Group Meeting - February 19, 2021

Paper review & Research progress

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Content

- Literature review:
  - **Multi-Objective Molecule Generation using Interpretable Substructures** (ICML 2020)
    `https://arxiv.org/abs/2002.03244`
- Research update:
  - TensorFlow API for Cormorant/N-Body.

Paper

Multi-Objective Molecule Generation using Interpretable Substructures (ICML 2020)

Wengong Jin, Regina Barzilay, Tommi Jaakkola

https://arxiv.org/abs/2002.03244

Note

Source code:

https://github.com/wengong-jin/multiobj-rationale

Tommi Jaakkola (MIT) – Representation and generation of molecular graphs:

https://www.youtube.com/watch?v=ISX-mHnQhaw

Proposal

Note

I think we can extend this work into **transfer learning** and combine with the ICML 2021 submission of Erik and Wenda into **neural architectural search**.

Proposal

This work is earlier than **Molecule Optimization by Explainable Evolution (ICLR 2021)**, but the objective is the same:

- Find the set of molecular **rationales** (subgraphs) to maximize the molecular properties.
- Complete or combine the rationales into a full molecule by a graph **generative** model.

Overview

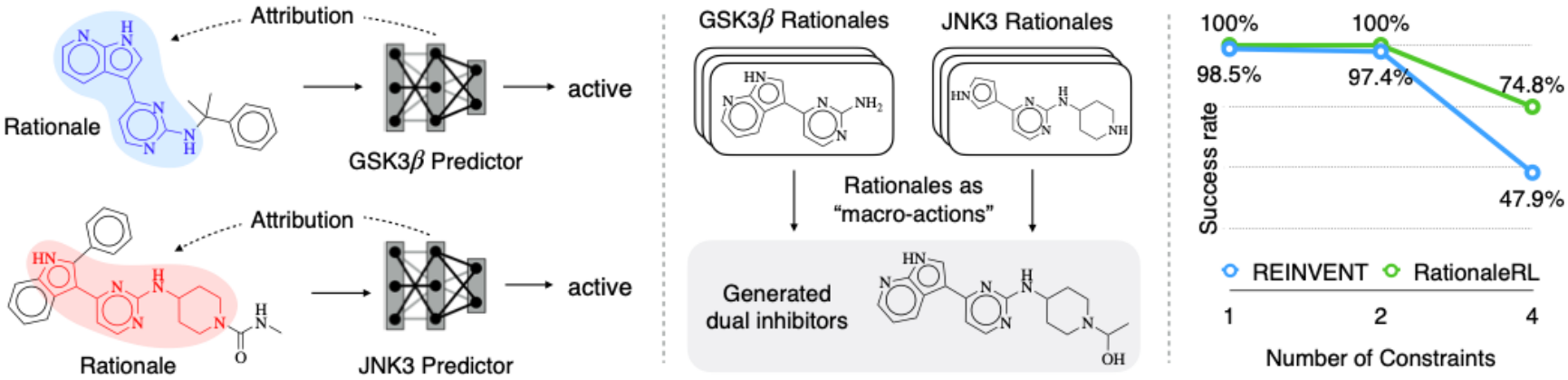


Figure 1. Illustration of RationaleRL. *Left*: To generate a dual inhibitor against biological targets GSK3β and JNK3, our model first identifies rationale substructures $\mathcal{S}$ for each property. Note that rationales are not provided as domain knowledge. *Middle*: The model learns to compose multiple rationales $\mathcal{S}$ into a complete molecule $\mathcal{G}$. *Right:* Our method achieves much higher success rate than the current state-of-the-art molecule design method REINVENT (Olivecrona et al., 2017)) under four property constraints.

Two complementary tasks:

1. Identification of the building blocks - rationales/subgraphs.
2. Assembling multiple rationales together into a fully formed target molecule.

Method (1)

A molecular graph $\mathcal{G}$ is generated from underlying rationale sets $\mathcal{S}$:

$$P(\mathcal{G}) = \sum_{\mathcal{S} \in V_{\mathcal{S}}^{|M|}} P(\mathcal{G}|\mathcal{S}) P(\mathcal{S})$$

where $V_{\mathcal{S}}^{|M|}$ is the vocabulary, and M is the number of property constraints. Property-constraint optimization:

$$\text{Find molecules} \quad \mathcal{G}$$

$$\text{Subject to} \quad r_i(\mathcal{G}) \geq \delta_i, \quad i = 1, .., M$$

For each property i, the property score $r_i(\mathcal{G}) \in [0, 1]$ of molecule $\mathcal{G}$ must be higher than threshold $\delta_i \in [0, 1]$.

Method (2)

Three modules:

1. **Rationale extraction:** Construct the rationale vocabulary $V_{\mathcal{S}}^{i}$ for each individual property i.
2. **Graph completion** $P(\mathcal{G}|\mathcal{S})$: Generate molecules $\mathcal{G}$ using multi-property rationales (note: M subgraphs) $S^{|M|} \in V_{\mathcal{S}}^{|M|}$.
3. **Rationale distribution** $P(\mathcal{S})$: A rationale $\mathcal{S}$ is sampled more frequently if it is more likely to be expanded into a positive molecule $\mathcal{G}$.

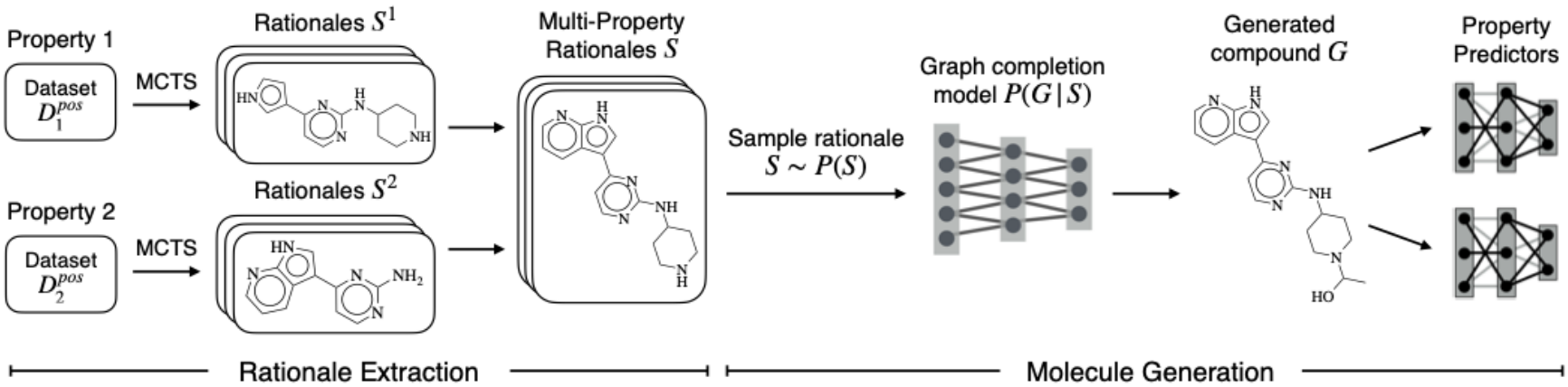


Figure 2. Overview of our approach. We first construct rationales for each individual property and then combine them as multi-property rationales. The method learns a graph completion model $P(\mathcal{G}|\mathcal{S})$ and rationale distribution $P(\mathcal{S})$ in order to generate positive molecules.

Rationale (subgraph structure) recognition model

Constraints:

1. The size of $\mathcal{S}^i$ should be smalle (less than 20 atoms) and **connected**.
2. Its predicted property score $r_i(\mathcal{S}^i) \geq \delta_i$. Property score is output of a pre-trained GNN.

Algorithm: Monte Carlo Tree Search (MCTS) – each step search to remove a substructure out of the current molecular graph.

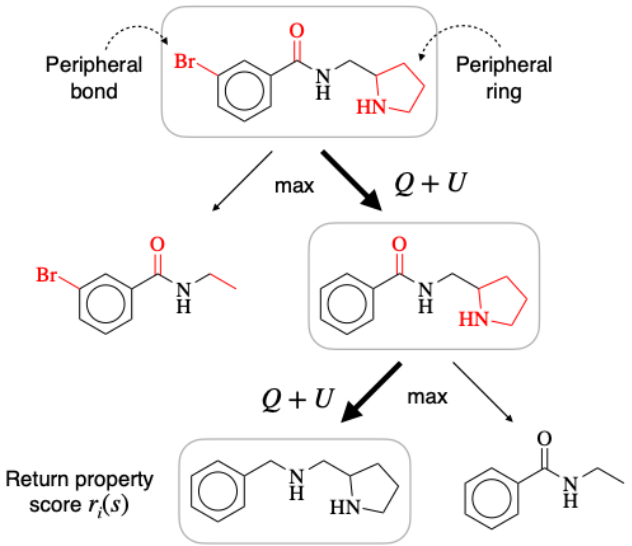


Figure 3. Illustration of Monte Carlo tree search for molecules. Peripheral bonds and rings are highlighted in red. In the forward pass, the model deletes a peripheral bond or ring from each state which has maximum $Q+U$ value (see Eq.(6)). In the backward pass, the model updates the statistics of each state.

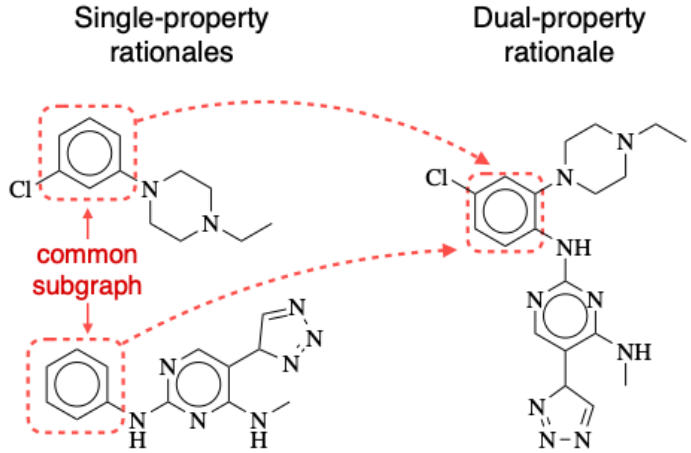


Figure 4. Illustration of multi-property rationale construction. Given two single-property rationales, we first find their maximum common substructure (MCS). If their MCS is not empty, we superpose one rationale on another so that their MCS coincides.

Graph completion

Two phases:

1. **Pre-training** $P(\mathcal{G}|\mathcal{S})$: use modified atom-by-atom VAE (CGVAE) – not generating from the scratch, but to incorporate the subgraph constraint.

$$P(\mathcal{G}|\mathcal{S}) = \int_{\boldsymbol{z}} P(\mathcal{G}|\mathcal{S}, \boldsymbol{z}) P(\boldsymbol{z}) d\boldsymbol{z}$$

2. **Fine-tuning** $P(\mathcal{G}|\mathcal{S})$: use gradient policy (reinforcement learning) with reward from property predictors (e.g. pre-trained GNN). Self-supervised manner.
   - Initialize the fine-tuning set $\mathcal{D}^f = \emptyset$. For each rationale $\mathcal{S}_i$, use the current model to sample K molecules:

$$\{\mathcal{G}_i^1, .., \mathcal{G}_i^K\} \sim P_\theta(\mathcal{G}|\mathcal{S}_i)$$

     Add $(\mathcal{G}_i^k, \mathcal{S}_i)$ to set $\mathcal{D}^f$ if $\mathcal{G}_i^k$ is predicted to be positive.
   - Update the model $P_\theta(\mathcal{G}|\mathcal{S})$ on the fine-tuning set $\mathcal{D}^f$ using policy gradient method.

Multi-property constraint optimization

Algorithm 1 Training method with n property constraints.

1: **for** $i = 1$ to M **do**
2: $V_{\mathcal{S}}^{i} \leftarrow$ rationales extracted from existing molecules $\mathcal{D}_{i}^{pos}$ positive to property i. (see §3.1)
3: **end for**
4: Construct multi-property rationales $V_{\mathcal{S}}^{[M]}$.
5: Pre-train $P(\mathcal{G}|\mathcal{S})$ on the pre-training dataset $\mathcal{D}^{pre}$.
6: Fine-tune model $P(\mathcal{G}|\mathcal{S})$ on $\mathcal{D}^{f}$ for L iterations using policy gradient.
7: Compute $P(\mathcal{S})$ based on Eq.(17) using fine-tuned model $P(\mathcal{G}|\mathcal{S})$.

Datasets

Datasets (note: only 1 property for each dataset, and very unbalanced between positives/negatives):

1. GNK3β: Inhibition against glycogen synthase kinase-3 beta. 2665 positives and 50K negative compounds.
2. JNK3: Inhibition against c-Jun N-terminal kinase-3. 740 positives and 50K negatives.

Metrics

Metrics (on 5,000 generated molecules):

1. **Success:** Fraction of sampled molecules predicted to be positive.
2. **Diversity:**

$$1 - \frac{2}{n(n-1)} \sum_{X,Y} \text{sim}(X, Y)$$

where $\text{sim}(X, Y)$ is pairwise molecular distance defined by the Tanimoto distance over Morgan fingerprints.

3. **Novelty:** For each generated positive compound $\mathcal{G}$, we find its nearest neighbor $\mathcal{G}_{\text{SNN}}$ from positive molecules in the training set.

$$\text{Novelty} = \frac{1}{n} \sum_{\mathcal{G}} 1[\text{sim}(\mathcal{G}, \mathcal{G}_{\text{SNN}}) < 0.4]$$

that is the fraction of molecules with nearest neighbor similarity lower than 0.4.

Experiments (1)

Table 1. Results on molecule design with one or two property constraints.

Method	GSK3β			JNK3			GSK3β + JNK3		
	Success	Novelty	Diversity	Success	Novelty	Diversity	Success	Novelty	Diversity
JT-VAE	32.2%	11.8%	0.901	23.5%	2.9%	0.882	3.3%	7.9%	**0.883**
GCPN	42.4%	11.6%	**0.904**	32.3%	4.4%	**0.884**	3.5%	8.0%	0.874
GVAE-RL	33.2%	76.4%	0.874	57.7%	62.6%	0.832	40.7%	80.3%	0.783
REINVENT	99.3%	**61.0%**	0.733	98.5%	31.6%	0.729	97.4%	39.7%	0.595
RationaleRL	**100%**	53.4%	0.888	**100%**	**46.2%**	0.862	**100%**	**97.3%**	0.824

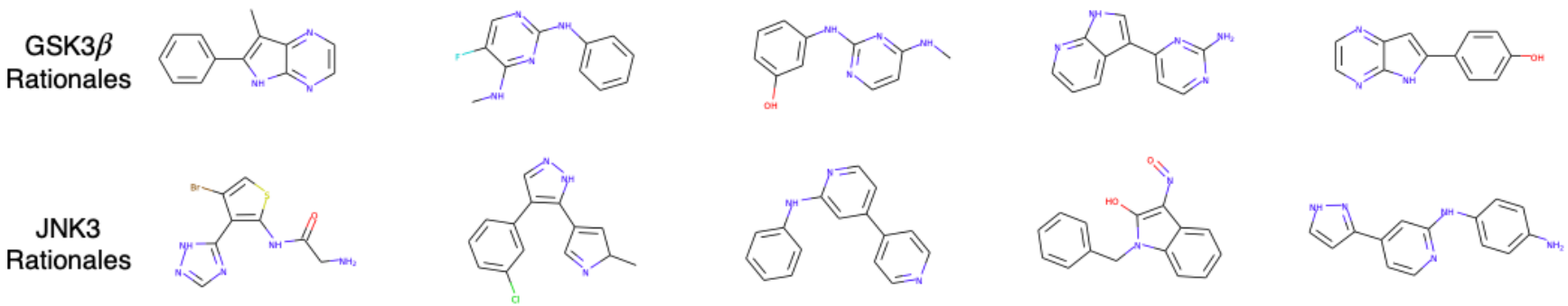


Figure 5. Sample rationales of GSK3β (top) and JNK3 (bottom).

Experiments (2)

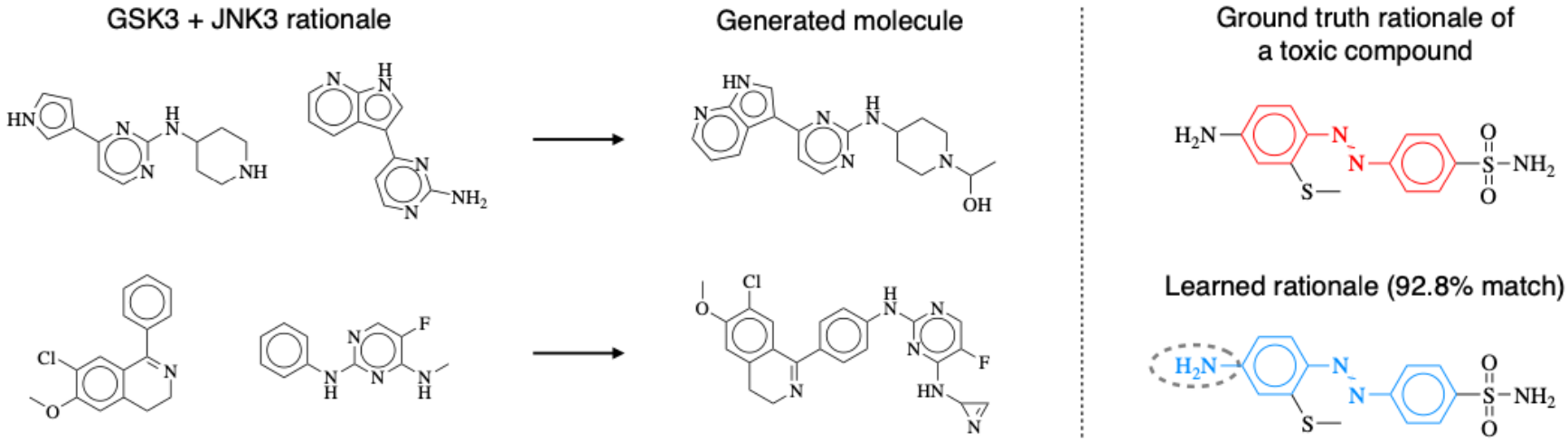


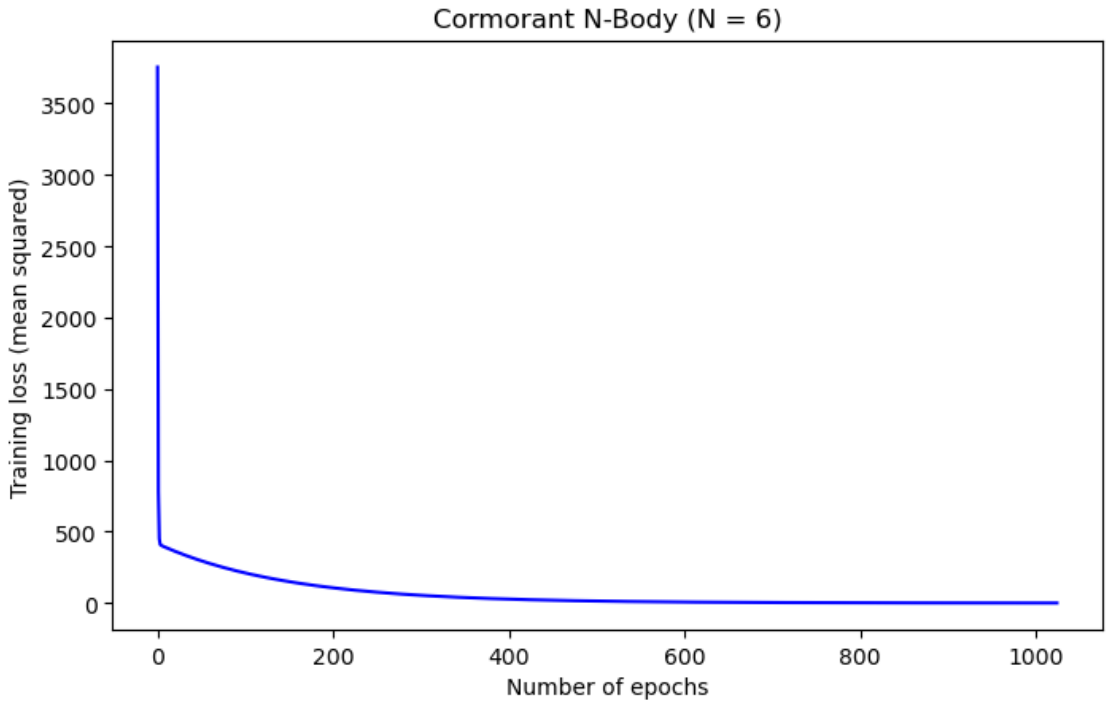
Figure 7. **Left**: Examples of molecules generated in the GSK3β+JNK3+QED+SA task. The model learns to combine two disjoint rationale graphs into a complete molecule. **Right**: Example structural alerts in the toxicity dataset. The ground truth rationale (Azobenzene) is highlighted in red. Our learned rationale almost matches the ground truth (error highlighted in dashed circle).

TensorFlow version 1 with non-eager mode

```
import tensorflow as tf
import numpy as np
# Load SO3 library
import sys
sys.path.append('SO3/')
import SO3_grad
SO3 = tf.load_op_library('SO3/SO3_batch.so')
# Interactive Session (TF 1)
sess = tf.InteractiveSession()
# Clebsch-Gordan table object (int64)
cg = sess.run(SO3.cg(5))
# Tensors
B = 100
nThreads = 14
v1 = [tf.random_normal([B, 1, 1, 2], dtype = tf.float32), tf.
    random_normal([B, 10, 3, 2], dtype = tf.float32), tf.
    random_normal([B, 20, 5, 2], dtype = tf.float32)]
v2 = [tf.random_normal([B, 3, 1, 2], dtype = tf.float32), tf.
    random_normal([B, 7, 3, 2], dtype = tf.float32)]
L1 = len(v1) - 1
L2 = len(v2) - 1
L = 3
# Tensor product
v = v1 + v2
product = SO3.tensor_product(v, cg = cg, L1 = L1, L2 = L2, L = L,
    nThreads = nThreads)
# Execution by Session
out = sess.run(product)
v_grad = tf.gradients(product, v)
```

TensorFlow version 2 with eager mode and Keras API

Passed the test of learning Coulomb force, the loss going down to 0 by SGD. For N = 6 atoms and 100 training examples:



The code snippet is too long to paste into a slide. The example is at `test_NBody.py`.