 2017), Graph Isomorphism Network (GIN) (Xu et al., 2018), can also be used in our setting. The initial node embeddings $\mathbf{H}^{(0)} = \mathbf{NE} \in \mathbb{R}^{K\times d}$ stacks basic embeddings of all the nodes in the scaffolding tree, d is the GCN hidden dimension, $\mathbf{N}$ is the node indicator matrix (Def. 4). $\mathbf{E} \in \mathbb{R}^{|\mathcal{S}|\times d}$ is the embedding matrix of all the substructures in vocabulary set $\mathcal{S}$, and is randomly initialized. The updating rule of GCN for the l-th layer is

$$\mathbf{H}^{(l)} = \text{RELU}\big(\mathbf{B}^{(l)} + \mathbf{A}(\mathbf{H}^{(l-1)}\mathbf{U}^{(l)})\big), \quad l = 1,\cdots,L, \tag{6}$$

where L is GCN's depth, $\mathbf{A}$ is the adjacency matrix (Def. 5), $\mathbf{H}^{(l)} \in \mathbb{R}^{K\times d}$ is the nodes' embedding of layer l, $\mathbf{B}^{(l)} = [\mathbf{b}^{(l)}, \mathbf{b}^{(l)}, \cdots, \mathbf{b}^{(l)}]^\top \in \mathbb{R}^{K\times d}$ and $\mathbf{U}^{(l)} \in \mathbb{R}^{d\times d}$ are bias and weight parameters of layer l, respectively.

We generalize the GNN from a discrete scaffolding tree to a differentiable one. Based on learnable weights for each node, we leverage the weighted average as the readout function of the last layer's (L-th) node embeddings, followed by multilayer perceptron (MLP) to yield the prediction $\widehat{y}$, i.e., $\widehat{y} = \text{MLP}\big(\frac{1}{\sum_{k=1}^K w_k} \sum_{k=1}^K w_k H_k^{(L)}\big)$, in discrete scaffolding tree, weights for all the nodes are equal to 1, $H_k^{(L)}$ is the k-th row of $H^{(L)}$. In sum, the prediction can be written as

$$\widehat{y} = \text{GNN}(X;\Theta) = \text{GNN}(\mathcal{T}_X = \{\mathbf{N},\mathbf{A},\mathbf{w}\};\Theta), \quad X \in \mathcal{Q} \tag{7}$$

where $\Theta = \{\mathbf{E}\} \cup \{\mathbf{B}^{(l)}, \mathbf{U}^{(l)}\}_{l=1}^L$ are the GNN's parameters. We train the GNN by minimizing the discrepancy between GNN prediction $\widehat{y}$ and the ground truth y.

$$\Theta_* = \arg\min_{\Theta} \sum_{(X,y)\in\mathcal{D}} \mathcal{L}\big(y = F(X;\mathcal{O}_1,\mathcal{O}_2,\cdots,\mathcal{O}_P),\ \widehat{y} = \text{GNN}(X;\Theta)\big), \tag{8}$$

where $\mathcal{L}$ is loss function, e.g., mean squared error; $\mathcal{D}$ is the training set. After training, we have GNN parameterized by Θ_* to approximate the black-box objective function F (Eq. 1). Worth to mention that Oracle GNN is trained once and for all. The training is separately from optimizing `DST` below.

3.3 OPTIMIZING DIFFERENTIABLE SCAFFOLDING TREE

Overview With a little abuse of notations, via introducing DST, we approximate molecule optimization as a *locally differentiable* problem

$$\underbrace{X = \arg\max_{X \in \mathcal{Q}} F(X)}_{\text{(I) structured combinatorial optimization}} \approx \underbrace{X^{(t+1)} = \arg\max_{X \in \mathcal{N}(X^{(t)})} F(X)}_{\text{(II) iterative local discrete search}}$$
$$\approx \underbrace{\mathcal{T}_{X^{(t+1)}} = \underset{X \in \mathcal{N}(X^{(t)})}{\arg\max} \ \text{GNN}(\mathcal{T}_X = \{\widetilde{\mathbf{N}}_{X^{(t)}}, \widetilde{\mathbf{A}}_{X^{(t)}}, \widetilde{\mathbf{w}}_{X^{(t)}}\}; \Theta_*)}_{\text{(III) local differentiable optimization}}, \tag{9}$$

where $X^{(t)}$ is the molecule at t-th iteration, $\mathcal{N}(X^{(t)}) \subseteq \mathcal{Q}$ is the neighborhood set of $X^{(t)}$ (Def. 10). Next, we explain the intuition behind these approximation steps. Molecular optimization is generally a discrete optimization task, which is prohibitively expensive due to exhaustive search. The first approximation is to formulate the problem as an iterative local discrete search via introducing a neighborhood molecule set $\mathcal{N}(X^{(t)})$. Second, to enable differentiable learning, we use GNN to imitate black-box objective F (Section 3.2) and further reformulated it into *a local differentiable optimization* problem. Then we can optimize DST ($\mathcal{T}_X = \{\widetilde{\mathbf{N}}_{X^{(t)}}, \widetilde{\mathbf{A}}_{X^{(t)}}, \widetilde{\mathbf{w}}_{X^{(t)}}\}$) in a continuous domain for $\mathcal{N}(X^{(t)})$ using gradient-based optimization method.

3.3.1 Local Editing Operations For a leaf node v in the scaffolding tree, we can perform three editing operations, (1) **SHRINK**: delete node v; (2) **REPLACE**: replace a new substructure over v; (3) **EXPAND**: add a new node u_v that connects to node v. For a nonleaf node v, we support (1) **EXPAND**: add a new node u_v connecting to v; (2) **do nothing**. If we EXPAND and REPLACE, the new substructures are sampled from the vocabulary $\mathcal{S}$. We define a molecule neighborhood set as below:

Definition 10 (Neighborhood set)**.** *Neighborhood set of molecule X, denoted $\mathcal{N}(X)$, is the set of all the possible molecules obtained by imposing one local editing operation to scaffolding tree $\mathcal{T}_X$ and assembling the edited trees into molecules.*

3.3.2 Optimizing DST Then within the domain of neighborhood molecule set $\mathcal{N}(X)$, the objective function can be represented as a differentiable function of X's DST ($\widetilde{\mathbf{N}}_X, \widetilde{\mathbf{A}}_X, \widetilde{\mathbf{w}}_X$). We address the following optimization problem to get the best scaffolding tree within $\mathcal{N}(X)$,

$$\widetilde{\mathbf{N}}_*, \widetilde{\mathbf{A}}_*, \widetilde{\mathbf{w}}_* = \arg\max_{\{\widetilde{\mathbf{N}}_X, \widetilde{\mathbf{A}}_X, \widetilde{\mathbf{w}}_X\}} \ \text{GNN}(\{\widetilde{\mathbf{N}}_X, \widetilde{\mathbf{A}}_X, \widetilde{\mathbf{w}}_X\}; \Theta_*), \tag{10}$$

where the GNN parameters Θ_* (Eq. (8)) are fixed. Comparing with Eq. (7), it is differentiable with regard to $\{\widetilde{\mathbf{N}}, \widetilde{\mathbf{A}}, \widetilde{\mathbf{w}}\}$ for all molecules in the neighborhood set $\mathcal{N}(X)$. Therefore, we can optimize the DST using gradient-based optimization method, e.g., an Adam optimizer (Kingma & Ba, 2014).

3.3.3 Sampling from DST Then we sample the new scaffolding tree from the optimized DST. Concretely, (i) for each leaf node $v \in \mathcal{V}_{\text{leaf}}$ and the corresponding expansion node $u_v \in \mathcal{V}_{\text{expand}}$, we select one of the following step with probabilities (w.p.) as follows,

$$\mathcal{T} \sim \text{DST-Sampler}(\widetilde{\mathbf{N}}_*, \widetilde{\mathbf{A}}_*, \widetilde{\mathbf{w}}_*)$$
$$= \begin{cases} \text{1. SHRINK: delete leaf node } v, & \text{w.p. } 1 - (\widetilde{\mathbf{w}}_*)_v, \\ \text{2. EXPAND: add } u_v\text{, select substructure at } u_v \text{ based on } (\widetilde{\mathbf{N}}_*)_{u_v}, & \text{w.p. } (\widetilde{\mathbf{w}}_*)_v (\widetilde{\mathbf{w}}_*)_{u_v|v}, \\ \text{3. REPLACE: select substructure at } v \text{ based on } (\widetilde{\mathbf{N}}_*)_u, & \text{w.p. } (\widetilde{\mathbf{w}}_*)_v (1 - (\widetilde{\mathbf{w}}_*)_{u_v|v}). \end{cases} \tag{11}$$

(ii) For each nonleaf node v, we expand a new node with probability $(\widetilde{\mathbf{w}}_*)_{u_v|v}$. If expanding, we select substructure at u_v based on $(\widetilde{\mathbf{N}}_*)_{u_v}$. Otherwise, we "**do nothing**".

3.3.4 Assemble Each scaffolding tree corresponds to multiple molecules due to the multiple ways substructures can be combined. We enumerate all the possible molecules following Jin et al. (2018) (See Section C.5 in Appendix for more details) for the further selection as described below.

3.4 MOLECULE DIVERSIFICATION

In the current iteration, we have generated M molecules $(X_1, \cdots, X_M)$ and need to select C molecules for the next iteration. We expect these molecules to have desirable chemical properties

(high F score) and simultaneously maintain higher structural diversity. To do so, we resort to the *determinantal point process (DPP)* (Kulesza & Taskar, 2012). DPP models the repulsive correlation between data points (Kulesza & Taskar, 2012) and has been successfully applied to many applications such as text summarization (Cho et al., 2019), mini-batch sampling (Zhang et al., 2017), and recommendation system (Chen et al., 2018). For M data points, whose indexes are $\{1, 2, \cdots, M\}$, $\mathbf{S} \in \mathbb{R}_+^{M \times M}$ denotes the similarity kernel matrix between these data points. To create a diverse subset (denoted $\mathcal{R}$) with fixed size C, the sampling probability should be proportional to the determinant of the submatrix $\mathbf{S}_{\mathcal{R}} \in \mathbb{R}^{C \times C}$, i.e., $P(\mathcal{R}) \propto \det(\boldsymbol{S}_{\mathcal{R}})$, where $\mathcal{R} \subseteq \{1, 2, \cdots, M\}$, $|\mathcal{R}| = C$. Combining the objective (F) value and diversity, the composite objective is

$$\underset{\mathcal{R} \subseteq \{1,2,\cdots,M\}, |\mathcal{R}|=C}{\arg\max} \mathcal{L}_{\text{DPP}}(\mathcal{R}) = \lambda \sum_{r \in \mathcal{R}} F(X_r) + \log P(\mathcal{R}) = \log\det(\boldsymbol{V}_{\mathcal{R}}) + \log\det(\boldsymbol{S}_{\mathcal{R}}), \quad (12)$$

where the hyperparamter $\lambda > 0$ balances the two terms, the diagonal scoring matrix $\boldsymbol{V} = \text{diag}\Big(\big[\exp(\lambda F(X_1)), \cdots, \exp(\lambda F(X_M))\big]\Big)$, $\boldsymbol{V}_{\mathcal{R}} \in \mathbb{R}^{C \times C}$ is a sub-matrix of $\mathbf{V}$ indexed by $\mathcal{R}$. When λ goes to infinity, it is equivalent to selecting C candidates with the highest F score regardless of diversity, same as conventional evolutionary learning in (Jensen, 2019; Nigam et al., 2020). Inspired by generalized DPP methods (Kulesza & Taskar, 2012; Chen et al., 2018), we further transform $\mathcal{L}_{\text{DPP}}(\mathcal{R})$,

$$\begin{aligned} \mathcal{L}_{\text{DPP}}(\mathcal{R}) &= \log\det(\boldsymbol{V}_{\mathcal{R}}) + \log\det(\boldsymbol{S}_{\mathcal{R}}) = \log\det(\boldsymbol{V}_{\mathcal{R}}\boldsymbol{S}_{\mathcal{R}}) \\ &= \log\det\left(\boldsymbol{V}_{\mathcal{R}}^{\frac{1}{2}}\boldsymbol{S}_{\mathcal{R}}\boldsymbol{V}_{\mathcal{R}}^{\frac{1}{2}}\right) = \log\det\left(\left(\boldsymbol{V}^{\frac{1}{2}}\mathbf{S}\boldsymbol{V}^{\frac{1}{2}}\right)_{\mathcal{R}}\right). \end{aligned} \quad (13)$$

where $\boldsymbol{V}^{\frac{1}{2}}\mathbf{S}\boldsymbol{V}^{\frac{1}{2}}$ is symmetric positive semi-definite. Then it can be solved by generalized DPP methods in $O(C^2M)$ (Chen et al., 2018) (Section F in Appendix). The computational complexity of `DST` is $O(TMC^2)$ (Section C.8 in Appendix). Algorithm 1 summarizes the entire algorithm.

Algorithm 1 Differentiable Scaffolding Tree (`DST`)

1: **Input**: Iteration T, population size C, input molecule $X^{(1)}$. Initial population $\Phi = \{X^{(1)}\}$.
2: **Output**: Generated Molecule Set Ω.
3: Learn GNN (Eq. 8): $\Theta_* = \arg\min_\Theta \sum_{(X,y)\in\mathcal{D}} \mathcal{L}(y, \widehat{y})$. # Section 3.2.
4: **for** $t = 1, 2, \cdots, T$ **do**
5: Initialize set $\Gamma = \{\}$.
6: **for** $X^{(t)} \in \Phi$ **do**
7: Initialize `DST` $\{\widetilde{\mathbf{N}}, \widetilde{\mathbf{A}}, \widetilde{\mathbf{w}}\}$ for $X^{(t)}$ (Eq. 2, 3, 4).
8: Optimize `DST`: $\mathbf{N}_*, \mathbf{A}_*, \mathbf{w}_* = \arg\max_{\{\widetilde{\mathbf{N}}, \widetilde{\mathbf{A}}, \widetilde{\mathbf{w}}\}}$ GNN$(\{\widetilde{\mathbf{N}}, \widetilde{\mathbf{A}}, \widetilde{\mathbf{w}}\}; \Theta_*)$ (Eq.10).
9: Sample from `DST`: $\mathcal{T}_j^{(t+1)} \sim$ DST-Sampler$\big(\mathbf{N}_*, \mathbf{A}_*, \mathbf{w}_*\big), j = 1, 2, \cdots$ (Eq.11); Assemble scaffolding tree $\mathcal{T}_j^{(t+1)}$ into molecules $X_j^{(t+1)}$.
10: $\Gamma = \Gamma \cup \{X_j^{(t+1)}\}$.
11: **end for**
12: Select $\Phi \subseteq \Gamma, |\Phi| = C$ based on Eq. 12; $\Omega = \Omega \cup \Phi$. # Section 3.4
13: **end for**

4 EXPERIMENT

In this section, we first describe the experimental setup in section 4.1 following Jin et al. (2018; 2019); You et al. (2018); Shi et al. (2020); Jin et al. (2020); Xie et al. (2021). Then we analyze the result from three aspects: optimization performance, oracle efficiency, and interpretability.

4.1 EXPERIMENTAL SETUP

Molecular Properties contains (1) **QED** (drug-likeness); (2) **LogP** (octanol-water partition coefficient); (3) **SA** (synthetic accessibility); (4) **JNK3** (c-Jun N-terminal Kinases-3); (5) **GSK3$\boldsymbol{\beta}$** (Glycogen synthase kinase 3 beta); For all the 5 scores (including normalized SA), higher is better. We conducted (1) single-objective generation that optimizes JNK3, GSK3$\boldsymbol{\beta}$ and LogP sepa-

rately and (2) multi-objective generation that optimizes the mean value of "JNK3+GSK3β" and "QED+SA+JNK3+GSK3β" in the main text. More details are given in Section C.3 in Appendix.

Dataset: ZINC 250K contains around 250K druglike molecules (Sterling & Irwin, 2015). We select the substructures that appear more than 1000 times in ZINC 250K as the vocabulary set $\mathcal{S}$, which contains 82 most frequent substructures. More details are in Section C.1 in Appendix.

Baselines. (1) **LigGPT** (string-based distribution learning model with Transformer as a decoder) (Bagal et al., 2021); (2) **GCPN** (Graph Convolutional Policy Network) (You et al., 2018); (3) **MolDQN** (Molecule Deep Q-Network) (Zhou et al., 2019); (4) **GA+D** (Genetic Algorithm with Discriminator network) (Nigam et al., 2020); (5) **MARS** (Markov Molecular Sampling) (Xie et al., 2021); (6) **RationaleRL** (Jin et al., 2020); Among them, LigGPT belongs to deep generative model, where all the oracle calls can be precomputed; GCPN and MolDQN are deep reinforcement learning methods; GA+D and MARS are evolutionary learning methods; RationaleRL is deep generative model fine-tuned with RL techniques. All the baselines except LigGPT require online oracle calls.

Metrics. (1) **Novelty**; (2) **Diversity**; (3) **SR** (Success Rate); (4) **# of oracle calls**: `DST` needs to call oracle in labeling data for GNN (**precomputed**) and `DST` based *de novo* generation (**online**), we show the costs for both steps. Details are provided in Section C.4.

Table 1: Multi-objective *de novo* design. #oracle = (1)"**precomputed** oracle call" (to label molecules in existing database) + (2)"**online** oracle call" (during learning).

Method	JNK3+GSK3β				QED+SA+JNK3+GSK3β			
	Nov↑	Div↑	SR↑	#oracle↓	Nov↑	Div↑	SR↑	#oracle↓
LigGPT	**100%**	**0.843**	0.0%	100k+0	100.0%	**0.906**	2.84%	100k+0
GCPN	**100.0%**	0.583	0.0%	0+200K	**100.0%**	0.590	12.7%	0+200K
MolDQN	**100.0%**	0.605	2.8%	0+200K	**100.0%**	0.599	1.0%	0+200K
GA+D	97.4%	0.660	48.0%	0+50K	96.4%	0.688	35.4%	0+50K
RationaleRL	93.6%	0.700	82.8%	25K+67K	94.0%	0.720	75.3%	25K+67K
MARS	97.9%	0.699	80.2%	0+50K	98.4%	0.714	73.4%	0+50K
`DST`	97.5%	0.745	**92.0%**	**10K+5K**	98.8%	0.774	**79.8%**	**20K+5K**

4.2 Optimization Performance

The results of multi-objective and single-objective generation are shown in Table 1 and 2. We find that DGM (LigGPT) and RL based methods (GCPN and MolDQN) fails in some tasks, which is consistent with the results reported in RationaleRL (Jin et al., 2020) and MARS (Xie et al., 2021). Overall, `DST` obtains the best results in most tasks. In terms of success rate and diversity, `DST` outperformed all baselines in most tasks. It also reached the highest scores within $T = 50$ iterations in most optimization tasks (see Table 5 and 6 in Appendix). Especially in optimizing LogP, the model successfully learned to add a six-member ring (see Figure 8 in Appendix) each step, which is theoretically the optimal strategy under our setting. Combined with the ablation study comparing with random selection (see Figure 11 in Appendix), our results show the local gradient defined by `DST` is a useful direction indicator, consistent with the concept of gradient. Further, achieving high diversity validates the effect of the DPP-based selection strategy. Although the novelty is not the highest, it is still comparable to baseline methods. These results show our gradient-based optimization strategy has a strong optimization ability to provide a diverse set of molecules with high objective functions.

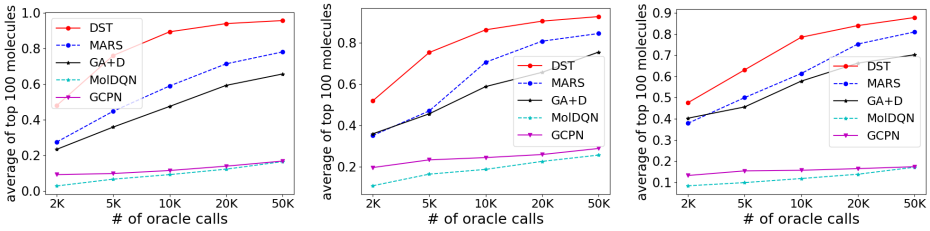
Table 2: Single-objective *de novo* molecular generation.

Method	JNK3				GSK3β				LogP			
	Nov↑	Div↑	SR↑	#oracle↓	Nov↑	Div↑	SR↑	#oracle↓	Nov↑	Div↑	SR↑	#oracle↓
LigGPT	**100.0%**	**0.853**	0.0%	100K+0	**100.0%**	**0.883**	0.0%	100K+0	**100.0%**	**0.856**	0.0%	100K+0
GCPN	**100.0%**	0.584	12.6%	0+200K	**100.0%**	0.519	14.5%	0+200K	**100.0%**	0.532	23.4%	0+200K
MolDQN	**100.0%**	0.603	23.0%	0+200K	**100.0%**	0.545	21.2%	0+200K	**100.0%**	0.470	20.5%	0+200K
GA+D	97.8%	0.681	70.3%	0+50K	95.2%	0.684	70.9%	0+50K	98.4%	0.701	96.4%	0+50K
RationaleRL	94.3%	0.682	88.7%	25K+32K	92.8%	0.732	85.2%	30K+45K	-	-	-	-
MARS	98.6%	0.709	89.3%	0+50K	97.6%	0.728	89.6%	0+50K	**100.0%**	0.689	**100.0%**	0+30K
`DST`	97.5%	0.732	**94.0%**	**10K+5K**	98.5%	0.748	**92.8%**	**10K+5K**	**100.0%**	0.704	**100.0%**	**10K+5K**

4.3 Oracle Efficiency

As mentioned above, oracle calls for realistic optimization tasks can be time-consuming and expensive. From Table 1 and 2, we can see that majority of de novo optimization methods require oracle calls online (instead of precomputation), including all of RL/evolutionary algorithm based baselines. `DST`

takes fewer oracle calls compared with baselines. DST can leverage the precomputed oracle calls to label the molecules in an existing database (i.e., ZINC) for training the oracle GNN and dramatically saving the oracle calls during reference. In the three tasks in Table 2, two-thirds of the oracle calls (10K) can be precomputed or collected from other sources. To further verify the oracle efficiency, we explore a special setting of molecule optimization where the budget of oracle calls is limited to a fixed number (2K, 5K, 10K, 20K, 50K) and compare the optimization performance. For GCPN, MolDQN, GA+D and MARS, the learning iteration number depends on the budget of oracle calls. RationaleRL (Jin et al., 2020) is not included because it requires intensive oracle calls to collect enough reference data, exceeding the oracle budget in this scenario. In DST, we use around 80% budget to label the dataset (i.e., training GNN) while the remaining budget to conduct *de novo* design. Specifically, for 2K, 5K, 10K, 20K, 50K, we use 1.5K, 4K, 8K, 16K and 40K oracle calls to label the data for learning GNN, respectively. We show the average objective values of top-100 molecules under different oracle budgets in Figure 3. Our method shows a significant advantage compared to all the baseline methods in all limited budget settings. We conclude the reason as supervised learning is a well-studied and much easier task than generative modeling.



(a) JNK3 (b) GSK3β (c) JNK3 + GSK3β

Figure 3: Oracle efficiency test. Top-100 average score v.s. number of oracle calls.

4.4 Interpretability Analysis

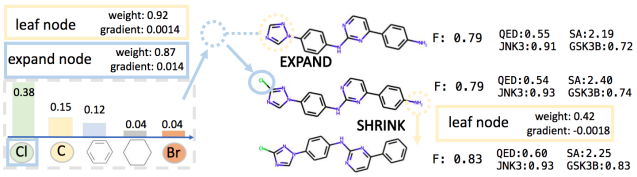


Figure 4: Two steps in optimizing "QED+SA+JNK3+GSK3β". Positive weight gradient on expand node means a positive contribution, while negative gradient represents negative one. Due to space limit, we only show the gradient of two expand nodes.

To obtain more insights from the local gradient, We visualize two modification steps in Figure 4. The node weights and their gradient values interpret the property at the substructure level. A positive gradient on expanding node means a positive contribution to the objective value increment, while a negative gradient represents negative one (at least locally). This is similar to most attribution-based interpretable machine learning methods, e.g., saliency map (Simonyan et al., 2013).

5 Conclusion

This paper proposed differentiable scaffolding tree (DST) to make a molecular graph locally differentiable, allowing a continuous gradient-based optimization. To the best of our knowledge, it is the first attempt to make the molecular optimization problem differentiable at the substructure level, rather than resorting to latent spaces or using RL/evolutionary algorithms. We constructed a general molecular optimization strategy based on DST, corroborated by thorough empirical studies.

REPRODUCIBILITY STATEMENT

The code repository is given in the supplementary material, including README file, all the codes for data preprocessing and model. All the data we use are publicly available. Section B describes experimental setup for all the baseline methods. We elaborate the implementation details in Section C. Concretely, Section C.1 describes the dataset we use; Section C.2 describes software and hardware configuration; Section C.3 introduces the target molecular properties to optimize; Section C.4 describes all the metrics for evaluating the performance; Section C.6 describes the setup for model implementation, especially hyperparameter setup.

REFERENCES

Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. LigGPT: Molecular generation using a transformer-decoder model. 2021.

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, 2012.

Rasmus Bro and Age K Smilde. Principal component analysis. *Analytical methods*, 6(9):2812–2831, 2014.

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.

Binghong Chen, Tianzhe Wang, Chengtao Li, Hanjun Dai, and Le Song. Molecule optimization by explainable evolution. In *International Conference on Learning Representations*, 2021.

Laming Chen, Guoxin Zhang, and Hanning Zhou. Fast greedy map inference for determinantal point process to improve recommendation diversity. In *Neural Information Processing Systems*, pp. 5627–5638, 2018.

Sangwoo Cho, Logan Lebanoff, Hassan Foroosh, and Fei Liu. Improving the similarity measure of determinantal point processes for extractive multi-document summarization. In *Association for Computational Linguistics, ACL*, 2019.

Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. In *ICLR*, 2018.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

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):8, 2009.

Tianfan Fu, Cao Xiao, Xinhao Li, Lucas M Glass, and Jimeng Sun. MIMOSA: Multi-constraint molecule sampling for molecule optimization. *arXiv preprint arXiv:2010.02318*, 2020.

Wenhao Gao and Connor W Coley. The synthesizability of molecules proposed by generative models. *Journal of chemical information and modeling*, 60(12):5714–5723, 2020.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 2018.

Sai Krishna Gottipati, Boris Sattarov, Sufeng Niu, Yashaswi Pathak, Haoran Wei, Shengchao Liu, Simon Blackburn, Karam Thomas, Connor Coley, Jian Tang, et al. Learning to navigate the synthetically accessible chemical space using reinforcement learning. In *International Conference on Machine Learning*, pp. 3668–3679. PMLR, 2020.

Johannes Hachmann, Roberto Olivares-Amaya, Sule Atahan-Evrenk, Carlos Amador-Bedolla, Roel S Sánchez-Carrera, Aryeh Gold-Parker, Leslie Vogt, Anna M Brockway, and Alán Aspuru-Guzik. The harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters*, 2(17): 2241–2251, 2011.

Shion Honda, Hirotaka Akita, Katsuhiko Ishiguro, Toshiki Nakanishi, and Kenta Oono. Graph residual flow for molecular graph generation. *arXiv preprint arXiv:1909.13521*, 2019.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. Therapeutics data commons: machine learning datasets and tasks for therapeutics. *arXiv preprint arXiv:2102.09548*, 2021.

Jon Paul Janet, Sahasrajit Ramesh, Chenru Duan, and Heather J Kulik. Accurate multiobjective design in a space of millions of transition metal complexes with neural-network-driven efficient global optimization. *ACS central science*, 6(4):513–524, 2020.

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.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *ICML*, 2018.

Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *ICLR*, 2019.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International Conference on Machine Learning*, pp. 4849–4859. PMLR, 2020.

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

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3393–3403. PMLR, 2020.

Panagiotis-Christos Kotsias, Josep Arús-Pous, Hongming Chen, Ola Engkvist, Christian Tyrchan, and Esben Jannik Bjerrum. Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks. *Nature Machine Intelligence*, 2(5):254–265, 2020.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*, 2012.

Irwin D Kuntz. Structure-based strategies for drug design and discovery. *Science*, 257(5073): 1078–1082, 1992.

Greg Landrum et al. RDKit: Open-source cheminformatics, 2006.

Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 2018.

Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. Graphebm: Molecular graph generation with energy-based models. *arXiv preprint arXiv:2102.00546*, 2021.

Mengqiu Long, Ling Tang, Dong Wang, Yuliang Li, and Zhigang Shuai. Electronic structure and carrier mobility in graphdiyne sheet and nanoribbons: theoretical predictions. *ACS nano*, 5(4): 2593–2600, 2011.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alán Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. In *ICLR*, 2020.

Nikolaos V Sahinidis and Mohit Tawarmalani. Applications of global optimization to process and molecular design. *Computers & Chemical Engineering*, 24(9-10):2157–2169, 2000.

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.

Cynthia Shen, Mario Krenn, Sagi Eppel, and Alan Aspuru-Guzik. Deep molecular dreaming: Inverse machine learning for de-novo molecular design and interpretability with surjective representations. *Machine Learning: Science and Technology*, 2021.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. GraphAF: a flow-based autoregressive model for molecular graph generation. In *ICLR*, 2020.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Manish Sinha, Luke EK Achenie, and Gennadi M Ostrovsky. Environmentally benign solvent design by global optimization. *Computers & Chemical Engineering*, 23(10):1381–1394, 1999.

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

Paraskevi Supsana, Theodoros Liaskopoulos, Stavroula Skoulika, Antonios Kolocouris, Petros G Tsoungas, and George Varvounis. Thermal rearrangement of spiro [naphthalene (naphthopyranofurazan)] oxides to spiro [naphthalene (phenalenofurazan) oxides. a probable furazan oxide triggered tandem isomerisation process. *Tetrahedron*, 61(25):6131–6137, 2005.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Yanli Wang, Stephen H Bryant, Tiejun Cheng, Jiyao Wang, Asta Gindulyte, Benjamin A Shoemaker, Paul A Thiessen, Siqian He, and Jian Zhang. Pubchem bioassay: 2017 update. *Nucleic acids research*, 45(D1):D955–D963, 2017.

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.

Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. MARS: Markov molecular sampling for multi-objective drug discovery. In *ICLR*, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *NIPS*, 2018.

Chengxi Zang and Fei Wang. MoFlow: an invertible flow model for generating molecular graphs. In *ACM SIGKDD*, pp. 617–626, 2020.

Cheng Zhang, Hedvig Kjellstrom, and Stephan Mandt. Determinantal point processes for mini-batch diversification. *Uncertainty in Artificial Intelligence (UAI)*, 2017.

Alex Zhavoronkov, Yan A Ivanenkov, Alex Aliper, Mark S Veselov, Vladimir A Aladinskiy, Anastasiya V Aladinskaya, Victor A Terentiev, Daniil A Polykovskiy, Maksim D Kuznetsov, Arip Asadulaev, et al. Deep learning enables rapid identification of potent ddr1 kinase inhibitors. *Nature biotechnology*, 37(9):1038–1040, 2019.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 2019.

Julie B Zimmerman, Paul T Anastas, Hanno C Erythropel, and Walter Leitner. Designing for a green chemistry future. *Science*, 367(6476):397–400, 2020.