Molecular Graph Generation by Decomposition and Reassembling

Masatsugu Yamada* and Mahito Sugiyama*
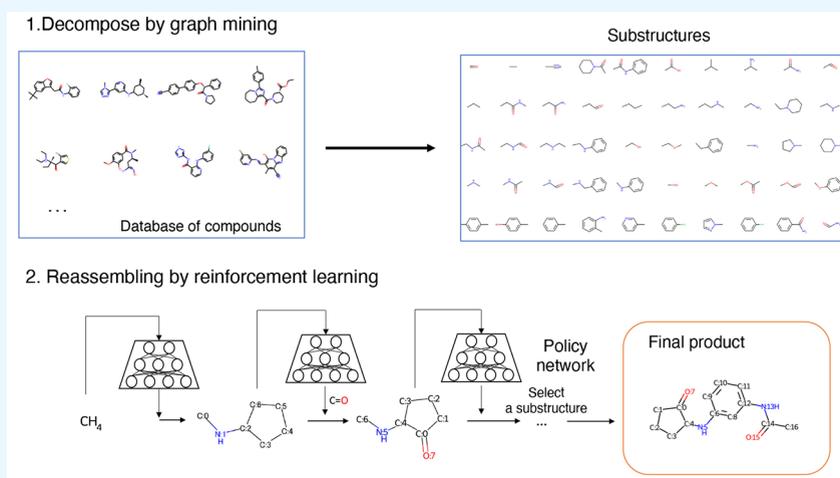
Read Online

ACCESS | 📊 Metrics & More | 📰 Article Recommendations

ABSTRACT: Designing molecular structures with desired chemical properties is an essential task in drug discovery and materials design. However, finding molecules with the optimized desired properties is still a challenging task due to combinatorial explosion of the candidate space of molecules. Here we propose a novel *decomposition-and-reassembling*-based approach, which does not include any optimization in hidden space, and our generation process is highly interpretable. Our method is a two-step procedure: In the first decomposition step, we apply frequent subgraph mining to a molecular database to collect a smaller size of subgraphs as building blocks of molecules. In the second reassembling step, we search desirable building blocks guided via reinforcement learning and combine them to generate new molecules. Our experiments show that our method not only can find better molecules in terms of two standard criteria, the penalized log *P* and druglikeness, but also can generate drug molecules showing the valid intermediate molecules.

INTRODUCTION

Designing new molecules for drugs and materials with desired properties is a challenging task due to the massive number of potential druglike molecules, which is estimated to be between 10^{23} and 10^{60}.[1,2] Although various types of representations have been investigated, molecules are essentially *graphs* with node and edge attributes.[3] The graph structure of chemical compounds makes it difficult to generate valid molecules with desired activities or properties even if you can build a Quantitative Structure−Activity Relationship (QSAR) model, which is a computational modeling method for revealing relationships between structural properties of chemical compounds and biological activities,[4] by designing descriptors of chemical features specifically for virtual screening. One of the straightforward ways of generating molecules is to solve the inverse QSAR problem through the objective function estimated from the molecular structures.[5−7] However, feature vectors extracted from molecular graphs are often highly correlated

between its features, which makes it challenging to reconstruct a new molecular graph from the optimized descriptors as it requires preserving such correlation information. In addition, molecules are often treated as 3D structures so that 3D descriptors can be calculated to predict desired properties more accurately. The inverse design of 3D molecular structures is also highly challenging because the coordination of atoms must be taken into consideration.[3,8−12]

A number of methods have been proposed to tackle the problem of molecular generation.[8,13,14] Recent advanced approaches to drug-candidate molecule findings have employed

deep generative models.[10−12,15−18] The basic idea of using generative models is to learn the latent representation of molecules, which enables us to reconstruct and explore molecules that satisfy target properties in the learned latent chemical space. In particular, for 3D molecular generation, not only the graph structures but also the conformation and distance between atoms are needed as additional node attributes.

Exploration methods such as Bayesian optimization are used to search the latent chemical space.[15] However, it is fundamentally difficult to reconstruct molecular graphs from the latent space and to search molecules with the desired property by extrapolation from a training data set as a large part of the latent space corresponds to invalid molecules.

Another strategy to search desired molecules is based on reinforcement learning. In the setting of reinforcement learning, an agent learns the optimal policy to maximize the cumulative reward, and the trained agent can take an action to generate the optimal molecules. When each molecule is represented as a string in the form of the simplified molecular-input line-entry system (SMILES),[19] the agent takes an action of the next character of SMILES based on the optimized policy, where recurrent neural networks (RNNs) are often used to generate strings. In the case of molecular graph generation using reinforcement learning, the agent takes an action of choosing the atom type and bond type between nodes to expand each molecule.[20] The state is represented as latent feature vectors by using RNNs or graph neural networks. However, both approaches of SMILES generation and nodewise molecular graph generation share the problem that the intermediate steps do not represent valid molecules, which significantly deteriorates the interpretability of resulting generated molecules. Moreover, the property and the state radically change if the ring structure appears, and it is fundamentally difficult to treat such binary response in optimization on a continuous latent space.

In this paper, we propose a novel molecular generation approach, called MOLDR (MOLecular graph Decomposition and Reassembling), which generates optimized new molecules by decomposing molecular graphs in a training data set into subgraphs and reassembling such obtained subgraphs again in a different way. Our key insight is that chemical properties depend on the combination of subgraphs, which correspond to the functional group or the motif of molecules in the context of chemoinformatics, and that it can be optimized when appropriate substructures are included in molecules. More specifically, MOLDR is composed of a *decomposition* step and a *reassembling* step. In the decomposition step, we first convert each molecular graph into a tree structure to efficiently obtain subgraphs, that is, functional groups, followed by extracting frequent subgraph structures by applying a graph mining method. In the reassembling step, we treat the extracted subgraphs as building blocks of molecular graphs and reassemble them in an autoregressive way by searching desired blocks according to the target property using reinforcement learning. Although MOLDR can employ other optimization methods such as Monte Carlo tree search (MCTS),[21,22] we consistently use reinforcement learning in our study as it is known to be effective in the context of molecular generation. We empirically evaluate molecular graphs generated by our method with respect to various well-established property scores, the penalized log P and Quantitative Estimation of Druglikeness (QED)[23] and multiobjective score of QED and Synthetic Accessibility (SA). In addition, we evaluate our method in the task of rediscovery of known drug molecules and show that our method is competitive with the state-of-the-art molecular generation methods, showing transition paths of generated molecules.

Our contributions are summarized as follows:

- Our method MOLDR explicitly constructs new molecules by combining substructures of molecules; hence, its generation process is highly interpretable.

- MOLDR can easily generate larger sizes of molecules out of distribution in a data set by combining subgraph structures.

- Molecules generated by MOLDR are superior to those by the current state-of-the-art generative models in terms of log P and QED (druglikeness).

■ RELATED WORKS

Yang et al.[24] and Olivecrona et al.[14] proposed SMILES generation approaches by RNNs and searched molecules with desired properties over SMILES representation using MCTS and policy gradients, respectively. Yang et al.[25] also proposed the massive parallel computation of MCTS to generate and search molecules. Instead of SMILES-based strategies, You et al.[20] proposed nodewise graph generation and property optimization using reinforcement learning. States of molecules are represented through graph convolutional networks, and the agent selects nodes, edges types, and the terminal to expand molecules. The policy is optimized through the Proximal Policy Optimization. These methods can generate valid molecules with desired properties at the final step. However, the generation process is a black-box by nature, and it is difficult to explain why and how such molecules are obtained.

Jin et al.[28] proposed a VAE model that generates junction trees over molecules. Nodes in a junction tree represent subgraphs extracted from a molecular data set, and a graph neural network determines which nodes or edges are combined with each other in the junction tree. To search molecules that optimize the desirable properties, it is necessary to search two vectors, what a tree structured scaffold is and how a molecule is reconstructed within the latent embedding space. In contrast, in our method, junction trees themselves are used to efficiently extract frequent substructures from a molecular data set, and we expand and search substructures directly to achieve target scores instead of generating junction trees from VAE.

Takeda et al.[13] proposed to generate molecules by combining substructures that contribute to the target properties, where candidate molecules are searched by McKay's Canonical Construction Path (MC-MCCP) algorithm.[26,27] Jin et al.[28] proposed the multiobjective molecule generation using interpretable substructures as rational for extracting substructures by MCTS to generate molecules by merging common substructures and graph completion. Although their approaches and our approach share the general strategy of constructing new molecules from its substructures, our method can cover a wider variety of substructures in molecular generation as we directly apply frequent subgraph mining to the entire molecular data set, which will lead to better new molecules.

Our approach of combining decomposition of molecules into subgraphs by graph mining and reassembling of subgraphs to generate new molecular graphs has not been studied at sufficient depth. There is a related approach in the task of planning of chemical synthesis, which also combines subgraphs and MCTS,[29] while it is not applied to the property optimization.

THE PROPOSED ALGORITHM: MOLDR

We introduce our molecular generation algorithm MOLDR. We provide the problem setting, tree decomposition preprocessing, graph mining, and the strategy to build up molecules via reinforcement learning.

Problem Setting. A *graph* is a tuple $G = (V, E)$, where V and E denote the set of nodes and edges, respectively. Nodes and edges can have labels (attributes) via label functions $l_V: V \to \Sigma_V$ for nodes and $l_E: E \to \Sigma_E$ for edges with some label domain Σ_V, Σ_E, which can be any set such as $\mathbb{Z}$ and $\mathbb{R}^d$. We assume that each molecule is represented as a graph. If we see a graph as a molecule, V is the set of atom types, and E is the set of bond types. For two graphs $G = (V, E)$ and $G' = (V', E')$, we say that G' is a *subgraph* of G, denoted by $G' \sqsubseteq G$, if $V' \subseteq V$ and $E' \subseteq (V' \times V') \cap E$.

Let $f(G)$ be some chemical property of a graph G, which is usually a real-valued function, and we assume that f is known beforehand and we can compute $f(G)$ for any graph G. For example, f can be the log P of a molecular G. Given a molecular data set, which is a collection of graphs, the problem of molecular generation is to explore a new graph G_{new} that has a high $f(G_{new})$ value as long as possible.

Graph Decomposition via Frequent Subgraph Mining. Given a collection of graphs as an input molecular data set, our idea is to apply frequent subgraph mining[30] to the data set, which finds all subgraphs that frequently appear in the graph data set. Formally, given a graph data set $\mathbf{D} = \{G_1, G_2, ..., G_n\}$ that contains n graphs, the objective of frequent subgraph mining is to find all subgraphs G satisfying the condition support$(G) \geq$ minsup, where support(G) is defined as

$$\text{support}(G) = \left| \{G_i \in \mathbf{D} \mid G \sqsubseteq G_i\} \right|$$

that is, the number of graphs in $\mathbf{D}$ that contain G as a subgraph, and minsup $\in \mathbb{N}$ is a frequency threshold.

We use the gSpan[31] algorithm, which is commonly used for the task of frequent subgraph mining. It enumerates subgraphs in a depth first manner. In gSpan, each graph is represented as the DFS code, which is constructed from a search tree based on a lexicographic order and enables us to efficiently check duplication of enumerated graphs. More precisely, for each explored graph during the enumeration, it checks whether or not its DFS code is canonical. After completion of gSpan, we check every enumerated subgraph and keep only subgraphs whose target property score is already higher than some threshold, which is determined beforehand, to efficiently reassemble them to construct new graphs in the next reassembling step.

Molecules are first converted into molecular graphs, where each node represents an atom type and each edge represents a bond type. However, if we directly apply gSpan to such molecular graphs, it gives a lot of invalid subgraphs in terms of molecules as building blocks for molecular generation. This is because gSpan does not know the chemical context and simply enumerates frequent subgraphs; hence, for example, the ring structure will be truncated by gSpan, while such truncated subgraphs are invalid and unnecessary for the reassembling step.

To circumvent this problem, we apply *tree decomposition* to molecular graphs as preprocessing before applying gSpan and convert them into molecular junction trees. A tree decomposition maps a graph $G = (V, E)$ into a *junction tree* $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{C_1, ..., C_n\}$ is a collection of subsets of V; that is, each $v_i \subseteq V$, and $\mathcal{E}$ is a set of edges between elements of $\mathcal{V}$. A junction tree satisfies the following properties:

1. The union of all sets $C_1, ..., C_n$ equals to V; that is, $\cup_i C_i = V$.
2. For every edge $(u, v) \in E$, there exists $C_i \in \mathcal{V}$ such that $u \in C_i$ and $v \in C_i$.
3. If C_k is on a path from C_i to C_j in $\mathcal{T}$, $V_i \cap V_j \subseteq V_k$.

By converting a graph into its corresponding junction tree, by definition, each cycle will be gathered as a single node, and all cycles will be eliminated. Therefore, if we apply gSpan not to the original graphs but instead to the converted junction trees, we can avoid enumerating invalid subgraphs in which the ring structure of a molecule, represented as a cycle on a graph, is truncated. In addition, gSpan on junction trees can dramatically reduce the number of frequent subgraphs. This is also an advantage of using junction trees in the decomposition step for molecular generation.

The edge label information and the node label information in each clique are lost in a junction tree; hence, we need to restore them after frequent subgraph mining. To achieve this task, we use a subgraph matching algorithm that matches between original graphs and obtained trees. We use the indexed based subgraph matching algorithm with general symmetries (IS-MAGS).[32] Since the size of each molecule is usually not so large and the number of nodes is mostly around 20−30 in the task of molecular generation, this restoring process is not computationally expensive.

Graph Reassembling from Frequent Subgraphs. Now we generate new molecules by reassembling frequent subgraphs obtained by the previous graph decomposition step. In contrast to our approach using subgraphs as building blocks, existing approaches are based on either text generation or nodewise graph generation. In the text generation approach[33] based on SMILES, an algorithm picks up a particular character which denotes the chemical state, such as the atom (C, N, O, F, ...), the bond type ($=$, $\equiv$), or the branched symbols, from the set of character types occurred in a training data set to generate and expand molecules. In the nodewise graph generation,[20,34] an algorithm selects a node (atom symbol) and the edge type between source and target atoms from the candidate set of atom and edge types. Our method can be more powerful and efficient as we directly combine subgraphs that already have desirable properties as building blocks in molecular generation.

To assemble molecular subgraphs, we pick up two graphs G_t and G'_t from building blocks and combine them to generate a new graph G_{t+1}, where t is the number of building up steps of molecules. As an example of such molecules, *2-acetyl-5-methylpyridine* and *naphthalene* are shown in Figure 1. Let us
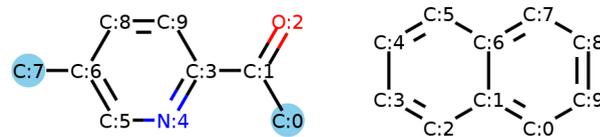

Figure 1. An example of molecular graphs G and G'. Atoms of C:0 and C:7 are candidates to merge.

assume that $G_t = (V(G_t), E(G_t))$ with $V(G_t) = \{v_1, ..., v_n\}$ and $G'_t = (V(G'_t), E(G'_t))$ with $V(G'_t) = \{u_1, ..., u_{n'}\}$. In the reassembling procedure with nodes, we select single nodes $v_i \in V(G_t)$ and $u_j \in V(G'_t)$ such that they have the same node labels: $l_v(v_i) = l_v(u_j)$. We overlay these two nodes as v_{t+1}; that is, $V(G_{t+1}) = V(G_t) \cup V(G'_t) \setminus \{v_i, u_j\} \cup \{v_{t+1}\}$ for a newly constructed graph G_{t+1}. All edges in G_t and G'_t are preserved in G_{t+1}, where if there is an edge
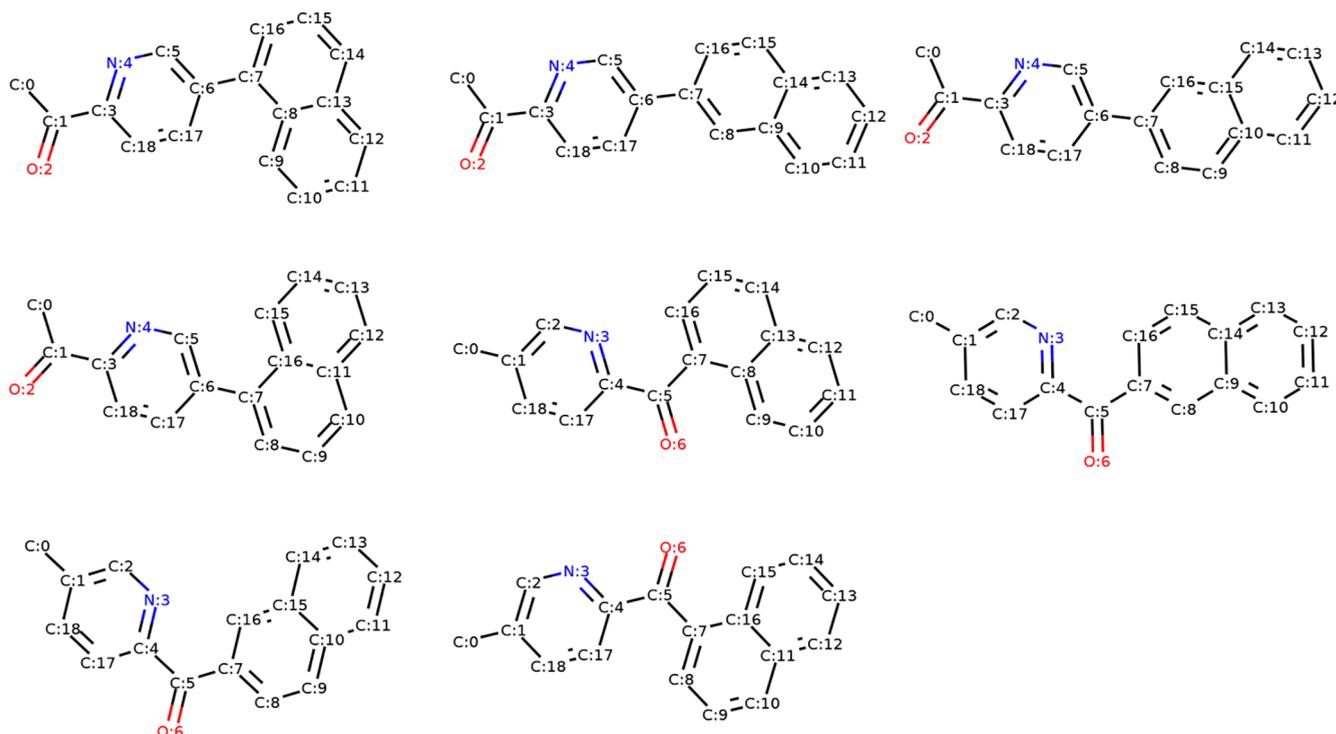
Figure 2. Reassembling two molecules with nodes in Figure 1. It shows merging with nodes labeled as C. In this example, reassembled molecules are sanitized to be valid molecules.
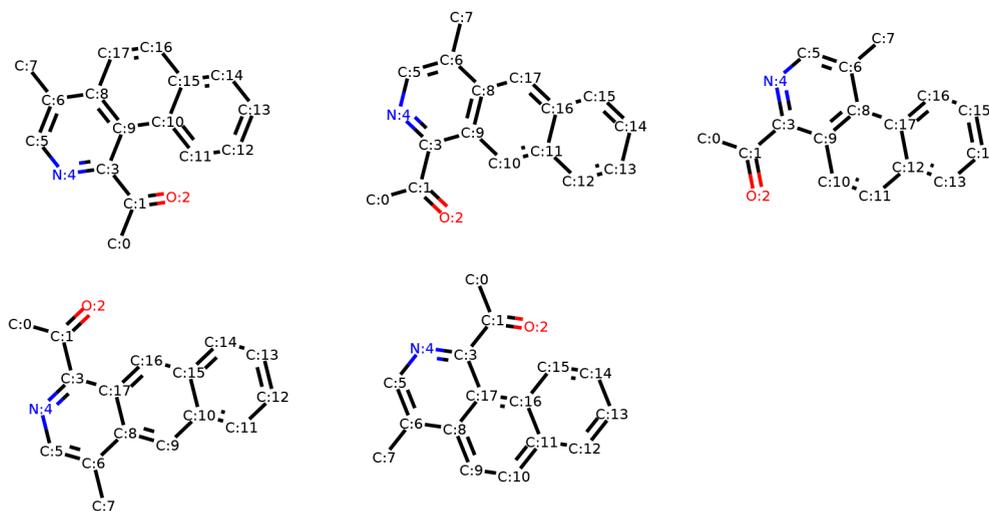

Figure 3. Reassembling two molecules with edges in Figure 1. Molecules were merged between edges in rings.

(v_i, v_k) or (u_j, u_l), it is replaced with (v_{t+1}, v_k) or (v_{t+1}, u_l). In the reassembling with edges, we select edges from rings and overlay them in the same manner as the assembling with edges.

This assembling is similar to reconstructing a graph from a junction tree; that is, nodes of a clique in a junction tree have intersected nodes that are connected with each other between subgraphs. Assembling two graphs is equivalent to choosing the intersection of nodes or edges. Figures 2 and 3 show the process of reassembling the two molecular graphs with nodes or edges, respectively. The candidate set of node label C for merging is {C:0: {C:0, C:2, C:3, C:4, C:5, C:7, C:8, C:9}, C:7: {C:0, C:2, C:3, C:4, C:5, C:7, C:8, C:9}}, where indices of nodes correspond to numbers in the illustration in Figure 1. We do not include internal nodes such as C:1, C:6 as the resulting graph

will be an invalid molecule nor include duplicated structures. In the reassembling with edges, the candidate set in the same node label (C, C) is {(C:8, C:9): {(C:0, C:9), (C:2, C:3), (C:3, C:4), (C:4, C:5), (C:8, C:9)}}, resulting in five new graphs as Kékule structures, and (C:0, C:9) and (C:7, C:8) are also the same due to symmetry structure.

The computational cost of combining two graphs depends on the number of nodes and the number of edges in rings. In the worst case, where we need to consider all combinations of nodes and edges of two graphs $G = (V, E)$ and $G' = (V', E')$, the complexity becomes $O(|V||V'| + |E||E'|)$. However, this can usually be reduced in practice by considering the symmetrical structure of a graph and some type of restrictions of chemical valency of an element in the case of molecules. We always check
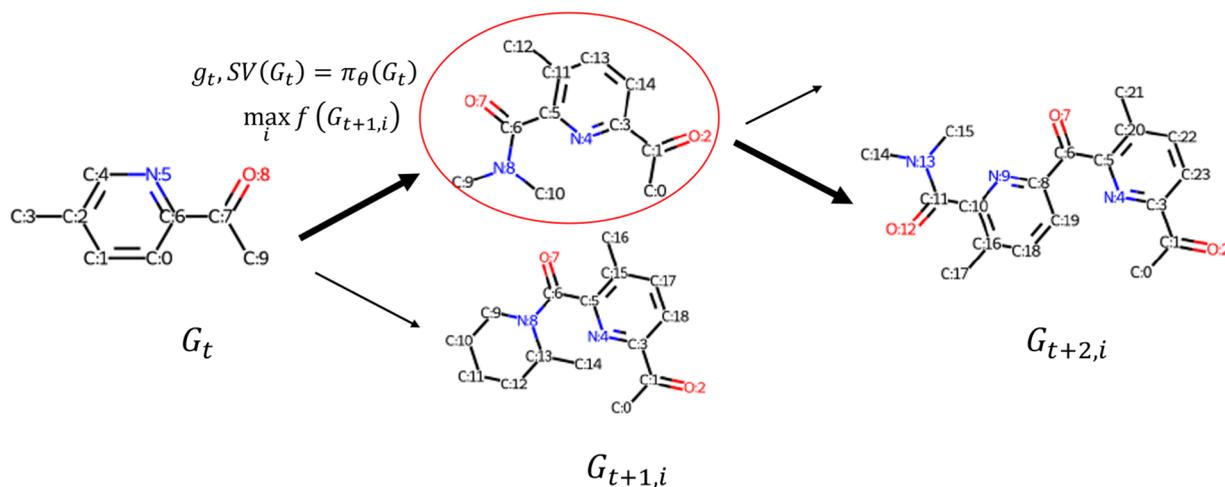
Figure 4. Diagram of molecular generation in the reassembling step. Mined subgraph structures are selected based on the policy network π_θ. After reassembling molecules, a new graph is selected based on the highest score of a target property computed from reward function r.

such conditions whenever a new molecule is generated and remove it if does not satisfy such conditions. Therefore, molecules generated by our method are always valid.

Finding Candidate Subgraphs by Reinforcement Learning. To efficiently find subgraphs that will lead to desirable molecules when assembled in graph generation, we use reinforcement learning. Another potential choice of a searching method is Monte Carlo tree search (MCTS), which is a search method that combines tree search with random sampling.[35] MCTS has been applied to a number of tasks when the search space is massive and achieved huge success in various fields such as the game of Go[36] and the planning of chemical syntheses.[29] However, the parallel computation of MCTS is difficult to implement, so we exploit the reinforcement learning and proximal policy optimization (PPO) that can be used for continuous control. In reinforcement learning, the agent takes action a based on a policy π, which is often represented as neural networks. The policy network returns the probability of each action $a \in \mathcal{A}$ and the state-value function V_π based on the state $s \in \mathcal{S}$. In other words, at a time step t, the action a_t is sampled with probability $V_{\pi_\theta}(s_t) = \pi_\theta(s_t|a_t)$. The agent is trained so that the expected cumulative reward $\mathbb{E}[\sum_{t=0}^{\infty} r_t]$ is maximized while interacting with the environment.

PPO is based on the trust region policy optimization (TRPO) method[37] to prevent the high variance in learning with the policy gradient.[38] The main objective is to optimize the parameter θ of a policy network π through the loss function L as the following:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t[\min(x_t(\theta)\hat{A}_t, \text{clip}(x_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)], \text{ where}$$

$$x_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)},$$

$$\hat{A}_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - \text{SV}_{\pi_\theta}(s_t)$$

where ϵ is a hyperparameter, γ is a discount factor, and SV_π is the state-value function denoting the expected return computed from the policy network. The function $\text{clip}(\cdot, 1-\epsilon, 1+\epsilon)$ clips the value of the first argument within the range from $1 - \epsilon$ to $1 + \epsilon$. The first term inside the min is conservative policy iteration.[39] The second term modifies the surrogate objective by clipping

the probability ratio, which removes the incentive for moving r_t outside of the interval $[1 - \epsilon, 1 + \epsilon]$.[38]

States Space. A graph G_t is mapped into F-dimensional continuous space in the form of a node feature matrix $H_t \in \mathbf{R}^{|V| \times F}$ converted by Mol2vec,[40] which is based on word2vec[41] and designed for molecular substructures obtained through Morgan fingerprints. Each computed node feature in a graph is reduced via the sum function over nodes into a single F-dimensional vector $s_t \in \mathbf{R}^F$, which incorporates graph topological information.

Action Space. Action space $\mathcal{A}$ is equivalent to the set of subgraphs enumerated from a subgraph mining algorithm. Each action $a \in \mathcal{A}$ is sampled according to softmax of the likelihood based on the state-value function $\text{SV}(s_t)$ and the expected cumulative reward.

Rewards. To generate molecules with target properties, we assume that target properties satisfy the additive composition-ality of subgraphs. Therefore, if some subgraph structure is not related to the target property, it is not selected as a building block. This assumption can be simply represented in the following strategy: If the difference of rewards at a time step t is negative, we stop the trial. The reward r_t is computed by a scoring function $f(G_t)$ at time step t. If the reward is below a certain threshold of the score, further search will be stopped (Figure 4).

■ EXPERIMENTS

We empirically examine the effectiveness of our proposed method MOLDR compared to the state-of-the-art molecular generation methods. In particular, first we examine the standard criteria, the penalized log P and the druglikeness score QED, of generated molecules. In addition, we also examine the multiobjective score of QED and SA. Furthermore, we benchmark the rediscovery molecules using the GuacaMol benchmark data set.[42]

All methods are implemented in Python 3.7.6. We used the gSpan library (https://github.com/betterenvi/gSpan) to obtain building blocks. All neural networks and reinforcement learning are implemented in RLlib[43] and PyTorch.[44] All experiments were conducted on Ubuntu 18.04.5LTS with 40 cores of 2.2 GHz Intel Xeon CPU E5-2698 v4, 256 GB of memory, and 32 GB Nvidia Tesla V100.

Data Set. We use the ZINC molecular data set and GuacaMol data set with about 1.5m molecules preprocessed with the ChEMBL data set. The ZINC data set is a freely available druglike molecular database.[45] There are 249 456 molecules in total, and the maximum numbers of nodes and edges are 38 and 45, respectively. The number of node labels is 8: {B, C, F, I, N, O, P, S}. The number of molecules on the GuacaMol data set is 1 591 378 in total, and the maximum numbers of nodes and edges are 88 and 87. The number of node labels is 12: {B, Br, C, Cl, F, I, N, O, P, S, Se, Si}. All molecules are prepossessed by RDKit[46] so that they are treated as graphs.

Before applying MOLDR, we converted molecules in the molecular data set into *junction trees*. As a result, on the ZINC data set, the number of cliques is 784, which are used as the node labels of junction trees. The maximum numbers of nodes and edges in junction trees become 31 and 30, respectively. On the GuacaMol data set, the number of cliques is 5106, and the maximum numbers of nodes and edges in junction trees become 88 and 87, respectively.

Experimental Setting. The gSpan algorithm was applied to converted junction trees with the minimum support of 10 000, 5 000, 1 000, and 100, where we enumerated molecules with more than seven nodes. To see the effectiveness of our junction tree-based enumeration, we also applied gSpan to the original ZINC data set without junction tree conversion. On the GuacaMol data set, we applied gSpan with a minimum support of 10 000.

In the molecular reassembling step in MOLDR, we use building blocks extracted under the condition of the minimum support of 1000. In reinforcement learning, we use the policy network with three-layer MLPs (256, 128, and 128 hidden units) to take actions (to choose building blocks and compute state-value functions), and the activation function is the ReLu function. The generalized advantage estimate (GAE) parameters are set as $\lambda = 1.0$ and $\gamma = 0.99$. The optimizer is stochastic gradient descent, where the minibatch size is 128 and the learning rate is 5.0×10^{-5}. The terminal condition is when the maximum number of nodes exceeds 100, or the previous reward exceeds the current reward or threshold.

Target Properties. As target chemical properties, we employ scores of the *penalized* log P[16] and QED.[23] These values are widely used as a benchmark for the task of a molecular generation. The penalized log P is a logarithm of the octanol−water partition coefficient with restrictions on the ring size and synthetic accessibility (SA).[47] The SA score is defined as follows:

$$\text{SA} = \text{fragment score} - \text{complex penalty}$$

The fragment score was introduced to capture the "historical synthetic knowledge" by analyzing common structural features in a large number of already synthesized molecules.[47] The complex penalty is computed from summation of each term as follows:

$$\text{ring complexity} = \log(n\text{RingBridgeAtoms} + 1) + \log(n\text{SpiroAtoms} - 1)$$

$$\text{stereo complexity} = \log(n\text{StereoCenters} + 1)$$

$$\text{macro cycle penalty} = \log(n\text{MacroCycles} + 1)$$

$$\text{size penalty} = n\text{Atoms}^{1.005} - n\text{Atoms}$$

where "n" denotes the number. We used the penalized log P normalized with the ZINC250k data set to compare the same setting with other methods; thus, direct comparison of scores is fair. QED is the score representing the druglike nature of molecular structures. QED represents the function of weighted chemical properties:

$$\text{QED} = \exp\left(\frac{\sum w_i \log d_i}{\sum w}\right)$$

where w is the weight of a molecule and each d_i is one of the following chemical properties: molecular weight (MW), octanol−water partition coefficient (ALOGP), number of hydrogen bond donors (HBD), number of hydrogen bond acceptors (HBA), molecular polar surface area (PSA), number of rotatable bonds (ROTB), the number of aromatic rings (AROM), or number of structural alerts (ALERTS). Thus, optimizing QED implies generating the molecules subject to these parameters. For guiding target values of properties, we optimize the molecule with log $P = 8.0$. For the multiobjective benchmark, we generate molecules such that QED is higher and SA is smaller (easy to synthesize). We choose the objective function proposed by Tan et al.[48] defined as follows:

$$f(G) = \max(\text{QED}(G) - 0.1\text{SA}(G))$$

Distribution Benchmarks. To investigate whether MOLDR can generate diverse molecules or not, we use the GuacaMol benchmark data set. The proposed scores are listed as follows:

- Validity: whether the generated molecules are actually valid computed in RDKit.
- Uniqueness: the ratio of molecules that are not duplicated to generate molecules.
- Novelty: the ratio of molecules that are not duplicated to the original data set.
- Kullback−Leibler (KD) divergence: measures how well a probability distribution Q approximates another distribution P: $D_{\text{KL}} = \sum_i P(i) \log \frac{P(i)}{Q(i)}$. The probability is calculated from physiochemical descriptors for the training set and the generated set.
- Fréchet ChemNet distance (FCD). Preuer et al.[49] introduced the Fréchet ChemNet distance as a measure of how close distributions of generated data are to the distribution of molecules in the training set. Low FCD values characterize similar molecule distributions

Rediscovery Molecules. We examined whether or not MOLDR can reconstruct target molecules such as drugs and can generate molecules exceeding some threshold of similarity between molecules, not just generating molecules with chemical properties. In such cases, we chose celecoxib, troglitazone, and thiothixene as rediscovery benchmarks, aripiprazole as a similarity benchmark, and ranolazine and osimertinib as MPO benchmarks.

■ RESULTS AND DISCUSSION

Table 1 shows results of applying gSpan to the ZINC database with varying the minimum support. We compare the number of obtained subgraphs and calculation time with or without molecular junction trees. We can see that enumeration based on junction trees is much faster than directly applying gSpan to molecular graphs. This result means that our junction tree-based enumeration is effective in the real-world ZINC database. Note

Table 1. Comparison of Frequent Subgraph Enumeration with or without Junction Trees

minsup	number of mined trees	number of mined graphs[a]
100 000	0	23 (1334 s)
10 000	8 (164.42 s)	4040 (106.5 min)
5 000	39 (216.21 s)	—
1 000	910 (342.20 s)	—
100	23 616 (775.23 s)	—

[a]"—" means that computation did not stop in 2 h.

that we can fully recover original subgraphs from mined trees using subgraph matching as we have discussed; thus, there is no information loss in the junction tree-based enumeration, and it can be viewed as loss-less compression of subgraphs. When the minimum support is 100, we could find a large number of subgraphs (23 616 subgraphs), and it is expected that we have collected enough substructures. Therefore, we stop decreasing the minimum support. Figure 5 shows examples of building blocks of substructures extracted from the ZINC 250k filter by the score of QED > 0.7. The obtained substructures are frequent subgraphs with minimum support of 100. These structures are used as building blocks for molecular graph reassembling.

Table 2 shows the top three generated molecules according to property scores of the penalized log P or QED. Scores of other methods come from the literature.[20,50] MOLDR is similar to the technique of JT-VAE as both methods use junction trees, while MOLDR outperforms both scores. The log P is related to the lipophilicity and hydrophilicity of a molecule. Hence, if nodes in a generated molecule have many carbon (C) and less imide (= NH) or hydroxyl groups (OH), the resulting log P becomes high. It means that the larger the number of the atom C is, the higher the log P value is. At the same time, we show penalized log P scores in which the ring size and synthetic accessibility are penalized in Table 2. In the case of penalized log P optimization, an approach of greedy search such as selecting only C can be

enough to maximize the score because the calculation of the log P score consists of additive compositionality. MOLDR can train such a strategy by optimizing the penalized log P (the molecule with the top score has only C (C43)) as shown in Figure 6a. The QED score is empirically derived from the combination of various chemical properties and chemical structures. Hence, it is not straightforward to maximize QED, unlike the case of log P. Nevertheless, MOLDR outperforms the score of JT-VAE and the top-1 and -2 molecules generated by GCPN. To increase the QED score, generated molecules need to follow the strict restriction of structures. Figure 6b illustrates examples of generated molecules with optimization of QED by MOLDR. In penalized log P optimization, when the molecular size becomes larger and molecules include a large number of C, the resulting log P increases. In QED optimization, the size of the molecule is smaller than in the case of log P optimization, and they have subgraphs that contribute to the QED. We remove the similar graphs with the highest score in Figure 6 to show the variety of generated molecules by our method.

Moreover, MOLDR is flexible in the sense that it can generate molecules with not only maximizing the target value like log P but also controlling it to be a specific value. As an example, we show molecules generated by MOLDR with specifying the target value log P = 8.0 in Figure 7.

In a multiobjective task with both QED and SA, Figure 8 shows results of generated molecules when only QED is optimized or both QED and SA are optimized. If only QED is optimized (Figure 8a and b), generated molecules tend to have a higher QED score with the SA score being around 3−5. Otherwise, if both QED and SA are optimized, the distribution of SA shifts left (Figure 8c and d) compared to the case of QED optimization.

Results of distribution benchmarks are shown in Table 3. This benchmark evaluates whether or not a model can generate valid, unique, and novel molecules from a training data set. The KL (Kullback−Leibler) divergence and the Fréchet ChemNet
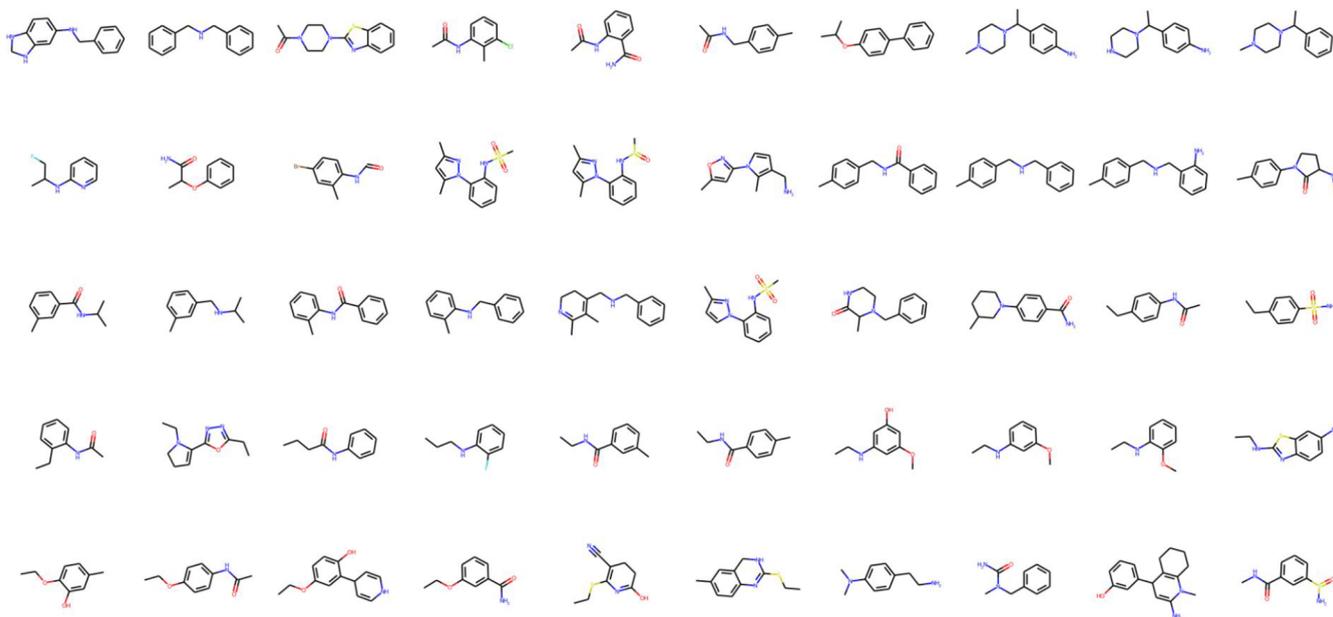

Figure 5. Examples of extracted substructures sorted by the score of QED. ZINC 250k molecules are decomposed into junction trees, gSpan enumerates frequent subtrees, and they are reconstructed into molecules by ISMAGS. These substructures become building blocks for molecular reassembling.

Table 2. Comparison of the Top Three Property Scores of Generated Molecules[a]

| method	penalized log P				QED			
	first	second	third	validity	first	second	third	validity
ZINC	4.52	4.30	4.23	100.0%	0.948	0.948	0.948	100.0%
ORGAN	3.63	3.49	3.44	0.4%	0.838	0.814	0.814	2.2%
JT-VAE	5.30	4.93	4.49	100.0%	0.925	0.911	0.910	100.0%
GCPN	7.98	7.85	7.80	100.0%	**0.948**	0.947	0.946	100.0%
GraphAF	12.23	11.29	11.05	100.0%	**0.948**	**0.948**	**0.947**	100.0%
MOLDR	**12.46**	**12.20**	**12.04**	**100.0%**	0.948	0.948	0.947	**100.0%**

[a]Scores for ORGAN, JT-VAE, and GCPN are from refs 20 and 50 on the ZINC data set.
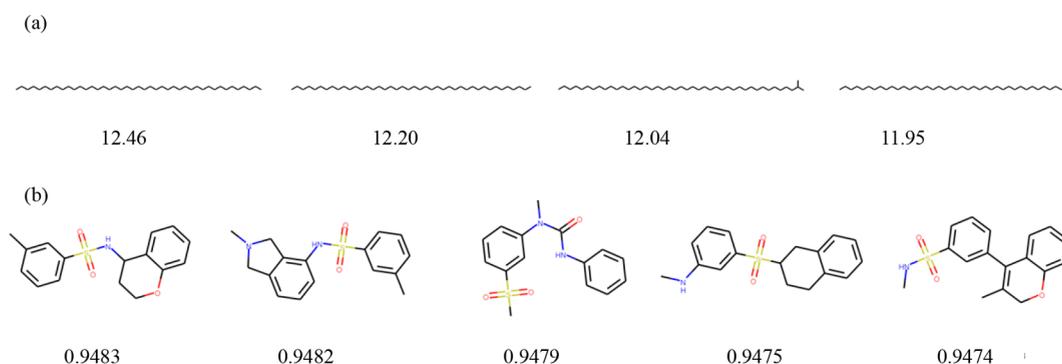
Figure 6. Generated molecules based on ZINC data set by MOLDR with penalized log P and QED scores.
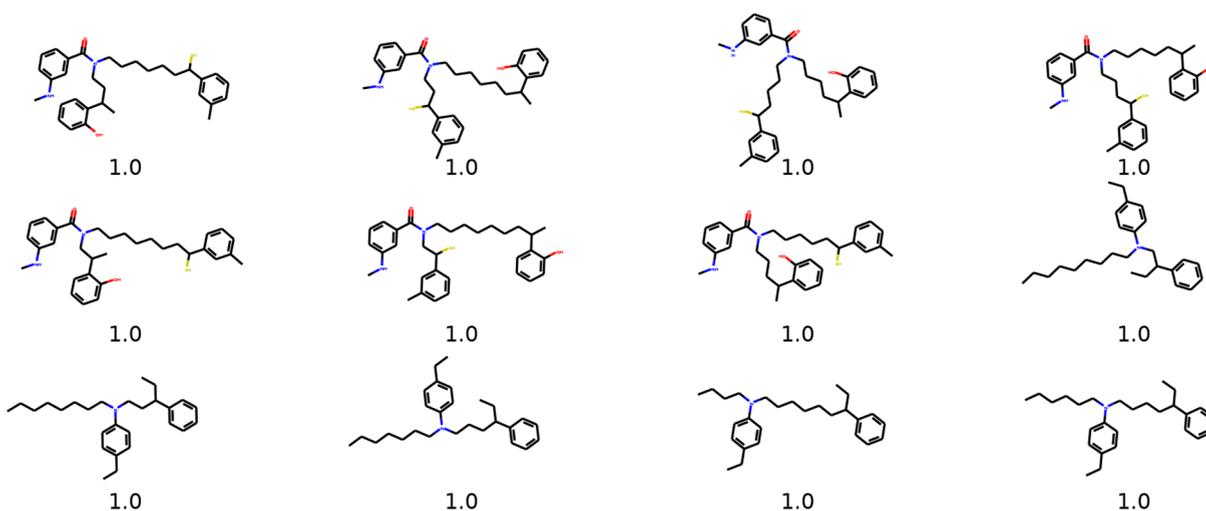
**Figure 7.** Generated molecules with log P = 8.0.

distance (FCD) between a training set and generated molecules are also used. In a decomposition step, 1 709 building blocks are mined from the GuacaMol data set with minsup = 10 000. The distribution benchmark is evaluated from 10k sampled molecules in a reassembling step. MOLDR can generate valid molecules due to reassembling a building block of molecules. Although the uniqueness is slightly smaller than in other models, MOLDR depends on random seeds to choose building blocks. In terms of the KL divergence and the FCD, MOLDR is inferior to SMILES LSTM and VAE. However, the score is similar to that of Graph MCTS because it is also a similar strategy to generate molecules. In addition, MOLDR can sample molecules randomly from an untrained policy network. Hence, MOLDR has a potential to generate molecules that are largely different from those in the training data set, leading to lower scores of the

KL divergence and the FCD. In order to improve the performance in terms of the KL divergence and the FCD, MOLDR would need to train the policy network and design an appropriate reward function, such as the similarity between the training data set and generated molecules.

Figure 9 shows results of the distribution of generated molecules when trained on the ZINC or the GuacaMol data set. Molecules are first mapped into 300-dimensional vectors using Mol2Vec, and t-distributed stochastic neighbor embedding (t-SNE) is applied to visualize the distribution of generated molecules and that of the training set. On the ZINC data set, generated molecules are mostly overlapped within the training set, while on the GuacaMol data set, the distribution of generated molecules goes beyond that of the training set. This means that the generated molecules are not similar to the
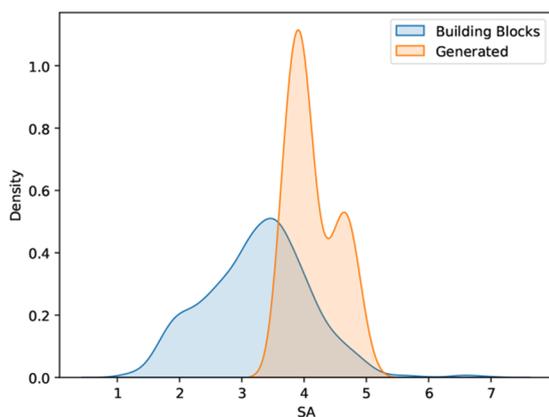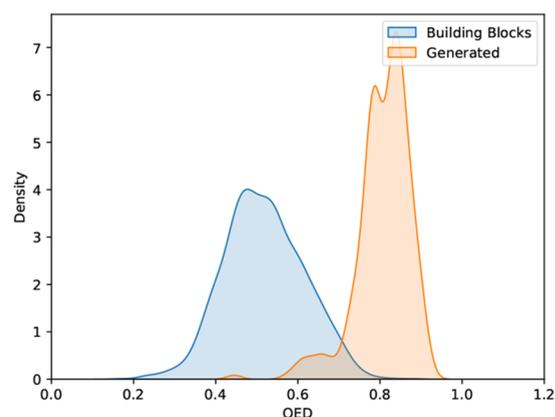
(a) SA distribution of building blocks and generated molecules when QED is optimized.
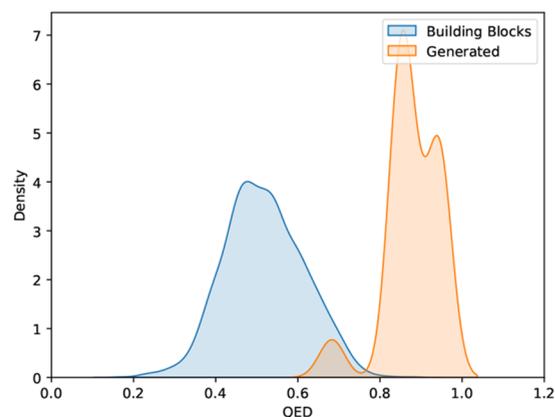
(b) QED distribution of building blocks and generated molecules when QED is optimized.
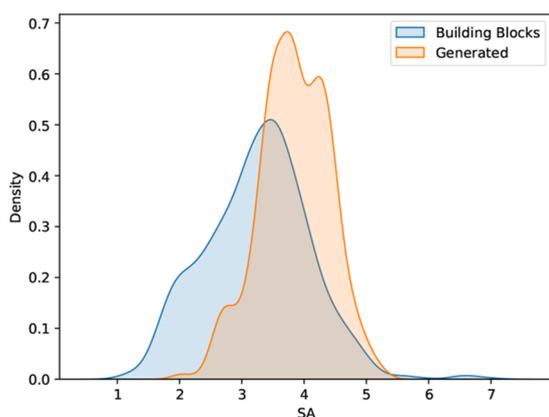
(c) SA distribution of building blocks and generated molecules when both QED and SA are optimized simultaneously.

(d) QED distribution of building blocks and generated molecules when both QED and SA are optimized simultaneously.

Figure 8. Distribution of generated compounds for optimizing both QED and SA in the multiobjective task on the GuacaMol data set.

Table 3. Distribution Benchmarks at 10k Molecules on the GuacaMol Data Set

| benchmark	random sampler	graph MCTS	SMILES LSTM	VAE	**MOLDR** (random)
validity	1.00	**1.000**	0.959	0.870	**1.000**
uniqueness	0.997	**1.000**	**1.000**	0.999	0.994
novelty	0.000	0.994	0.912	0.974	**0.996**
KL divergence	0.998	0.522	**0.991**	0.982	0.442
Fréchet ChemNet distance	0.929	0.015	**0.913**	0.863	0.029

training set; therefore, the KL divergence and FCD scores are likely to be lower. However, random sampling from MOLDR can generate larger molecules out of distribution.

Table 4 shows the result of rediscovery benchmarks. MOLDR can generate the target molecules with high accuracy, whose scores are competitive with SMILES LSTM and Graph GA. The most notable difference between those models is that our model can visualize the generating process of molecules, not just generating the target molecule. Although in SMILES LSTM,

intermediate molecules are evaluated from the state-value function, and the SMILES character is selected based on the state, it is not easy to interpret why the character is vital at a particular time step, especially when generating a ring. In contrast, in MOLDR, building blocks are directly selected, and the substructure affects the target directly. The generating process is shown in Figures 10 and 11. Since the generation performance of MOLDR depends on the building blocks obtained from graph mining, in practical applications, it is important to prepare an appropriate data set and set an appropriate minimum support based on a priori knowledge.

■ CONCLUSION

We have proposed a new molecular generation method, called MOLDR, which decomposes graph structures and reassembles them. In our experiments on the ZINC database, MOLDR can find better molecules in terms of two properties, the penalized log P and the druglikeness score QED, than the state-of-the-art molecular generation methods using generative models and reinforcement learning. In terms of GuacaMol benchmarks, MOLDR can also reconstruct the target molecule if the
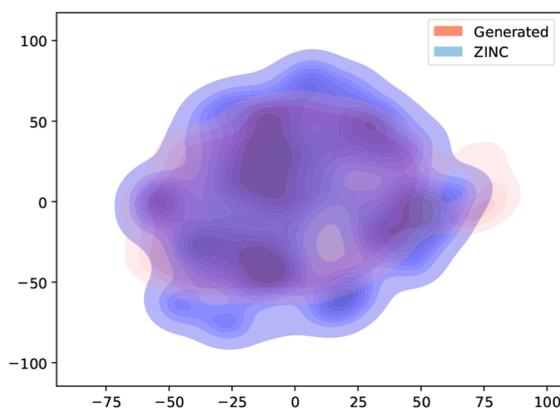
(a) Distribution on ZINC dataset
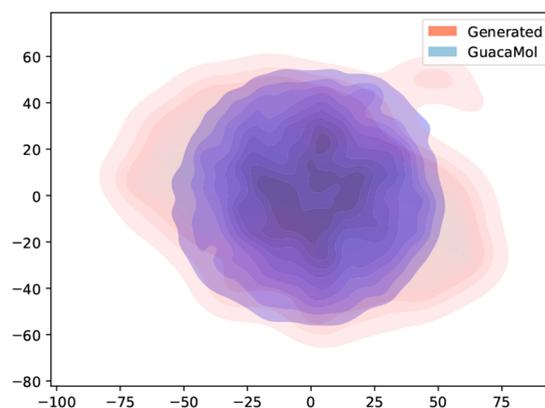
(b) Distribution on GuacaMol dataset

Figure 9. Distributions of training sets in the GuacaMol and ZINC data sets and generated molecules. Molecules are mapped into vectors using Mol2vec, and then we apply t-SNE dimensionality reduction for visualization.

Table 4. Goal Directed Benchmarks

benchmark	best in data set	SMILES LSTM	graph GA	MOLDR
celecoxib rediscovery	0.505	1.000	1.000	1.000
troglitazone rediscovery	0.419	1.000	1.000	1.000
aripiprazole similarity	0.595	1.000	1.000	1.000
osimertinib MPO	0.839	0.907	0.953	0.898
ranolazine MPO	0.792	0.855	0.920	0.864

substructures exist. Our approach is general; hence, it can also be applied to any graph generation problem as well as molecular graph generation. MOLDR can also incorporate a priori knowledge about substructures by selecting specific data sets and/or designing reward functions.

There are various prospective directions as our future work. First, it is interesting to explore clustering of subgraphs extracted by graph mining as graph mining tends to generate many similar graphs, and they are often redundant. Second, since the graph construction step is interpretable in MOLDR, it may be possible to understand chemical reaction in the retrosynthesis analysis by incorporating MOLDR since it can generate a path of graph generation steps. Third, our approach can be extended to
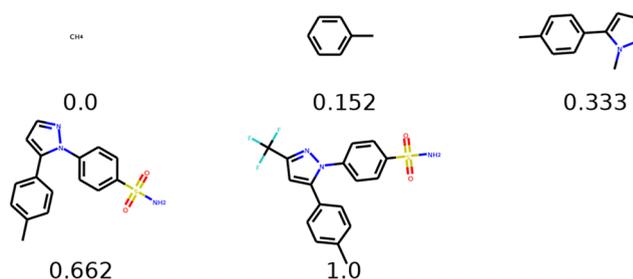

Figure 11. Generating process on celecoxib rediscovery.

generation of 3D molecules by designing the reward function that can take the molecular geometry into account when reassembling substructures.

■ ASSOCIATED CONTENT

Data Availability Statement

The open-sourced codes and data for MOLDR are available at https://github.com/Masatsugar/graph-decomposition-reassembling.
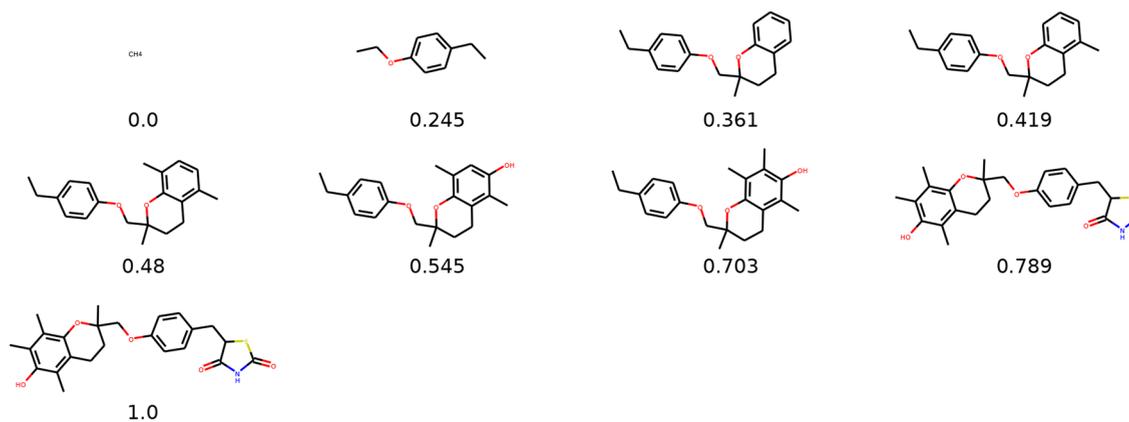

Figure 10. Generating process of troglitazone rediscovery. The number under the molecules denotes the similarity score between a generated molecule and the target. MOLDR can generate troglitazone in eight steps.

AUTHOR INFORMATION

Corresponding Authors

Masatsugu Yamada — *School of Multidisciplinary Sciences, Department of Informatics, The Graduate University for Advanced Studies, SOKENDAI, Kanagawa 240-0115, Japan; National Institute of Informatics, Tokyo 101-8430, Japan; Innovative Technology Laboratories, AGC Inc., 230-0045 Kanagawa, Japan;* ⊙ orcid.org/0000-0001-9469-0943; Email: masatsugu-yamada@nii.ac.jp

Mahito Sugiyama — *School of Multidisciplinary Sciences, Department of Informatics, The Graduate University for Advanced Studies, SOKENDAI, Kanagawa 240-0115, Japan; National Institute of Informatics, Tokyo 101-8430, Japan;* Email: mahito@nii.ac.jp

Complete contact information is available at:
https://pubs.acs.org/10.1021/acsomega.3c01078

Notes
The authors declare no competing financial interest.

ACKNOWLEDGMENTS

REFERENCES

(1) Polishchuk, P. G.; Madzhidov, T. I.; Varnek, A. Estimation of the size of drug-like chemical space based on GDB-17 data. *Journal of Computer-Aided Molecular Design* **2013**, *27*, 675−679.

(2) Kirkpatrick, P.; Ellis, C. Chemical space. *Nature* **2004**, *432*, 823.

(3) David, L.; Thakkar, A.; Mercado, R.; Engkvist, O. Molecular representations in AI-driven drug discovery: a review and practical guide. *J. Cheminform.* **2020**, *12*, 56.

(4) Kwon, S.; Bae, H.; Jo, J.; Yoon, S. Comprehensive ensemble in QSAR prediction for drug discovery. *BMC Bioinformatics* **2019**, *20*, 521.

(5) Wong, W. W.; Burkowski, F. J. A constructive approach for discovering new drug leads: Using a kernel methodology for the inverse-QSAR problem. *J. Cheminform.* **2009**, *1*, 4.

(6) Miyao, T.; Kaneko, H.; Funatsu, K. Inverse QSPR/QSAR Analysis for Chemical Structure Generation (from y to x). *J. Chem. Inf. Model.* **2016**, *56*, 286−299.

(7) Churchwell, C. J.; Rintoul, M. D.; Martin, S.; Visco, D. P.; Kotu, A.; Larson, R. S.; Sillerud, L. O.; Brown, D. C.; Faulon, J.-L. The signature molecular descriptor: 3. Inverse-quantitative structure-activity relationship of ICAM-1 inhibitory peptides. *Journal of Molecular Graphics and Modelling* **2004**, *22*, 263−273.

(8) Hemmer, M. C.; Steinhauer, V.; Gasteiger, J. Deriving the 3D structure of organic molecules from their infrared spectra. *Vib. Spectrosc.* **1999**, *19*, 151−164.

(9) Lorits, E. M.; Gubar, E. A.; Novikov, A. S. Design of the Algorithm for Packaging of Water Molecules in a Fixed Volume. *Symmetry* **2022**, *14*, 2453.

(10) Kuzminykh, D.; Polykovskiy, D.; Kadurin, A.; Zhebrak, A.; Baskov, I.; Nikolenko, S.; Shayakhmetov, R.; Zhavoronkov, A. 3D Molecular Representations Based on the Wave Transform for Convolutional Neural Networks. *Mol. Pharmaceutics* **2018**, *15*, 4378−4385.

(11) Gebauer, N. W. A.; Gastegger, M.; Hessmann, S. S. P.; Müller, K.-R.; Schütt, K. T. Inverse design of 3d molecular structures with conditional generative neural networks. *Nat. Commun.* **2022**, *13*, 973.

(12) Xu, M.; Huang, W.; Xu, M.; Lei, J.; Chen, H. 3D Conformational Generative Models for Biological Structures Using Graph Information-Embedded Relative Coordinates. *Molecules* **2023**, *28*, 321.

(13) Takeda, S.; et al. Molecular Inverse-Design Platform for Material Industries. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; 2020; pp 2961−2969.

(14) Olivecrona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular de-novo design through deep reinforcement learning. *J. Cheminform.* **2017**, *9*, 48.

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

(16) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder. *arXiv:1703.01925* **2017**, .

(17) Guimaraes, G. L.; Sanchez-Lengeling, B.; Farias, P. L. C.; Aspuru-Guzik, A. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv:1705.10843* **2017**, DOI: 10.48550/arXiv.1705.10843.

(18) Jin, W.; Barzilay, R.; Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden,* July 2018; Vol. 80, pp 2328−2337

(19) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31−36.

(20) You, J.; Liu, B.; Ying, Z.; Pande, V.; Leskovec, J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. *Proceedings of the 32nd International Conference on Neural Information Processing Systems*; 2018; pp 6412−6422.

(21) Coulom, R. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. *Proceedings of the 5th International Conference on Computers and Games*, Berlin, Heidelberg, 2006; pp 72−83.

(22) Kocsis, L.; Szepesvári, C. Bandit Based Monte-Carlo Planning. In *Machine Learning: ECML 2006*; Lecture Notes in Computer Science 4212; Fürnkranz, J., Scheffer, T., Spiliopoulou, M., Eds.; Springer: Berlin, Heidelberg, 2006; Vol. *4212*, pp 282−293.

(23) Bickerton, G. R.; Paolini, G. V.; Besnard, J.; Muresan, S.; Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nat. Chem.* **2012**, *4*, 90−98.

(24) Yang, X.; Zhang, J.; Yoshizoe, K.; Terayama, K.; Tsuda, K. ChemTS: an efficient python library for de novo molecular generation. *Sci. Technol. Adv. Mater.* **2017**, *18*, 972−976.

(25) Yang, X.; Aasawat, T.; Yoshizoe, K. Practical Massively Parallel Monte-Carlo Tree Search Applied to Molecular Design. *International Conference on Learning Representations*; 2021.

(26) McKay, B. D. Isomorph-Free Exhaustive Generation. *Journal of Algorithms* **1998**, *26*, 306−324.

(27) Stephen, H. G.; Andrew, R. J. Mckay's canonical graph labeling algorithm. *Communicating Mathematics*; American Mathematical Society: Providence, RI, 2009; Vol. *479*, pp 99−111.

(28) Jin, W.; Barzilay, R.; Jaakkola, T. S. Composing Molecules with Multiple Property Constraints. *arXiv.2002.03244* **2020**, DOI: 10.48550/arXiv.2002.03244.

(29) Segler, M. H. S.; Preuss, M.; Waller, M. P. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature* **2018**, *555*, 604−610.

(30) Zaki, M. J.; Meira, W., Jr. *Data Mining and Analysis: Fundamental Concepts and Algorithms*; Cambridge University Press: Cambridge, 2014.

(31) Xifeng Yan, J. H. gSpan: Graph-Based Substructure Pattern Mining. *International Conference on Data Mining*; 2002; pp 721−724.

(32) Houbraken, M.; Demeyer, S.; Michoel, T.; Audenaert, P.; Colle, D.; Pickavet, M. The Index-Based Subgraph Matching Algorithm with General Symmetries (ISMAGS): Exploiting Symmetry for Faster Subgraph Enumeration. *PLoS One* **2014**, *9*, e97896.

(33) Segler, M. H. S.; Kogej, T.; Tyrchan, C.; Waller, M. P. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS central science* **2018**, *4*, 120−131.

(34) Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; Battaglia, P. Learning Deep Generative Models of Graphs. 2018; https://openreview.net/forum?id=Hy1d-ebAb.

(35) Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; Colton, S. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* **2012**, *4*, 1−43.

(36) Silver, D.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484−489.

(37) Schulman, J.; Levine, S.; Moritz, P.; Jordan, M. I.; Abbeel, P. Trust Region Policy Optimization. 2015; https://arxiv.org/abs/1502.05477.

(38) Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. 2017; https://arxiv.org/abs/1707.06347.

(39) Kakade, S.; Langford, J. Approximately Optimal Approximate Reinforcement Learning. *Proceedings of the Nineteenth International Conference on Machine Learning*, San Francisco, CA, 2002; pp 267−274.

(40) Jaeger, S.; Fulle, S.; Turk, S. Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition. *J. Chem. Inf. Model.* **2018**, *58*, 27−35.

(41) Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *Adv. Neural Inform. Process. Systems.* **2013**.

(42) Brown, N.; Fiscato, M.; Segler, M. H.; Vaucher, A. C. GuacaMol: Benchmarking Models for de Novo Molecular Design. *J. Chem. Inf. Model.* **2019**, *59*, 1096−1108.

(43) Liang, E.; Liaw, R.; Nishihara, R.; Moritz, P.; Fox, R.; Goldberg, K.; Gonzalez, J.; Jordan, M.; Stoica, I. RLlib: Abstractions for Distributed Reinforcement Learning. *Proc. MLR* **2018**, *80*, 3053−3062.

(44) Paszke, A.; et al. *Advances in Neural Information Processing Systems 32*; Curran Associates, Inc.: Red Hook, NY, 2019; pp 8024−8035.

(45) Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; Coleman, R. G. ZINC: A Free Tool to Discover Chemistry for Biology. *J. Chem. Inf. Model.* **2012**, *52*, 1757−1768.

(46) Landrum, G. *Rdkit: Open-source cheminformatics*; www.rdkit.org, 2006.

(47) Ertl, P.; Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. In *Dyes and Drugs*; Apple Academic Press: New York, 2011.

(48) Tan, Y.; Dai, L.; Huang, W.; Guo, Y.; Zheng, S.; Lei, J.; Chen, H.; Yang, Y. DRlinker: Deep Reinforcement Learning for Optimization in Fragment Linking Design. *J. Chem. Inf. Model.* **2022**, *62*, 5907.

(49) Preuer, K.; Renz, P.; Unterthiner, T.; Hochreiter, S.; Klambauer, G. Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery. *J. Chem. Inf. Model.* **2018**, *58*, 1736−1741.

(50) Shi, C.; Xu, M.; Zhu, Z.; Zhang, W.; Zhang, M.; Tang, J. GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation. *arXiv.2001.09382* **2020**, DOI: 10.48550/arXiv.2001.09382.