Tree-Invent: A novel molecular generative model constrained with topological tree

Mingyuan Xu¹, Hongming Chen¹,*

¹ Guangzhou Laboratory, No. 9 XingDaoHuanBei Road, Guangzhou International Bio Island, Guangzhou 510005, Guangdong Province, China

*Correspondence e-mail: chen_hongming@gzlab.ac.cn

Abstract:

*De novo* molecular design plays an important role in drug discovery. Here a novel generative model, Tree-Invent, was proposed to integrate topological constraints in the generation of molecular graph. In this model, a molecular graph is represented as a topological tree in which ring system, non-ring atom and chemical bond are regarded as ring node, single node and edge respectively. The molecule generation is driven by three independent sub-models for carrying out operations of node addition, ring generation and node connection. One unique feature of the generative model is that topological tree structure can be specified as constraint for structure generation, which provides more precise control on structure generation. Combining with reinforcement learning, Tree-Invent model could efficiently explore targeted chemical space. Moreover, Tree-Invent model is flexible enough to be used in versatile molecule design settings such as scaffold decoration, scaffold hopping and linker generation.

Introduction

Drug discovery is a notorious lengthy and costly process which is often described as finding a needle in a hay-stack. In this case, the haystack comprises on the order of $10^{60}$-$10^{100}$ synthetically feasible molecules¹, of which we need to find a perfect compound which satisfies a plethora of criteria including bioactivity, drug metabolism and pharmacokinetic (DMPK) profile, synthetic accessibility, etc. *De novo* molecular design has been proposed as a way to accelerate the drug discovery process. Although successes have been achieved for various *de novo* molecular design methods², efficiently exploring drug-like chemical space is still quite challenging owing to the discrete nature of its representation and the gigantic scale³.
Recently, deep generative models have been proposed as a novel way for doing de novo molecular design\(^2\) and its core idea is to learn the underlying distribution of structure data, allowing the generation of novel drug-like molecules from a learned distribution. Various neural network architectures, including recurrent neural networks (RNN)\(^{4-6}\), variational auto-encoders\(^7,8\), generative adversarial networks\(^9\) and transformer etc\(^{10}\), have been explored to build molecular generative models. So far SMILES and 2D molecular graph are the most frequently used representations for generative models. Additionally, SELFIES\(^11\), a special form of string-based representation, was proposed for building generative model. Generative models can also be classified based on the way of structure generation, one type is the so-called one-shot generation, in which molecular structures are generated/translated directly from an embedding vector in one step and it is typically used in VAE based models. The other type of model is the so-called auto-regressive model, where structure generation is a multi-step process and only part of structure is generated in each step according to the predicted probability distribution of actions. Comparing with one-shot model, auto-regressive generative model provides larger flexibility for structure generation.

Among auto-regressive models, no matter SMILES or graph-based models, employing atom-based action space is the most common way for structure generation, in which a structure grows in an atom-by-atom manner, i.e. only one atom is added to the structure in each step.\(^2\) Alternatively, models using fragment-based action space were also proposed, in which functional groups were used as building blocks for structure growing. For example, Jin et al proposed a novel coarse-grained molecule representation by leveraging junction tree (JT) decomposition to a molecule graph;\(^{12}\) Ishitani et al proposed an RJT-RL methods based on a reversible junction tree representation;\(^3\) Poelking et al proposed a hierarchical model LIBPQR with multi-level self-contrastive learning to improve bias control and data efficiency.\(^{13}\) Among these methods, a set of primary chemical group is pre-defined to form the action space, this type of model can, at some extent, improve the structure validity, while, as a drawback, restraining the search space of generative model.

A challenging task deep generative models are trying to address is inverse molecular design, i.e. not generating random structures but the structures that meet desired conditions, such as specific physiochemical properties or properties predicted by quantitative structure–
activity relationship (QSAR) models. In order to guide the generative model in exploring specific chemical space, Bayesian optimization, Monte Carlo Tree Search or meta-heuristics based methods can be used to optimize molecular properties during molecular generation process in case such properties are not differentiable. Reinforcement learning (RL) is another way to optimize structure generation with regard to certain molecular properties, in which the molecule generation problem is formulated as a Markov decision process wherein an agent (neural network) learns the optimal policy based on the rewards offered by its surrounding environments. The RL has been widely used in both SMILES-based and graph-based generative models, for example, the REINVENT, GCPN, MolDQN, RL-Graph-INVENT models etc.

Here, we proposed a novel auto-regressive generative model, Tree-Invent, by merging the fragment based topological tree generation and atom-based graph generation in one model. In this model, molecular structure is represented in two hierarchical layers, one layer is the topological tree in which ring system, non-ring atom and chemical bond are abstracted as ring node, single atom node and edge respectively; The other layer is the normal molecular graph representing full information of a molecule. Thus, the structure generation process comprises two steps, in which a topological tree is firstly generated following by the generation of full molecular graph. The generative model contains three independent modules for operating node addition, ring formation and node connection respectively. A unique mask mechanism is designed to make sure that generated molecules fulfill certain structure constraints. Different from JT methods and LibPQR which utilize topological tree to facilitate the encoding of the whole structure, Tree-Invent integrates topological tree into auto-regressive structure generation process. The way of employing topological structural constraint in Tree-Invent not only imposes precise control of structure generation but also provides flexibility for dealing with diverse structure generation settings. Currently generative models employing structure constraints are highly task specific and can’t be used interchangeably, for example generative models for doing scaffold decoration and the ones for doing linker generation usually use different neural network structures and it is difficult to solve both tasks with the same type of model. While in current study, several examples were given to demonstrate that Tree-invent model can not only efficiently generate diverse chemical structures, but also carry out multiple
structure constrained generation tasks such as scaffold decoration, scaffold hopping and linker design.

Methods and materials
A. The basic concept of Tree-Invent

In Tree-Invent model, a molecular graph is represented as a multi-forked topology tree by simplifying the ring systems in the structure as coarse-grained ring nodes, as shown in Figure 1a, while other non-ring atoms represented as single atom nodes. Thus, the molecular generation becomes a process of tree generation following by ring generation to form a full molecule graph. In our model, any ring system is designated with a node topology fingerprint (NTF, as shown in Figure 1b), which describe the number of ring in the smallest set of smallest ring (SSSR) search\(^2\), the number of aromatic ring, appearance of exocyclic double bond, and number of C, N, O, F, P, S, Cl, Br, I atom. So, the ring node of the topology tree should be regarded as a fuzzy ring which represents a class of ring system instead of a deterministic ring structure. Once a ring node is generated, its exact ring structure will be instantiated by a separated ring formation module. By doing structure generation in this way, the validity of ring structures can be improved.
**Figure 1.** (a) The topological tree for representing molecule, (b) The composition of NTF for a ring node and (c) a single atom node. The $\oplus$ refers to concatenation.

The structure generation in Tree-Invent is done by repetitively carrying out five steps as following:

1. **Initialization.** Tree-Invent starts from an empty graph and a pre-defined tree structure constraint. For unconstrained molecule generation, the tree constraint will then be skipped.

2. **Node sampling.** The $Node_{add}$ model is trained to sample tree nodes given current molecular graph, either single atom node or ring node. Once a ring node is sampled, its NTF is used as control information for the $Ring_{gen}$ model to generate ring.

3. **Generation of ring.** The $Ring_{gen}$ model takes the NTF of the ring node, current molecular graph, and current sub-graph of the ring node to update the ring structure until the ring formation is finished.

4. **Node connection.** When ring formation of the ring node is finished or a single atom node is sampled in the node sampling step, the $Node_{conn}$ model connects the sub-graph of the sampled node with current molecular graph, under the condition of the pre-defined tree constraints, and then update current molecular graph.

5. **Termination.** If the $Node_{add}$ model predicts that the updated molecular graph should be terminated or the total atom number exceeds the predefined $V_{max}^M$, the molecular graph is then terminated. Otherwise, steps (2) ~ (4) are repeated.
Figure 2. The basic molecular generation workflow of Tree-Invent

**B. The architecture of Tree-Invent model**

Tree-Invent consists of three main modules, i.e., $Node_{add}$, $Ring_{gen}$, and $Node_{conn}$ module as discussed above, for structure generation. Their architectures are shown in Figure 3.
Figure 3. Model architectures of (a) $Node_{add}$, (b) $Ring_{gen}$, and (c) $Node_{conn}$. The $\oplus$ refers to concatenation.

**Graph neural networks (GNN) blocks in Tree-Invent**

Among all three sub-models, graph convolution network (GCN) block is used as the base unit to learn representation of graph structure. A given molecular graph or sub-graph of ring node can be expressed as a graph $G = (V, E)$ including the features for all nodes $x_v$, $\forall v \in V$ and features for all edges $x_{e_{vw}}$, $\forall e_{vw} \in E$. For a given node $v$, its initial node feature $H^0_v$ is a one-hot vector encoding its atomic number and formal charge. For the bond $e_{vw}$ connecting
node \( v \) and \( w \), its initial bond/edge feature \( H^0_{e_{v,w}} \) encodes the bond type. The dimension and composition of the initial embedding \( H^0 \) and \( H^0_{e_{v,w}} \) is shown in Table 1.

### Table 1. The composition of node feature \( x_v \) and edge feature \( x_{e_{v,w}} \)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Set label</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node feature ( x_v )</td>
<td>Atom type</td>
<td>( \mathcal{A} ) {C,N,O,F,P,S,Cl,Br,I}</td>
</tr>
<tr>
<td></td>
<td>Formal Charge</td>
<td>( \mathcal{F} ) {-2,-1,0,1,2}</td>
</tr>
<tr>
<td>Bond feature ( x_{e_{v,w}} )</td>
<td>Bond type</td>
<td>( \mathcal{B} ) {Single, Double, Triple}</td>
</tr>
</tbody>
</table>

Node embedding \( H_i^{l+1} \) after \( l + 1 \)th message passing is calculated by the so-called gated graph neural network (GGNN) as following:

\[
m_i^{l+1} = \sum_{v_j \in N(v_i)} MLP^e(H_j^l)e_{ij} \quad \forall l \in L
\]

\[
H_i^{l+1} = GRU(m_i^{l+1}, H_i^l) \quad \forall l \in L
\]

Where the \( MLP^e \) represents the multi-layer perceptron given edge type \( e \), the \( GRU \) represents the gated recurrent unit\(^{22} \), and \( N(v_i) \) represents the set of node bonded to node \( v_i \).

The graph readout function in GGNN is expressed in Eq (3).

\[
H_G = \sum_{v_j \in N(v_i)} \sigma \left( MLP^a(H_j^L) \otimes \tanh \left( MLP^b([H_i^L, H_i^0]) \right) \right)
\]

Where \( \otimes \) represents the element-wise multiplication (Hadamard product), \([H_i^L, H_i^0]\) refers to the concatenation of \( H_i^L \) and \( H_i^0 \).

**Node addition module \( Node_{add} \)**

\( Node_{add} \) module is served to predict the next action given current graph, i.e. if the graph should stop growing and, if not, what next node should be selected for the graph. The action probability distribution (APD) for current graph can be calculated through Eq (4) ~ (6). Thus, its action space is split into two parts: \( T_{Node} \), which predicts the type of next node appended to the current molecular graph, and \( T_{End} \), which predicts if the graph generation process should be terminated or not. The dimension of \( T_{Node} \) is the summation of the number of atom types \( N_{at} \) and ring system classes \( N_{rt} \), while \( T_{End} \) is a binary variable representing TRUE (“1”) or FALSE (“0”) for termination of molecular graph \( G_M \).
In $Node_{add}$, two MLPs, $MLP^{T_{Node}}$ and $MLP^{T_{End}}$, take the embedding of the molecular graph $H_{GM}$ as input and predict APDs of $T_{Node}$ and $T_{End}$ respectively. Finally, $Node_{add}$ outputs the action probability distribution $APD_{add}$ by using SoftMax function given the concatenation of $T_{Node}$ and $T_{End}$.

$$T_{Node} = MLP^{T_{Node}}(H_{GM})$$

$$T_{End} = MLP^{T_{End}}(H_{GM})$$

$$APD_{add} = SoftMax([T_{Node}, T_{End}])$$

**Ring generation module $Ring_{gen}$**

As shown in Figure 2, once $Node_{add}$ samples a ring node corresponding to a specific NTF, the generative model $Ring_{gen}$ is then employed to generate ring structure under the constraint of NTF. The action space of $Ring_{gen}$ is divided into four parts: 1) *Initialization action*, which adds a single atom to the empty graph of the ring; 2) *Appending action*, which adds a new atom into the current sub-graph and connects it to an existing atom in the graph; 3) *Connecting action*, which connects the last appended atom to another atom in the graph; 4) *Termination action*, which ends the ring generation process.

$Ring_{gen}$ module contains two GGNN blocks for gathering the graph representation of molecular graph $G_{M}$ and sub-graph $G_{R}$, respectively, as shown in Figure 3 (b). It generates ring structure by taking current sub-graph of the ring as well as the graph of whole molecule as input. $R_{add}$ refers to the action distribution of *initiation* and *appending*, consisting of atom type, formal charge, index of the existing connected atom and bond type for the created bond, and is predicted by two MLP neural networks, $MLP_{1}^{R_{add}}$ and $MLP_{2}^{R_{add}}$. The former network transforms the node embeddings $H_{L}^{R}$ of sub-graph to $h_{i}^{L,R}$, while the latter one takes the concatenation of the set $[h_{1}^{L,R} \sim h_{N}^{L,R}]$, the graph embedding $H_{GM}$ of the molecular graph $G_{M}$, and the NTF of ring node as input and predicts the $R_{add}$ as shown in Eq (7)–(8).

$$h_{i}^{R_{add}} = MLP_{1}^{R_{add}}(H_{L}^{R})$$

$$R_{add} = MLP_{2}^{R_{add}}([h_{1}^{L,R}, \ldots h_{N}^{L,R}, H_{GM}, RTF])$$
Similarly, \( R_{\text{conn}} \) represents the action distribution of connecting, consisting of the index of the existing connected atom and the bond type of the connecting bond. \( R_{\text{conn}} \) is predicted by \( MLP_{R_{\text{conn}}}^1 \) and \( MLP_{R_{\text{conn}}}^2 \) in the same way as \( R_{\text{add}} \) as expressed in Eq (9) ~ (10).

\[
h_{i_{\text{conn}}} = MLP_{R_{\text{conn}}}^1(h_{i_{\text{conn}}}^{L,R}) \tag{9}
\]

\[
R_{\text{conn}} = MLP_{R_{\text{conn}}}^2([h_{i_{\text{conn}}}^{L,R}, \ldots h_{N_{\text{conn}}}^{L,R}, H_{G_M}, RTF]) \tag{10}
\]

A binary variable \( R_{\text{End}} \) represents the probability of terminating the ring generation. It is predicted by \( MLP_{R_{\text{End}}}^0 \) with the graph embedding \( H_{G_R} \) expressed in Eq (11) as input. The final action probability distribution \( APD_{\text{Ring}} \) for ring construction is expressed as the concatenation of \( R_{\text{add}}, R_{\text{conn}}, \) and \( R_{\text{End}} \).

\[
R_{\text{End}} = MLP_{R_{\text{End}}}^0(H_{G_R}) \tag{11}
\]

\[
APD_{\text{Ring}} = \text{SoftMax}([R_{\text{add}}, R_{\text{conn}}, R_{\text{End}}]) \tag{12}
\]

**Node connection module Node\(_{\text{conn}}\)**

Here, the \( \text{Node}_{\text{conn}} \) module is responsible for connecting the sampled node with molecular graph according to the predicted \( APD_{\text{conn}} \), which consists of distribution probability of bond type of the created bond and index of anchoring atom in the molecular graph. Similar to \( \text{Ring}_{\text{gen}} \), there are two GGNN blocks in \( \text{Node}_{\text{conn}} \) for learning embeddings of the sampled node \( G_s \) and the molecule graph \( G_M \). If the sampled node is a ring node, the node \( G_s \) is a subgraph, otherwise \( G_s \) is a single atom node. Two MLPs, i.e. \( MLP_{T_{\text{conn}}}^1 \) and \( MLP_{T_{\text{conn}}}^2 \), are used for predicting \( APD_{\text{conn}} \). As shown in Eq (13) ~ (15), \( MLP_{T_{\text{conn}}}^1 \) shapes the node embedding of molecule graph \( H_{G_M} \) to \( h_{\text{conn}} \). \( MLP_{T_{\text{conn}}}^2 \) takes the concatenation of the set of embedding \( [h_{i_{\text{conn}}}^{\text{conn}} \sim h_{N_{\text{conn}}}^{\text{conn}}] \), the node embedding of anchoring atom of sub-graph \( H_a \), the graph embedding of sub-graph \( H_{G_s} \) and molecular graph \( H_{G_M} \) to predict the \( APD_{\text{conn}} \). In case the sampled node is a single atom node, the embedding \( H_{G_s} \), \( H_a \) are the same.

\[
h_{i_{\text{conn}}} = MLP_{T_{\text{conn}}}^1(h_{i_{\text{conn}}}^{L,M}) \tag{13}
\]

\[
APD_{\text{conn}} = \text{SoftMax} \left( MLP_{T_{\text{conn}}}^2([h_{i_{\text{conn}}}^{\text{conn}}, \ldots h_{N_{\text{conn}}}^{\text{conn}}, H_a, H_{G_M}, H_{G_s}]) \right) \tag{14}
\]

**The training process of Tree-Invent**
For training Tree-Invent, every molecular graph of training set is segmented into a sequence of action according to specific atom traversing rules to prepare training data (as exemplified in Figure 4). As discussed in previous section, molecule graph is expressed as a multi-fork topological tree, traversing rule on the tree and ring structure should be considered separately.

Figure 4. Segmentation of an example structure for preparing training data. The left part in each box is the segmented fragment serving as input for neural network and the right part is the output. The bond with dotted line is the bond supposed to be created.

To save the memory of both CPU and GPU, the compound segmentation in training set was done on-the-fly instead of generated and stored in hard disk in advance. Three models \( \text{Node}_{\text{add}}, \text{Ring}_{\text{gen}}, \) and \( \text{Node}_{\text{conn}} \) were trained separately, their loss functions were set as the negative likelihood loss value as shown in Eq (15) – (17).

\[
\text{Loss}_{\text{add}} = \text{NLL}\left(\text{APD}_{\text{predicted}}^{\text{add}}, \text{APD}_{\text{add}}^{\text{reference}}\right) \tag{15}
\]

\[
\text{Loss}_{\text{ring}} = \text{NLL}\left(\text{APD}_{\text{predicted}}^{\text{ring}}, \text{APD}_{\text{ring}}^{\text{reference}}\right) \tag{16}
\]

\[
\text{Loss}_{\text{conn}} = \text{NLL}\left(\text{APD}_{\text{predicted}}^{\text{conn}}, \text{APD}_{\text{conn}}^{\text{reference}}\right) \tag{17}
\]
Once training was done, Node\textsubscript{add}, Ring\textsubscript{gen} and Node\textsubscript{conn} models were saved separately, they were assembled in the workflow as shown in Figure 2 for structure generation.

C. Unconstrained molecular generation

In the generation mode of Tree-Invent, actions are sequentially sampled from probability distribution tensor $APD_{add}, APD_{ring}, \text{and } APD_{conn}$. The shape and composition of APDs are shown in Table 2.

**Table 2.** $APD_{add}, APD_{ring}, \text{and } APD_{conn}$ shape tensor for a single step.

<table>
<thead>
<tr>
<th>APDs</th>
<th>Composition</th>
<th>Dim</th>
<th>Property</th>
<th>Size</th>
<th>Elements Corresponding to index</th>
</tr>
</thead>
<tbody>
<tr>
<td>$APD_{add}$</td>
<td>$T_{node}$</td>
<td>0</td>
<td>Node types $A^M$</td>
<td>$</td>
<td>A^M</td>
</tr>
<tr>
<td></td>
<td>$T_{end}$</td>
<td>0</td>
<td>termination of molecular graph $E^M$</td>
<td>1</td>
<td>$(0,1)$</td>
</tr>
<tr>
<td>$R_{add}$</td>
<td></td>
<td>0</td>
<td>Node $V^R_{add}$ to connect in $V^R$</td>
<td>$</td>
<td>V^R_{max}</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>New atom type $A^R$</td>
<td>$</td>
<td>A^R</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>New formal charge $F$</td>
<td>$</td>
<td>F</td>
<td>$</td>
</tr>
<tr>
<td>$APD_{ring}$</td>
<td></td>
<td>3</td>
<td>New bond type $B^R_{add}$</td>
<td>$</td>
<td>B</td>
</tr>
<tr>
<td>$R_{conn}$</td>
<td></td>
<td>0</td>
<td>Node $V^R_{conn}$ to connect in $V^R$</td>
<td>$</td>
<td>V^R_{max}</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>New bond type $B^R_{conn}$</td>
<td>$</td>
<td>B</td>
<td>$</td>
</tr>
<tr>
<td>$R_{end}$</td>
<td></td>
<td>0</td>
<td>termination of sub-graph $E^R$</td>
<td>1</td>
<td>$(0,1)$</td>
</tr>
<tr>
<td>$APD_{conn}$</td>
<td>/</td>
<td>0</td>
<td>Node to connect in $V^M$</td>
<td>$</td>
<td>V^M_{max}</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>New bond type $B$</td>
<td>$</td>
<td>B</td>
<td>$</td>
</tr>
</tbody>
</table>

In the node sampling step, node types $A^M$, and termination of molecule graph $E$ for the node addition action are sampled according to Eq (18):

$$A^M, E^M = Sample_{add}\left(\exp\left(\frac{\log(APD_{add})}{T}\right)\right)$$  \hspace{1cm} (18)

Where $T$ is the temperature for controlling the sampling diversity. $Sample_{add}$ refers to routines sampling actions under given APD and the detail can be found in Routine S1 of supporting materials.

In the ring growth step, source atom $V^R_{add}$ of the ring sub-graph, the atom type $A^R$, the formal charge $F$, and the bond type $B^R_{add}$ of the new atom (as target atom), the existing atom (as target atom) of ring sub-graph $V^R_{conn}$, the bond type $B^R_{conn}$ for connecting the existing atom, and termination of ring $E^R$ are sampled from $APD_{ring}$ individually as shown in Eq (19), where $Sample_{ring}$ is shown in Routine S2:

$$V^R_{add}, A^R, F, B_{add}, V^R_{conn}, B^R_{conn}, E^R = Sample_{ring}(APD_{ring})$$  \hspace{1cm} (19)
It is worthwhile to mention that, in Tree-Invent, ring structures are generated by the \( R_i g \) model and are not hard-coded as those junction tree-based methods do\(^\text{12}\). By doing in this way, defining a cost inefficient large vocabulary for rings can be avoided.

In the node connection step, the new node is connected to the molecule graph by sampling a connecting atom \( V_{\text{conn}}^M \) and bond type \( B \) as shown in Eq (20), where \( \text{Sample}_{\text{conn}} \) is defined in Routine S3.

\[
V_{\text{conn}}^M, B = \text{Sample}_{\text{conn}}(ADP_{\text{conn}})
\]  

(20)

**D. Molecular generation with topological constraints**

One unique feature of Tree-Invent is its capability of incorporating topological features as constraints for structure generation. By topology feature, we mean certain number of atom with specific connectivity pattern existed in the molecular graph or part of the molecular graph, for example some topology features are defined in Figure 6. Among those topological features, the connection between nodes and even the type of connection can be precisely specified. In order to generate structures fulfilling these topological features, during the course of structure generation, a novel masking scheme is introduced in certain steps for restraining the ADP of node sampling and node connection. Here several types of constraint are introduced to highlight that Tree-Invent is flexible enough to generate structures under various settings.

Given a molecular graph \( G_M^i \) and its corresponding topological graph \( T_M^i \) (as shown in Figure 5a), in case a ring node will be sampled, the node type constrains for the next \textit{Node add} action could be defined by specifying the maximum and minimum number of SSSR \((N_R^{\text{min}}, N_R^{\text{max}})\), aromatic ring \((N_{\text{aro}}^{\text{min}}, N_{\text{aro}}^{\text{max}})\), exocyclic double bond \((N_{\text{br}}^{\text{min}}, N_{\text{br}}^{\text{max}})\), and atoms for specified element type \((A: N_A^{\text{min}}, \ldots), (A: N_A^{\text{max}}, \ldots)\), subsequently a mask \( M_{\text{add}}^i \) is then constructed in which bits for suitable node types are set to “1” and applied in the \textit{ith node addition} step as shown in Eq (21).

\[
A^{M,i}, B^{M,i} = \text{sample}_{\text{add}} \left( \exp \left( \frac{\log(ADP_{\text{add}}^i)}{T} \right) * M_{\text{add}}^i \right)
\]  

(21)

As shown in Figure 5b, we would like to constrain the 3th node as a single aromatic ring containing 2 carbon and 3 nitrogen atoms without branches, which corresponds to the node of type 18th with topological fingerprint of "\( R0 - C2 - N3 - O0 - F0 - P0 - S0 - CI0 - Br0 - I0 - AR1 - BR0 "\). By applying the mask \( M_{\text{add}}^3 \) in the sampling step, node type 18th
will get a higher probability being sampled and the $R_{gen}$ would then intend to generate four different rings shown in Figure 6b, corresponding to the specified topological fingerprints.

In the node connection step, we could also constrain the attachment position of current molecular graph for connecting the new node and the bond type for the connection, i.e. the node id in topology tree $Tree_{conn}$ and the bond type $B_{conn}$, and possible element type $A_{conn}$ of the existing atom to be connected. Similarly, these connection conditions could also be transformed to the mask $M^i_{conn}$ in the $i$th node connection step as shown in Eq (22) and Figure 5c. There are five possible connect positions when the new node is constrained for connecting to the node 1 of the topology tree.

$$V^{M,i}_{conn}, B^{i}_{conn} = sample_{conn}(APD^{i}_{conn} * M^{i}_{conn})$$

**Figure 5.** (a) The topological graph and molecular graph of fluorobenzene. (b) The node type constraints of the next node addition step. (c) The connection constraints of the next node connection step. (d) The definition of user-defined special node.

Tree-Invent allows defining special node for user specified molecular fragment in structure generation, which is useful in scaffold decoration and linker design settings. As shown in Figure
5 (d), in a special node, a list of saturation atom $S$ and the anchor atom $A_{anchor}$ are defined to specify the positions in the fragment to which substitutions cannot be added and the anchor position in the fragment. This information will be used for defining the connection constraints.

Figure 6. Some examples of topology constraint: (a) a template of molecular topology tree with which generated structures must satisfy; (b) The two central nodes (scaffold) are varied while their substitutes are fixed; (c) The scaffold on the LHS is fixed and the substitutes must have fixed topology; (d) Molecules have the specified terminal groups and linker part must has two nodes.

The above-mentioned topology masks can be employed in Tree-Invent model to carry out topology constrained structure generation and deal with tasks such as scaffold hopping, scaffold decoration and linker design etc. In the following section, we will go through the examples in Figure 6 to describe these individual generation processes.

(i) Molecular generation with given topology template

As shown in Figure 6a, a topology template is pre-defined and generated structures must satisfy with the template. Since any node type is allowed for the topology tree, no $M_{add}$ mask is used in the node sampling steps and only $M_{conn}$ masks are applied in each connection step. By doing in this way, Tree-Invent can generate structures satisfying the topology tree template. It is worthwhile to reminder that given the requirement of the topology tree, the constraint of termination is applied in the last step to finalize the structure generation.

(ii) Scaffold hopping with template
It is quite common in scaffold hopping exercises that ring systems are mutated to identify bioisosteric scaffold. For example, we would like to generate structures having topological constraints shown in Figure 6b, where only two central nodes (ring system or single atom) can be varied and all remaining atoms are fixed as shown in the template definition. Then we first traverse the topology graph of the template molecule with BFS order to decide node sequence. Following the node generation sequence, specific node addition constraint $M^l_{add}$ and connection constraint $M^l_{conn}$ are applied for sampling the fixed atoms, while for those unconstrained atoms, the constraints are spared in their sampling steps accordingly.

(iii) Scaffold decoration

Scaffold decoration is an exercise commonly used by medicinal chemists in lead optimization in which the scaffold part is kept constant and various substituents are explored to optimize bioactivity. In this scenario, the scaffold part is predefined as a special node in the topological graph, in which all atoms except atom 12 are in the saturated atom list. Tree-Invent also support the topology definition of R-group. As shown in Figure 6c, the R group is specified to be a topology tree of two branches connected to the C12 atom. In this case, the connection constraint for atom 13 is defined that the C12 position is the only position to be sampled. For sampling other atoms in the R-group, no constraint is applied in the node addition steps and only connection constraints are applied.

(iv) Linker design

Linking fragments is an important design strategy in fragment-based drug design, development of PROTAC, and scaffold hopping. Linker design can also be regarded as a topology constrained structure generation problem and can be addressed by Tree-Invent model. As shown in Figure 6d, the carboxylic acid and fluorobenzene groups are terminal groups and are defined as special nodes with specific saturated atom list. The linker part contains two nodes as depicted in the figure. If we started from the node of carboxylic acid to grow, the node 3 is firstly sampled without applying constraint and then connects to the C1 atom in carboxylic acid via applying a connection constraint. Similarly, the node 4 is sampled freely and connected to node 3 via constraint. Finally, the special node fluorobenzene is then connected to node 3 on its anchor atom C5.

E. Integrating Reinforcement learning (RL) with Tree-Invent
In order to guide the structure generation toward targeted properties, RL algorithm is employed in Tree-Invent to facilitate structure optimization. Molecule generation with Tree-Invent could be formalized as a Markov decision process with a set of actions \( A = \{a_0, a_1 ... a_n-1\} \), corresponding to graph updating from an empty graph \( G_0^M \) to the final molecular graph \( G_N^M \). The RL algorithm used in REINVENT\textsuperscript{18} and RL-Graph-INVENT\textsuperscript{23} was implemented here. A pre-trained Tree-Invent model was used as the Prior model, an Agent model was initialized with the same parameters of the Prior model and updated to optimize compound scores. The probability of sampled molecular graph \( G_M \) with tree constraints \( C_M \) is the product of action probabilities as shown in Eq (23)

\[
P(\mathcal{A}) = \prod_{n=1}^{N} \pi(a_n|G_n^M, C_n^M)
\]  

(23)

where \( \pi \) is the likelihood of next action \( a_n \) in Tree-Invent model and \( G_n^M \) represents the molecular graph at \( n \)th step. Assuming \( S(\mathcal{A}) \in [-1,1] \) is a scoring function that rates the desirability of generated action sequence. The goal of RL is to update the agent policy \( \pi \) from the prior policy \( \pi_{\text{prior}} \) to optimize the score of generated sequences while still anchoring to the prior policy, which is learnt from molecular structures in training set. Thus, an augmented likelihood \( \log P(\mathcal{A})_u \) is defined as Eq. (24):

\[
\log P(\mathcal{A})_u = \log P(\mathcal{A})_{\text{prior}} + \sigma S(\mathcal{A})
\]  

(24)

Where \( \sigma \) is a scalar coefficient. The return \( G(\mathcal{A}) \) in this case can be seen as the agreement between the agent likelihood \( \log P(\mathcal{A})_A \) and the augmented likelihood:

\[
G(\mathcal{A}) = -[\log P(\mathcal{A})_u - \log P(\mathcal{A})_A]^2
\]  

(25)

Then, the Agent model can learn the optimal policy by minimizing the cost function about the model parameters \( \theta \) as shown in Eq (26):

\[
L(\theta) = -G(\mathcal{A})
\]  

(26)

\textbf{F. Computational details and datasets}

Our Tree-Invent model was trained on the GuacaMol dataset\textsuperscript{24} consisting of 1.2 million molecules selected from ChEMBL\textsuperscript{24,25}. As synthesizable rings in drug-like compounds are limited, a ring system survey was done on the GuacaMol database by extracting all existing ring systems. Obtained ring systems were classified with the NTF mentioned before and 242 classes of ring system, which cover 97% ring system in the GuacaMol database, were selected
as pre-defined ring node for structure generation. The frequency distribution of the top 50 ring classes was listed in Figure S1 of the supporting material.

The maximal atom number of molecule $V_{max}^M$ and ring system $V_{max}^R$ in GuacaMol dataset were set to 46 and 34, respectively and the maximal ring node and tree node were set to 8 and 42. For training data generation, the maximal state for $Node_{add}$, $Ring_{gen}$, and $Node_{conn}$ were set to 42, 62, and 42. As discussed previously, there were 9 types of single atom node and 242 ring classes. Thus, the number of node type (i.e. $|A_M|$) was 251. All molecules were neutralized and saved in Kekule form, i.e., only single, double and triple bond were considered in the model. The whole GuacaMol dataset was split into the training set, validation set, and test set with a ratio of 0.9:0.05:0.05.

In Tree-Invent, all MLP blocks were constructed with four hidden layers and each layer had 800 neurons. In GGNN blocks, the times of message passing propagation for each graph node was set to 3. Here, three models using different atom traverse methods were built: 1) Tree-Invent-RDFS model, topological tree was traversed with breadth-first search (BFS) from the start point specified by canonical order and the ring structure was done with depth-first search (DFS) ; 2) Tree-Invent-RBFS model, topological tree was traversed with BFS from the start point specified by canonical order and the ring was done with BFS; 3) Tree-Invent-Random model, topological tree was traversed with BFS from random start point and the ring was done in DFS. All models used the Tanh activation function and were optimized with the Adam algorithm and decayed learning rate. The training process stopped when learning rates dropped to $10^{-6}$. We adopted an early stop strategy for saving computational cost, and the evolution of $Loss_{add}$, $Loss_{ring}$ and $Loss_{conn}$ can be seen in Figure S2.

Additionally, several target specific datasets were selected for utility of Tree-Invent in RL, transfer learning as well as multiple topology constrained tasks such as scaffold hopping, scaffold decoration and linker design etc. Their information can be found in Table 3. In the case of DRD2, ADAM17 and S1PR1, Scikit-learn SVC models were built as bioactivity scoring function for RL.

**Table 3.** Several datasets used for transfer learning and RL and the classification performance of SVC model for test sets.
Datasets | actives | inactives | Mean accuracy of SVC model | ROC-AUC
--- | --- | --- | --- | ---
DRD2 | 7218 | 343204 | 98.84% | 98.85%
CDK4 | 193 | / | / | /
ADAM17 | 1507 | 365560 | 99.65% | 99.67%
S1PR1 | 1366 | 70698 | 98.50% | 98.48%

Results and Discussions

A. Performance of Tree-Invent on GuacaMol benchmark dataset

Tree-Invent model was first trained on the GuacaMol training set and its performance was evaluated on a sampled set containing 50000 compounds. A set of parameter including validity, uniqueness, novelty, Frechet ChemNet Distance (FCD)\(^26\) and KL divergence\(^27\) between the sampled set and training set was shown in Table 4 in comparing with some well-known generative models. As mentioned in previous section, using different traversing method could lead to different generative models. It can be seen from Table 4 that structure validity of Tree-Invent models is in general higher than that of LSTM, Graph-Invent, auto-encoder and GAN models, indicating Tree-Invent’s capability of learning grammar of constructing molecule graph. Uniqueness and novelty of Tree-Invent models are close to 95% and 92%, which is comparable to other models. The KL divergence to training set is not as good as SMILES LSTM and VAE, but it is in the same level to Graph-Invent and better than Graph MCTS and ORGAN. The same trend can be seen for comparison of FCD metrics. Among three Tree-invent models, Tree-Invent-RBFS achieved best results in KL divergence and FCD metrics and the performance on other metrics are quite similar. Thus, Tree-Invent-RBFS was used as the default model in the following discussion.

Table 4. Performance comparison of Tree-Invent with other benchmark models

<table>
<thead>
<tr>
<th>Model</th>
<th>Validity↑</th>
<th>Uniqueness↑</th>
<th>Novelty↑</th>
<th>KL divergence↑</th>
<th>FCD↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random sampler(^a)</td>
<td>1.000</td>
<td>0.997</td>
<td>0.000</td>
<td>0.998</td>
<td>0.929</td>
</tr>
<tr>
<td>SMILES LSTM(^24)</td>
<td>0.959</td>
<td>1.000</td>
<td>0.912</td>
<td>0.991</td>
<td>0.913</td>
</tr>
<tr>
<td>Graph MCTS(^24)</td>
<td>1.000</td>
<td>1.000</td>
<td>0.994</td>
<td>0.522</td>
<td>0.015</td>
</tr>
<tr>
<td>AAE(^24)</td>
<td>0.822</td>
<td>1.000</td>
<td>0.998</td>
<td>0.886</td>
<td>0.529</td>
</tr>
<tr>
<td>ORGAN(^24)</td>
<td>0.379</td>
<td>0.841</td>
<td>0.687</td>
<td>0.267</td>
<td>0.000</td>
</tr>
<tr>
<td>VAE(^24)</td>
<td>0.870</td>
<td>0.999</td>
<td>0.974</td>
<td>0.982</td>
<td>0.863</td>
</tr>
<tr>
<td>Tree-Invent-RDFS</td>
<td>0.980</td>
<td>0.945</td>
<td>0.921</td>
<td>0.839</td>
<td>0.521</td>
</tr>
<tr>
<td>Tree-Invent-RBFS</td>
<td>0.979</td>
<td>0.956</td>
<td>0.921</td>
<td>0.862</td>
<td>0.581</td>
</tr>
<tr>
<td>Tree-Invent-Random</td>
<td>0.986</td>
<td>0.963</td>
<td>0.933</td>
<td>0.828</td>
<td>0.534</td>
</tr>
</tbody>
</table>
Note: a) randomly picking structures from GuacaMol training set

B. Reinforcement learning with Tree-Invent for DRD2

Here, we evaluated the performance of Tree-Invent on bioactivity guided molecular generation with reinforcement learning (RL) in the case of generating actives of dopamine type 2 receptor DRD2. Previously Olivecrona et al applied RL to train REINVENT model for DRD2 and their dataset was adopted for our model building.\textsuperscript{18}
Figure 7. The evolution of average activity scores $S_{\text{active}}$ (a), validity and uniqueness of generated molecules in RL training process of Tree-Invent (b). 10 random selected molecules generated in 15th (c), 40th (d) and 115th (e) epoch and their activity scores.

This dataset has 7,218 actives (pIC50>5) and 343,204 inactive as shown in Table 3. All the actives and 10000 randomly selected inactive are used as the reference set for training a support vector machine classifier with a Gaussian kernel. We split the datasets into training set and test set with ratio of 0.9:0.1. For a given molecule, ECFP4 fingerprints was used for building SVC classification model of Seikit-learn. The prediction accuracy and the area under the roc curve (ROC-AUC) of the SVC model reaches 98.84% and 98.85% for the test set. Thus, SVC predicted probability of being active was used as the scoring function (as in Eq. 27) to guide the Tree-Invent model exploring chemical space of DRD2 actives.

$$S_{\text{active}}(\text{Mol}) = \begin{cases} 0 & \text{if invalid or not unique} \\ SVC(\text{ECFP4}_\text{Mol}) & \text{if valid and unique} \end{cases}$$  \hspace{1cm} (27)

The evolution of average activity score of generated molecules is shown in Figure 7 (a). It increases quickly and converges to 0.95 after 100 epochs of RL iterations, indicating that Tree-Invent can efficiently explore the DRD2 chemical space through RL. This can be exemplified by the structures and activity scores of ten randomly sampled molecules at 15, 40, and 115th epoch as shown in Figure 7c-7e, respectively. This case study demonstrated that Tree-Invent model can be optimized toward target properties through reinforcement learning.

C. Transfer learning with Tree-Invent for CDK4 inhibitor design.

Transfer learning is a useful method for guiding the generative model to sample in the chemical space near a set of known molecules. Here, we evaluate the learning capability of Tree-Invent in terms of doing transfer learning to generate similar molecules to known inhibitors of cyclin-dependent kinase 4 (CDK4, ChemBL ID: CHEMBL331). In total, 1980 known CDK4 inhibitors were extracted from ChemBL and a clustering analysis was done by using ECFP4 fingerprint and similarity cut-off of 0.7. 193 compounds of the largest cluster were selected as training set for transfer learning. The Tree-Invent model trained on GuacaMol dataset was used as a Prior model and trained on the CDK4 training set. At each training epoch, a batch of 64 molecules was sampled from the model. For each compound in the sampled set, its highest pairwise Tanimoto similarity to the compounds of training set was obtained and the
average similarity \((Sim_{ave})\) between the sample set and training set was then calculated to measure the distance between the two sets. As shown in Figure 8, the \(Sim_{ave}\) quickly increases to 0.9 within 17 epochs, indicating fast learning of structural features of CDK4 inhibitors by the model.

![Figure 8](image)

**Figure 8.** (a) The average Tanimoto similarity \(Sim_{ave}\) between generated molecules and training sets in transfer learning process. (b) Tree-Invent generated molecules after transfer learning on CDK4 inhibitors. The max similarity to the training set is labeled below the molecules.

### D. Structure generation with topology constraints

Scaffold hopping, scaffold decoration and linker design are common exercises in drug discovery. Its key idea is to design novel compounds by only replacing part of a reference structure to make sure the bioactivity of new compounds can be at the same level with that of reference structure or even better. Developing generative model for these scenarios would require molecule generation under structure constraints. Previous efforts for developing structural constrained generative model can be found in literature. For example, Syntalinker\textsuperscript{31},
DR-linker\textsuperscript{32}, Link-Invent\textsuperscript{33}, DiffLinker\textsuperscript{34} and De-linker\textsuperscript{35} algorithms were developed for linker generation; GraphGMVAE\textsuperscript{36}, and Deep-Hops\textsuperscript{37} methods were proposed for scaffold hopping; Scaffold Decorator\textsuperscript{38}, SAMOA\textsuperscript{39} and Lib-Invent\textsuperscript{40} were dedicated for scaffold decoration. So far to best of our knowledge, to gain precise structure control in these various scenarios would need using generative model with different architectures and there is no single model can address all these tasks. One unique feature of Tree-Invent is that topological tree is used as constraint to gain precise control in the structure generation and it can deal with all three tasks. The performance of topology constrained molecular generation with Tree-Invent in the case of scaffold hopping, scaffold decoration and linker design will be discussed in the following section.

**Topology constrained scaffold hopping**

In current study, nucleoside analogue generation was used as an example to demonstrate the capability of Tree-Invent in generating topology constrained structures. Nucleosides and their analogues have been widely used in treating acute and chronic viral infections due to their inhibition of viral replication. Nucleoside scaffolds are mainly focused on five natural nucleosides, ie. Adenine, Guanine, Cytosine, Uracil and Thymine nucleosides, whose patent space is very crowded. Identifying novel nucleoside scaffold is highly desirable for developing anti-virus drugs.

The structure of adenine nucleoside is shown in Figure 9a, consisting of a 5,6 fused ring system, a ribose ring, and amine and alcoholic substituents. Its molecular graph could be expressed as a tree structure shown in Figure 9b and a topological tree constraint was defined, in which the ribose ring, amine and alcoholic substituents should be kept unchanged and Nitrogen containing fused ring system is allowed to be varied. So, our task here is to train a Tree-Invent model for generating structures satisfying the tree constraint. Firstly, a list of mask tensor was defined according to the tree constraint and applied in the model sampling process. The Tree-Invent prior model trained on GuacaMol was used, together with those mask tensors, to sample molecules. 1200 compounds were sampled from the model, the validity is around 0.95. twenty randomly selected generated molecules are shown in Figure 9c and they all fulfill the topology tree constraints. Among them, compound 19 has the adenine nucleoside structure and, interestingly, the compound 2 has the same nucleoside group of Remdesivir, an approved
drug for the SARS-Cov2 virus. These results indicate that the Tree-Invent model can combine with tree constraints for structure constrained molecule generation.

![Figure 9](image)

**Figure 9.** (a) Structure of adenine, (b) The tree structure of Adenine (c) Generated molecules with tree constraints.

We also explored Tree-Invent for scaffold hopping in the RL setting, which means the topology constrained structure generation is embedding into the RL process. The Cox2 inhibitor Celecoxib was taken as an example and the aim is to generate topology constrained compounds with high similarity to the reference structure. Here, two kinds of topology constraints were defined for the similarity guided molecular generation (Figure 10). In the first case, a topology tree was defined as in Figure 10d (based on the topology tree of Celecoxib), it contains 9 single
atom nodes for any atom type and three ring nodes in while two of them could be any aromatic ring and one node is for any five-member ring containing two nitrogen atoms. For the other topology tree constraint (shown in Figure 10g), those red nodes were determined atoms corresponding to the substituents in Celecoxib and the three grey nodes composed the scaffold of Celecoxib which need to be sampled. Any type of ring is allowed to be sampled for grey nodes. For RL iterations, the similarity \( S_{\text{sim}} \) as shown in Eq. 28 to the reference with a threshold \( k \) of 0.75 was used as the RL scoring function:

\[
S_{\text{sim}} = \begin{cases} 
0 & \text{if invalid or not unique} \\
\min(Sim, 0.75) / 0.75 & \text{if valid and unique}
\end{cases}
\]  

(28)

The first constraint is used to guide generation of similar compounds to the reference and the second constraint is more or less mimicking the scaffold hopping exercise, in which non-ring atoms are fixed and ring nodes have fixed connection and their node type can be varied. The training processes under both constraints can be seen in Figure 10e, h, and in both cases, the \( S_{\text{sim}} \) of generated batch exceeds 0.9 after roughly 40 epochs and validity can reach 1. In general, the uniqueness can keep at 0.8 in both constrained molecular generations. As a comparison, we also perform an unconstrained RL with the same scoring function as shown in Figure 10b, c, the \( S_{\text{sim}} \) of generated batch can only converge at 0.45. These results indicate the constrained RL iteration are more efficient in similarity guided molecular generations. Some randomly selected examples from the RL can be seen in Figure 11.
Figure 10. The structure of Celecoxib (a) and two different topology constraints used for structure generation (d, g). The evolution of average Tanimoto similarity of generated batch $S_{slm}$, validity and uniqueness during RL under non-constraint (b, c), constraint d (e, f), constraint g (h, i). Here valid molecule means that the compound fulfills structure validity as well as the topology constraint.
Figure 11. (a) Generated structures without constraint; (b) Generated structures under constraint in Figure 10d; (c) Generated structures under constraint in Figure 10g. The numbers refer to the $S_{sim}$ to Celecoxib.

Topology constrained scaffold decoration

The performance of Tree-Invent on topology constrained scaffold decoration in RL setting was also explored. The goal of utilizing RL is to generate structures with optimal bioactivity to target protein. Here, ADAM metallopeptidase domain 17 (ADAM17), which is critically involved in diverse signaling pathways controlling physiological and pathophysiological processes, was chosen as the target protein. The ADAM17 dataset which contains 1507 actives
(pIC50>5) and 365560 inactives was downloaded from Excape-DB. Following the same procedure of building the above DRD2 prediction model, all ADAM17 actives and the same number of randomly selected inactives were used as the dataset, the training set and test set were split with ratio of 0.9:0.1 and a support vector machine classifier with a Gaussian kernel was trained and tested. The mean accuracy and ROC-AUC of the ADAM17 prediction model were both 0.997 on test set and the scoring function of RL is the same to Eq. 27.

**Figure 12.** Scaffold decoration on 4-(phenylsulfonyl) thiomorpholine of ADAM17 actives. (a, d) Defined topology constraints; Evolution of average activity score (b), validity and uniqueness (c) for RL running under constraint a; Evolution of average activity score (e), validity and uniqueness (f) for RL running under constraint d. Here valid molecule means that the compound fulfills structure validity as well as the topology constraint.

Here, the common substructure of 4-(phenylsulfonyl)thiomorpholine in actives were defined as the scaffold for decoration. Two topology constraints were defined in the study (as shown in Figure 12a, d) and two substitution positions $T_1$ and $T_2$ were specified for adding R-group. For the first constraint, only attachment points on the scaffold were defined and no topology constraint of R-group was specified. It can be seen from Figure 12b that after 100
epochs, the average activity score can reach 0.95 and the validity and uniqueness of generated batches can converge at 1.0 and 0.9 respectively. Generated structures using the first constraint can be seen in Figure 13a, it was noted that most of structures have hydrogen bond acceptor group in T2 position closed to the thiomorpholine ring, which can probably form metal ion interaction with the Zn ion in the ADAM17 catalytic site.

![Example ADAM17 active](image)

Figure 13. (a) Scaffold decoration examples generated with only position constraints, (b) Scaffold decoration examples generated with topology constraints. An example ADAM17 active structure is shown at the top and the activity scores of generations are displayed in the figures.

In the second case, a stricter constraint as shown in Figure 12d was defined for structure generation. The R-group at T1 was required to have a ring and an oxygen atom, while the T2 substitution consists of 4 unconstrained nodes and 1 ring node. After 300 epochs the average
activity score increased to around 0.9. The validity can reach 1.0 and uniqueness converged to 0.75. Some randomly chosen structures are listed in Figure 13b. Similarly, it seems all structures fulfill the topology constraint and highly scored compounds have a hydrogen bond acceptor group in T2 closed to the thiomorpholine ring, probably suggesting the interaction with the Zn ion in the catalytic site.

**Topology constrained linker design**

Linker design using Tree-Invent model was also explored in current study and one dataset were selected for demonstration. It is a dataset of sphingosine-1-phosphate receptor 1 (S1PR1) inhibitor consisting of 1366 actives as listed in Table 3. Similar to above examples, all the actives and same number of randomly selected inactive were used as the reference set for training a support vector machine classifier (SVC) with a Gaussian kernel. The training set and test set were split with ratio of 0.9:0.1 and the mean accuracy and ROC-AUC of classification models were both 0.985 on test set of S1PR1. Terminal group and linker fragment defined for each target can be seen in Figure 14a, where specific topology constraints were imposed on the linker moieties. Similar to the scaffold decoration case, RL processes were carried out to generate linkers under topology constraints. Interestingly, Tree-Invent can find proper solutions with less than 100 RL iterations and the activity score quickly converged at 0.9 for S1PR1, while the validity reached 1.0 and the uniqueness exceeded 0.75. Some selected examples can be seen in Figure 14d, it seems that all structure satisfy the specified topology constraints.

Through the above examples, it can be seen that Tree-Invent model can not only generate structures in unconstrained setting, but also works nicely with RL and transfer learning. More importantly, it is a highly flexible generative model for dealing with multiple tasks which need to incorporate topology constraints, including scaffold hopping, scaffold decoration and linker design.
Figure 14. (a) Defined topology constraint for linker of S1PR1 inhibitor. Evolution of average activity score (b), validity and uniqueness (c) of generated HTR1A inhibitors. Here valid molecule means that the compound fulfills structure validity as well as the topology constraint. Molecules generated with linker topology constraints for S1PR1 are provided (d). The average activity scores are displayed in the figure and two example actives are displayed.

Conclusions

Here, we proposed a novel molecular generative model, Tree-Invent, which is a hierarchical generative model. In this model, the complex ring system, non-ring atoms, and chemical bonds were represented by virtual ring nodes, single nodes, and edges, so that molecular graph is equivalent to a multi-fork tree. In Tree-Invent methodology, three sub-models were trained independently for carrying out node addition, ring formation and node connection operations and they can be integrated conveniently for doing auto-regressive molecule generation. By introducing a unique mask mechanism, Tree-Invent model can be easily adapted to structure generation under topology constraint and provide structure control more precisely. By combining with reinforcement learning, Tree-Invent is capable of dealing with versatile constrained structure generation tasks, e.g. scaffold hopping, scaffold decoration.
and linker design etc. Given those unique features, we expect Tree-Invent could be served as a powerful tool for doing \textit{de novo} drug design.

\textbf{Availability:}

The source code and related examples of Tree-Invent are available from https://github.com/MingyuanXu/Tree-Invent.

\textbf{Supporting Information Available}

1. The routine of \textit{sample}_{add}, \textit{sample}_{ring}, \textit{sample}_{conn} and algorithms of translation between user-defined conditions on molecular topology $C_{add}, C_{conn}$ to $M_{add}$ and $M_{conn}$.
2. The frequency of top 50 class of ring systems and its finger prints are as shown in Figure S1.
3. The training loss and validation loss of three modules are as shown in Figure S2
4. The training sets in case of CDK4 inhibitors are as shown in Figure S3.

\textbf{Acknowledgement}

M.X. would like to thanks Dr Y. Zhang for her help in compiling the manuscript.

\textbf{Reference:}


Dey, R. & Salem, F. M. in 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS). 1597-1600 (IEEE).


TOC:
Supporting Information

Tree-Invent: Novel molecular generative model constrained with topological tree

Mingyuan Xu¹, Hongming Chen¹,*

¹ Guangzhou Laboratory, No. 9 XingDaoHuanBei Road, Guangzhou International Bio Island, Guangzhou 510005, Guangdong Province, China

*Correspondence e-mail: chen_hongming@gzlab.ac.cn;

Routine S1. sample $A^M, E^M$ from APD for Node add action.:

```python
def sample_add(APD):
    id_add = choice(APD)
    $A^M, E^M = \begin{cases}
    (id_add, 0) & \text{if } id_add < |A^M| \\
    (\text{None}, 1) & \text{if } id_add \geq |A^M|
    \end{cases}$
    return $A^M, E^M$
```

Routine S2. sample $V^R_{\text{add}}, A^R, F, B_{\text{add}}, V^R_{\text{conn}}, B^R_{\text{conn}}, E^R$ from APD$_{\text{ring}}$ for ring node generation action.:

```python
def sample_ring(APD$_{\text{ring}}$):
    id_ring = choice(APD$_{\text{ring}}$)
    L$_{\text{add}} = \text{product}(V^R_{\text{max}}, |A^R|, |F|, |B|)$
    R$_{\text{add}} = \text{APD}_{\text{ring}}[: L_{\text{add}}], \text{reshape}(V^R_{\text{max}}, |A^R|, |F|, |B|)$
    $V_{\text{add}}, A^R, F, B_{\text{add}} = \begin{cases}
    \text{unzip} \left( \text{index}(\text{APD}_{\text{ring}}[id_{\text{ring}}, R_{\text{add}}]) \right) & \text{if } id_{\text{ring}} < L_{\text{add}} \\
    \text{None}, \text{None}, \text{None}, \text{None} & \text{if } id_{\text{ring}} \geq L_{\text{add}}
    \end{cases}$
    L$_{\text{conn}} = \text{product}(V^R_{\text{max}}, |B|)$
    R$_{\text{conn}} = \text{APD}_{\text{ring}}[\text{product}(V^R_{\text{max}} & |A^R|, |F|, |B|) : -2]$
    $V^R_{\text{conn}}, B^R_{\text{conn}} = \begin{cases}
    \text{unzip} \left( \text{index}(\text{APD}_{\text{ring}}[id_{\text{ring}}, R_{\text{conn}}]) \right) & \text{if } L_{\text{add}} < id_{\text{ring}} < L_{\text{add}} + L_{\text{conn}} \\
    \text{None}, \text{None} & \text{if } id_{\text{ring}} > L_{\text{add}} + L_{\text{conn}}
    \end{cases}$
```
\[ E^R = \begin{cases} 1 & \text{if } id_{add} = L_{R_{add}} + L_{R_{conn}} + 1 \\ 0 & \text{if } id_{add} \neq L_{R_{add}} + L_{R_{conn}} + 1 \end{cases} \]

return \( V_{add}^R, A^R, F, B_{add}, V_{conn}^R, B_{conn}^R, E^R \)

**Routine S3.** Sample \( V_{conn}^M, B_{conn}^M \) from \( APD_{conn} \) for node connect action.

```
def sample_conn(APD_conn):
    id_conn = choice(APD_conn)
    \( V_{conn}^M, B_{conn}^M = unzip(index(APD_conn[id_conn], APD_conn, reshape(V_{max}^M, |B|))) \)
    return \( V_{conn}^M, B_{conn}^M \)
```
Algorithm S1. Generation of $M_{\text{add}}$ with predefined conditions about node $C_{\text{add}}$.

\[ i = 0; M_{\text{add}} = [] \]

for $i$ in range($N_{\text{step}}$):

\[ M_{\text{add}}^i = \text{ones}(|\mathcal{A}^M|) \]

if $C_{\text{add}}^i = \text{None}$:

\[ N_R^{\text{max}}, N_R^{\text{min}}, N_A^{\text{max}}, N_A^{\text{min}}, N_{\text{aro}}^{\text{max}}, N_{\text{aro}}^{\text{min}}, N_{\text{br}}^{\text{max}}, N_{\text{br}}^{\text{min}} = \text{unzip}(C_{\text{add}}^i) \]

for $j$ in range($|\mathcal{A}^M|$):

if $N_R^{j} > N_R^{\text{max}}$ or $N_R^{j} < N_R^{\text{min}}$:

\[ M_{\text{add}}^i[j] = 0 \]

for $k$ in range($N_A$):

if $N_A^{j}[k] > N_A^{\text{max}}[k]$:

\[ M_{\text{add}}^i[j] = 0 \]

if $N_{\text{aro}}^{j} > N_{\text{aro}}^{\text{max}}$ or $N_{\text{aro}}^{j} < N_{\text{aro}}^{\text{min}}$:

\[ M_{\text{add}}^i[j] = 0 \]

if $N_{\text{br}}^{j} > N_{\text{br}}^{\text{max}}$ or $N_{\text{br}}^{j} < N_{\text{br}}^{\text{min}}$:

\[ M_{\text{add}}^i[j] = 0 \]

\[ M_{\text{add}} \cdot \text{append}(M_{\text{add}}^i) \]
Algorithm S2. Generation of $M_{\text{conn}}$ with predefined conditions about node $C_{\text{conn}}$.

$i = 0$

$M_{\text{conn}}=[]$

for $i$ in range($N_{\text{step}}$):

$M_{\text{conn}}^i = \text{ones}(V_{\text{max}}^M, 3)$

if $C_{\text{conn}}^i = \text{None}$:

$G_{\text{conn}}, B_{\text{conn}}, A_{\text{conn}} = \text{ unzip}(C_{\text{conn}}^i)$

for $j$ in range($V_{\text{max}}^M$):

if $G_{\text{id}}[j] \neq G_{\text{conn}}$:

$M_{\text{conn}}^i[j] = 0$

else:

if $A[j] \not\ni A_{\text{conn}}$:

$M_{\text{conn}}^i[j] = 0$

else:

for $k$ in range(3):

if $k \not\ni B_{\text{conn}}$:

$M_{\text{conn}}^i[j,k] = 0$

$M_{\text{conn}}.\text{append}(M_{\text{conn}}^i)$
Figure S1. The frequency of top 50 class of ring systems in GuacaMol datasets
Figure S2. The evolution of $Loss_{add}$ (a), $Loss_{ring}$ (b) and $Loss_{conn}$ (c) of $Node_{add}$ module in training sets and validation sets. Evolution of losses of $Ring_{gen}$ (d–f) and $Node_{conn}$ (g–i).
Figure S3. 60 random selected molecules in training set of CDK4 inhibitors