JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim Article

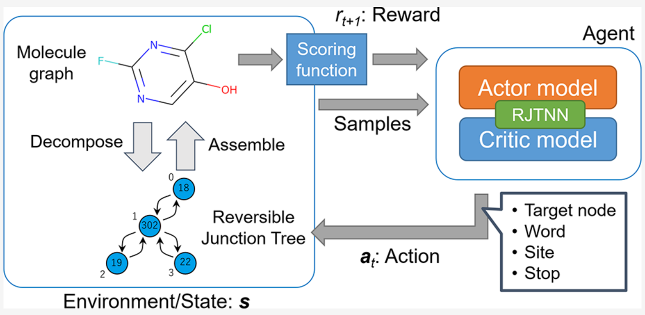
Molecular Design Method Using a Reversible Tree Representation of Chemical Compounds and Deep Reinforcement Learning

Ryuichiro Ishitani,* Toshiki Kataoka, and Kentaro Rikimaru

Cite This: J. Chem. Inf. Model. 2022, 62, 4032–4048 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Automatic design of molecules with specific chemical and biochemical properties is an important process in material informatics and computational drug discovery. In this study, we designed a novel coarse-grained tree representation of molecules (Reversible Junction Tree; "RJT") for the aforementioned purposes, which is reversely convertible to the original molecule without external information. By leveraging this representation, we further formulated the molecular design and optimization problem as a tree-structure construction using deep reinforcement learning ("RJT-RL"). In this method, all of the intermediate and final states of reinforcement learning are convertible to valid molecules, which could efficiently guide the optimization process in simple benchmark tasks. We further examined the multiobjective optimization and fine-tuning of the reinforcement learning models using RJT-RL, demonstrating the applicability of our method to more realistic tasks in drug discovery.

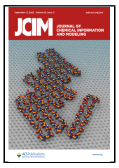


■ INTRODUCTION

Exploration of novel chemical compounds is an important process in drug and material discovery. The process entails iterative trials, including the proposal of candidate compounds and their experimental evaluation, which are usually expensive and time-consuming. Furthermore, while the chemical space of feasible and chemically synthesizable compounds is astronomically large (estimated to be on the order of 10^{60}), only a small portion of this space has been explored for drugs or functional materials. As it is impossible to explore compounds having the desired properties from this vast chemical space using *ad-hoc* experimental trials, computational exploration of chemical space has been proposed and is now gaining considerable attention owing to recent advancements in computational power. In computational exploration, all of the steps in the iterative trials, including the proposal of novel chemical compounds and their property evaluation, were performed *in silico*. Using extensive computational resources, this method extends the exploration range within the vast chemical space remarkably. However, owing to the discrete nature of chemical structure representation (*i.e.*, graphs in discrete mathematics), the difficulty to efficiently explore the chemical space still persists.

To address this problem, new exploration methods using deep neural networks[1] (DNNs; for comprehensive reviews see refs 2, 3) have been proposed. One example is the latent-space-based method, which involves a variational autoencoder (VAE[4]). In this method, a DNN, known as a generative model, learns the mappings between the discrete space of the chemical compounds and the continuous latent manifold. After learning the generative model, exploration is performed over the continuous latent manifold.[5] Bayesian optimization or metaheuristics-based methods were used for the optimization process because the molecular properties used as scoring functions are usually not differentiable with respect to the latent manifold, even though the latent space itself is continuous. However, this method has several limitations. The first concerns the dimensions of the latent manifold. For example, on using the practical size of the training dataset (10^5–10^6 compounds), learned latent manifolds with a reasonable reconstitution rate usually result in dimensions greater than 50, which is not easy for the available optimization algorithms. Second, no reliable indicator exists for assessing the rationality of learned mapping. Although we can evaluate the model using the reconstruction error against the training dataset, it is unclear whether a low reconstruction error is sufficient to guide the chemical structure optimization with the given scoring functions. The third problem is that the learning process of the generative model is detached from the optimization process of the chemical compounds with regard

Received: March 29, 2022
Published: August 12, 2022



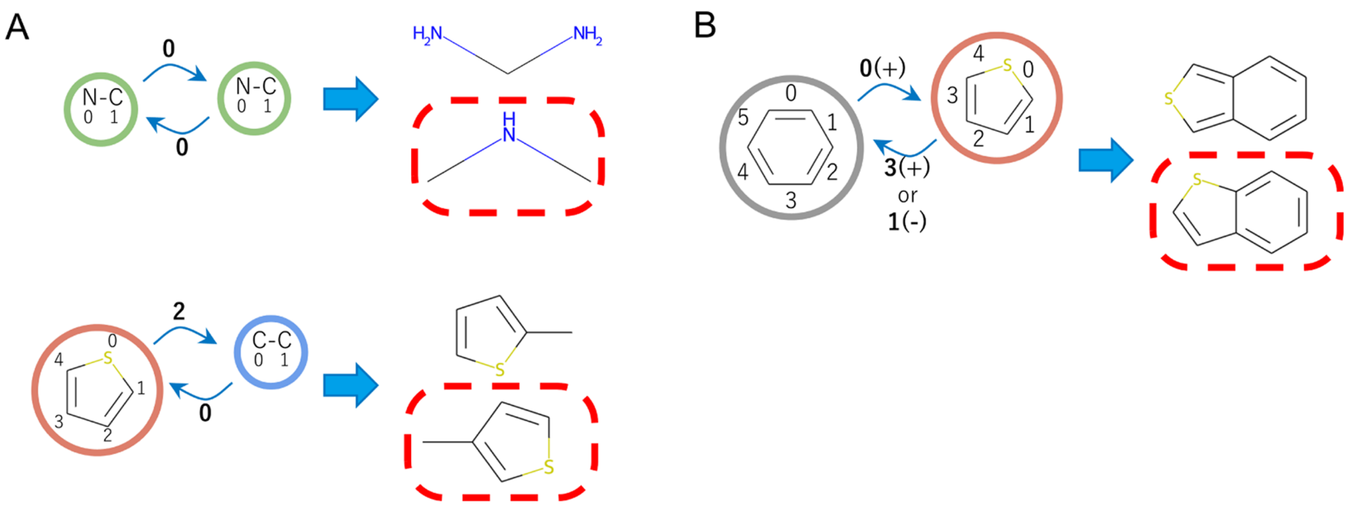


Figure 1. Definition of the site information that enables the reversible conversion to the original molecular structure. (A) Site information for type-1 edges. In the left panels, circles and arrows indicate the nodes and edges of the tree representation, respectively. Numbers near the arrows indicate site information. The left panels show the possible molecular structures assembled from the tree representation. Using site information, the original structure indicated by the dashed line can be selected without ambiguity. (B) Site information for type-2 edges. The numbers and (+)/(−) signs near the arrows indicate site information.

to the scoring function. The pretrained generative model remained unchanged during optimization and did not learn the scoring function. Recently, to address the third problem, the retraining of the generative model based on optimization results has been proposed.[6] This method requires an iterative process of optimization and training, which is computationally expensive and time consuming.

Another example is a reinforcement learning (RL)-based method. In this method, the molecular design problem is formulated as a Markov decision process wherein an agent learns the optimal policy based on the rewards offered by its surrounding environment. The RL-based method can partly alleviate the problem of latent-space-based methods because the policy approximated by the DNN model learns the scoring function during optimization. RL-based methods can be classified into two types based on the representation of chemical compounds. The first type uses text representation (*e.g.*, SMILES), which allows us to leverage techniques used in natural language processing. Although this representation is widely used (including in latent-space-based methods[5]), the generated text may contain grammatically invalid results that can affect optimization performance. This type of method includes the REINVENT.[7] The second type uses a graph representation of the chemical structure. Examples of this type include GCPN[8] and MolDQN.[9] In this representation, the generated graphs are theoretically decodable to valid molecules, thereby avoiding the grammatically invalid results in the text representation. However, the molecular properties can change drastically even with a single action of RL, such as in the last step of closing a conjugated aromatic ring. This makes it difficult to guide the agent directly to the optimal action at each RL step. Thus, the molecular representation using chemical groups (*e.g.*, phenyl groups) as building blocks seem effective for RL-based molecular design. In contrast, molecular structure generation algorithms using predefined fragments as building blocks have been continuously studied over the last century, and many approaches have been proposed.[10−15] However, it is difficult to integrate such methods with learning-based optimization algorithms, including RL.

Recently, Jin et al.[16] proposed a novel coarse-grained molecule representation by leveraging junction tree (JT) decomposition to a molecule graph. In this method, a molecule is represented as a tree (*i.e.*, a graph without circular or closed paths) with nodes corresponding to rings or bonds. Similar to the all-atom graph representation, this JT representation always corresponds to a valid molecule by composing valid chemical fragments as building blocks. This method has been successfully applied to several molecular-generative models.[16,17] However, the JT representation cannot be reversibly converted to the original molecule without auxiliary information. Therefore, additional neural networks (such as atom-based graph convolution networks) are required to supplement this information for decoding molecules from the generated JT representation, making this technique complicated and computationally intensive.

In this study, we propose a novel RL-based molecular generation and optimization framework that leverages JT-based representation. For this purpose, we tailored the JT representation to be reverse-convertible to the original molecule without auxiliary neural networks (reversible JT; RJT). This reversible molecular representation enabled us to formulate the molecular generation and optimization problem as tree-structure construction using RL (RJT-RL). The reversible nature of the RJT enables the evaluation of molecules decoded from the intermediate states in the RL episodes. We tested the method using simple molecular design tasks as benchmarks and subsequently applied it to more realistic tasks in drug discovery involving multiobjective scoring functions.

■ METHODS

Reversible Tree Representation of Molecules. In the original JT representation, the molecular graph $\mathcal{G}$ is converted into a tree representation $\mathcal{T}$ using the JT algorithm.[16] Briefly, the molecular graph $\mathcal{G}$ is fragmented into bonds and rings using the smallest set of smallest rings (SSSR) algorithm implemented in RDKit.[18] Then, a node is assigned to each bond or ring, and the nodes are connected by an edge when they share atoms in $\mathcal{G}$ to construct the tree representation $\mathcal{T}$ (Figure S1). If the bond nodes intersect at more than one node, the resulting $\mathcal{T}$ creates a cyclic structure. This situation is avoided by inserting a single atom shared between these nodes as a "singleton node" (see the original study[16] for a detailed description of the algorithm). Hereafter, we term the nodes representing the bond and ring as "bond nodes" and "ring nodes," respectively. A possible combination of unique atom clusters in the dataset forms a vocabulary $\mathbb{X}$. Each node N_i is assigned a word ID $w_i \in \mathbb{X}$ based on the atom cluster in the node. This coarse-grained representation $\mathcal{T}$ allows us to build a molecule from chemically valid fragments, thereby avoiding atom-by-atom molecular creation through chemically invalid intermediates. However, the JT representation $\mathcal{T}$ cannot be reversibly converted into the original molecule because the connection information between the nodes is lost in this coarse-grained representation. In previous studies,[16,17] this problem was avoided by introducing supplementary neural networks, which complicated the JT representation framework.

In this study, we introduce the reversible JT (RJT) representation ($\tilde{\mathcal{T}}$) of molecules by extending the JT decomposition of $\mathcal{G}$. To eliminate arbitrariness in the node connection (*i.e.*, to determine atoms shared between two nodes connected by an edge), we record the ID of the atoms shared between two adjacent nodes (N_i and N_j) as the "site information" $\sigma_{i,j}$. The site information is defined as follows (Figure 1). The original JT implementation involves sharing of one or two atoms between adjacent nodes connected by the JT edge. Hereafter, the JT edge sharing one atom is denoted as a "type-1 edge" (Figure 1A), while that sharing two atoms as a "type-2 edge" (Figure 1B). Type-1 edges can be further classified into three cases: edges connecting (i) bonds and singleton nodes, (ii) two bond nodes, and (iii) bond and ring nodes. In these three cases, the indices of the shared atom in both nodes are recorded as the site information $\sigma_{i,j}$. In contrast, type-2 edges always connect two ring nodes, and the indices of the two atoms in both the ring nodes are recorded. Because the atoms shared between the nodes are always adjacently located in the ring, the indices of the first atom and the direction ID (+1 or −1) of the second atom are recorded as site information $\sigma_{i,j}$. A spiro connection between two rings is treated as a special case of the type-2 edge by assigning a direction ID of 0, although only one atom is shared between the nodes. The most appropriate location for storing this site information is edge $E_{i,j}$, which represents the existence of shared atoms in N_i and N_j in $\mathcal{T}$. Thus, in this study, the site information is encoded as edge features.

The algorithm for converting the RJT representation $\tilde{\mathcal{T}}$ to molecular graph $\mathcal{G}$ (Algorithm 1) can be described as follows: this method is similar to the assembly algorithm of Jin et al.,[16] except that enumerating all combinations of node-to-node attachments in $\mathcal{G}$ is not necessary. In our method, the predicted $\tilde{\mathcal{T}}$ is traversed in the depth-first order by attaching a subgraph, corresponding to the child node, to the constructed graph. Using the site information, the atom(s) shared between two nodes can be uniquely determined, and the nodes could be deterministically assembled to convert $\tilde{\mathcal{T}}$ to $\mathcal{G}$. However, it is possible that the predicted site information is incompatible with the node types (*e.g.*, more than four atoms are connected to one carbon atom). In such cases, the second (or third) probable site information can be utilized based on the output of the softmax logits of the neural network (described later). Alternatively, an exception could be raised to indicate that a particular RJT contains invalid information. The latter implementation was used to simplify the code.

Algorithm 1 Depth-first assembly of $\tilde{\mathcal{T}}$

```
Require: Reversible tree representation of mol $\tilde{\mathcal{T}}$
 1: Initialize: $i \leftarrow$ One node selected from $\tilde{\mathcal{T}}$
 2: Initialize: $j \leftarrow \emptyset$
 3: Initialize: $\mathcal{G} \leftarrow$ mol graph of node $i$
 4: function DepthFirstAssemble($i, j, \mathcal{G}$)
 5:     for $k \in N(i)\backslash j$ do
 6:         Set $\sigma_{i,k} \leftarrow$ site information of $i$ against $k$
 7:         Set $\sigma_{k,i} \leftarrow$ site information of $k$ against $i$
 8:         Set $\mathcal{G}' \leftarrow$ mol graph of node $k$
 9:         Attach mol graph $\mathcal{G}'$ to $\mathcal{G}$ using $\sigma_{i,k}$ and $\sigma_{k,i}$
10:         DepthFirstAssemble($k, i, \mathcal{G}$)
11:     end for
12: end function
```

RJT-Based Neural Network. The neural network architecture that encodes the RJT representation into hidden vectors with fixed sizes ($\{\mathbf{h}_i\}$) can be described as follows

$$\{\mathbf{h}_i\} = \mathrm{RJTNN}(\tilde{\mathcal{T}})$$

Each node N_i is represented by a one-hot vector of word ID w_i, while edge $E_{i,j}$ is represented by a one-hot vector of site information $\sigma_{i,j}$. These one-hot representations of w_i and $\sigma_{i,j}$ are then converted to node and edge features $\mathbf{x}_i$ and $\mathbf{y}_{i,j}$ using the learnable embedding matrices $\mathbf{E}^{\mathrm{node}}$ and $\mathbf{E}^{\mathrm{edge}}$, respectively, as follows

$$\mathbf{x}_i = \mathbf{E}^{\mathrm{node}} \cdot w_i$$

$$\mathbf{y}_{i,j} = \mathbf{E}^{\mathrm{edge}} \cdot \sigma_{i,j} \;\oplus\; \mathbf{E}^{\mathrm{edge}} \cdot \sigma_{j,i}$$

To encode the RJT, including both node and edge features, we extended the tree-based gated recurrent unit (GRU)[16] as follows

$$\mathbf{s}_{i,j} = \sum_{k \in N(i)\backslash j} (\mathbf{m}_{k,i} \oplus \mathbf{y}_{k,i})$$

$$\mathbf{z}_{i,j} = \mathrm{sigmoid}(\mathbf{W}^{\mathrm{z}}\mathbf{x}_i + \mathbf{U}^{\mathrm{z}}\mathbf{s}_{i,j} + \mathbf{b}^{\mathrm{z}})$$

$$\mathbf{r}_{k,i} = \mathrm{sigmoid}(\mathbf{W}^{\mathrm{r}}\mathbf{x}_i + \mathbf{U}^{\mathrm{r}}(\mathbf{m}_{k,i} \oplus \mathbf{y}_{k,i}) + \mathbf{b}^{\mathrm{r}})$$

$$\tilde{\mathbf{m}}_{i,j} = \tanh\left(\mathbf{W}^{\mathrm{m}}\mathbf{x}_i + \mathbf{U}^{\mathrm{m}} \sum_{k \in N(i)\backslash j} \mathbf{r}_{k,i} \odot (\mathbf{m}_{k,j} \oplus \mathbf{y}_{k,j})\right)$$

$$\mathbf{m}_{i,j} = (1 - \mathbf{z}_{i,j}) \odot \mathbf{s}_{i,j} + \mathbf{z}_{i,j} \odot \tilde{\mathbf{m}}_{i,j}$$

In the above formulas, we extended the tree-based GRU; however, the tree-based long short-term memory (LSTM)[17] could also be extended to accommodate the edge features in a similar manner. The message vectors $\mathbf{m}_{i,j}$ are updated in two phases (bottom-up and top-down), as in the original tree-based GRU. Finally, the hidden vector $\mathbf{h}_i$ of node N_i is computed by aggregating the message vectors $\mathbf{m}_{i,j}$ as follows

$$\mathbf{h}_i = \mathrm{ReLU}\left(\mathbf{W}^{\mathrm{o}}\mathbf{x}_i + \sum_{k \in N(i)} \mathbf{U}^{\mathrm{o}}\mathbf{m}_{k,i}\right)$$

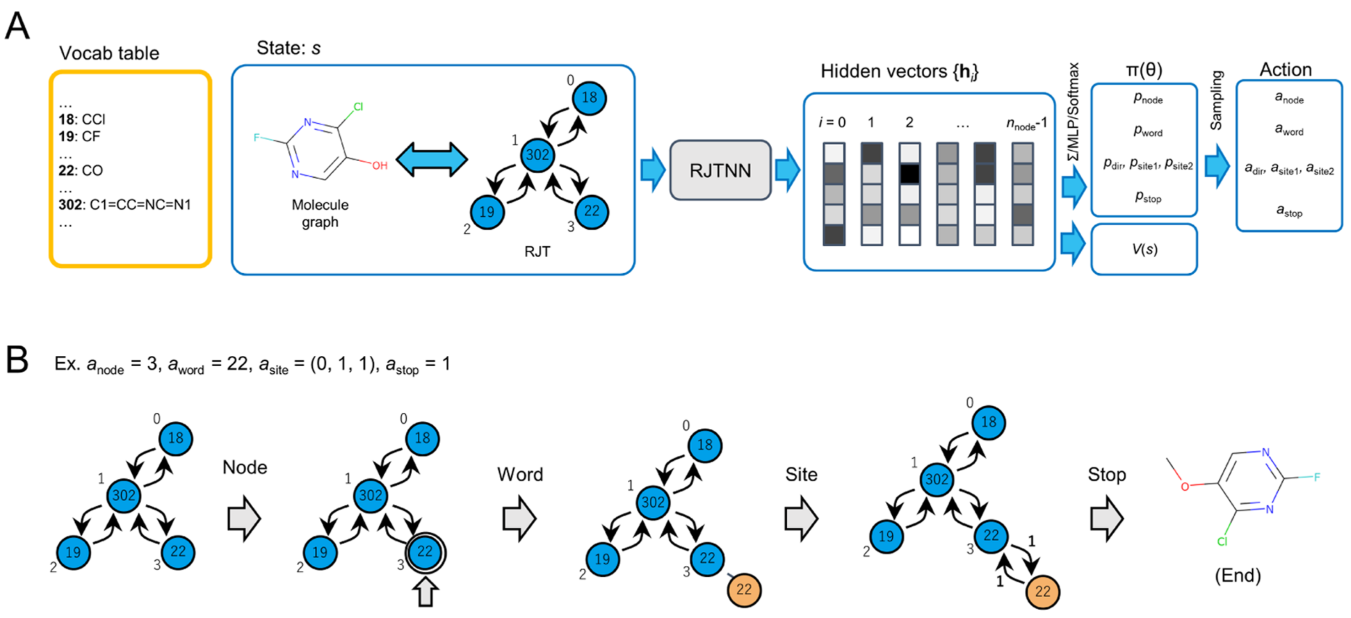


Figure 2. Network architecture of RJT-RL and the actions taken by the agent. (A) Calculation flow of the action and value function estimates. The RJT representing the state is converted to hidden vectors $\{\mathbf{h}_i\}$ by the RJTNN network defined in the main text, using which the policy distribution π_θ and value function estimate V_θ are calculated using subsequent networks. The action taken by the agent is determined by sampling from this policy distribution. The numbers in the tree nodes indicate the word ID in the vocabulary table. (B) Example of an action sampled from the policy distribution and modification to an RJT representing the state. The state before the application of the action represents the molecule shown in panel (A). After applying the action, the state (*i.e.*, RJT) is modified to represent the molecule shown in the leftmost panel.

where ReLU stands for the rectified linear unit, defined as $\mathrm{ReLU}(x) = \max(0, x)$.

RL Using the RJT Representation. In this study, we formulated a molecular design task for tree generation using RL.[19] In general RL settings, we consider that an agent receives state $s_t \in \mathbb{S}$ from the environment and selects an action $a_t \in \mathbb{A}(s_t)$ according to its policy $\pi = \pi(a_t|s_t)$ at each time step t, where $\mathbb{S}$ is a set of possible states and $\mathbb{A}(s_t)$ is a set of possible actions at state s_t.

After taking an action, the agent receives the next state s_{t+1} and a scalar reward r_t, and proceeds to the next step $t + 1$. The episode ends when the agent receives a terminal state at time step T and then proceeds to the next episode with an initial time step $t = 0$. The return R_t is the total accumulated reward from time step t to T with discount rate γ, as follows

$$R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$$

Action value $Q^\pi(s, a)$ is defined as the expected return on selecting action a in state s after following policy π, whereas the value of state $V^\pi(s)$ is defined as the expected return from state s after following policy π. The goal of the agent is to maximize the value of state s_t, which is the expected return R_t from each state s_t.

Policy-based RL methods directly parameterize the policy π_θ and optimize its parameter θ using the gradient estimator of $\mathbb{E}[R_t]$. The policy gradient theorem states that the unbiased estimate of $\nabla_\theta \mathbb{E}[R_t]$ can be estimated as[20]

$$\nabla_\theta \mathbb{E}[R_t] \propto \mathbb{E}[\nabla_\theta \log \pi_\theta(a_t|s_t) R_t] = L_t^{\mathrm{PG}}$$

To reduce the variance of this gradient estimate, the estimate of advantage ($\hat{A}_t$; $A(s, a) = Q(s, a) - V(s)$) was used instead of R_t.[21,22] Although the use of this advantage increases stability, it often leads to destructively large policy updates.

In the proximal policy optimization (PPO) algorithm,[23] the clipped surrogate objective L_t^{CLIP} is optimized instead of L_t^{PG} to alleviate this problem as follows

$$L_t^{\mathrm{CLIP}}(\theta) = \mathbb{E}[\min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$$

where $r_t(\theta) = \pi_\theta(a_t|s_t)/(\pi_{\theta_{\mathrm{old}}}(a_t|s_t))$, and $\mathrm{clip}(\bullet)$ clips $r_t(\theta)$ outside the interval between $[1 - \epsilon, 1 + \epsilon]$. Then, the total loss function L_t^{PPO}, given as

$$L_t^{\mathrm{PPO}}(\theta) = \mathbb{E}[L_t^{\mathrm{CLIP}}(\theta) - c_{\mathrm{VF}} L_t^{\mathrm{VF}}(\theta) + c_{\mathrm{ent}} S[\pi_\theta](s_t)]$$

is optimized, where $L_t^{\mathrm{VF}}(\theta) = (V_\theta(s_t) - V_t^{\mathrm{targ}})^2$ is the squared error loss of the value function and $S[\pi_\theta]$ (s_t) is an entropy bonus term to ensure efficient exploration. For $\hat{A}_t$, the truncated version of the generalized advantage estimation is used, as in the original PPO implementation, as follows

$$\hat{A}_t = \sum_{T-1}^{i=t} (\gamma\lambda)^{i-t} \delta_i$$

where $\delta_i = r_t + \gamma V(s_{t+1}) - V(s_t)$, and λ is the generalized advantage estimator parameter.[22]

Definition of Policy and Value Function Networks. The RJT representation of the molecule was used as the state $s_t \in \mathbb{S}$ in RL, where $\mathbb{S}$ is a possible set of molecules that can be converted to RJT. The agent takes the action $a_t \in \mathbb{A}(s_t)$ to modify the RJT of s_t and constructs the RJT in the next step (Figure 2A). Here, the action a_t is defined using the following four components: (i) selection of the word ID w_i of the new node N_i, (ii) selection of the existing node N_j for attaching to the new node, (iii) prediction of the site information $\sigma_{i,j}$ (for connecting the two subgraphs represented by N_i and N_j), and (iv) determining whether the episode ends (Figure 2B). The policy can then be expressed as a probability distribution function for each component of the action defined here. In this study, these probability distribution functions were approximated by RJTNN, using the RJT of s_t ($\tilde{\mathcal{T}}$) as an input (Figure 2A).

First, we calculated the hidden vectors $\{\mathbf{h}_i\}$ for each node of the RJT using RJTNN, as follows

$$\{\mathbf{h}_i\} = \text{RJTNN}(\tilde{\mathcal{T}})$$

Then, each component of the policy distribution was calculated as follows

$$p_{\text{stop}}(\tilde{\mathcal{T}}) = \text{sigmoid}\left(\text{MLP}\left(\sum_i \mathbf{W}^{\text{a}}\mathbf{h}_i\right)\right)$$

$$p_{\text{node}}(\tilde{\mathcal{T}}) = \text{softmax}_{n_{\text{node}}}(\text{MLP}(\mathbf{h}_i))$$

$$p_{\text{word}}(\tilde{\mathcal{T}}) = \text{softmax}_{n_{\text{voc}}}(\text{MLP}(\mathbf{h}_{a_{\text{node}}}))$$

$$p_{\text{dir}}(\mathcal{T}) = \text{softmax}_3(\text{MLP}(\mathbf{h}_{a_{\text{node}}}))$$

$$p_{\text{site1}}(\tilde{\mathcal{T}}) = \text{softmax}_{n_{\text{site}}}(\text{MLP}(\mathbf{h}_{a_{\text{node}}}))$$

$$p_{\text{site2}}(\tilde{\mathcal{T}}) = \text{softmax}_{n_{\text{site}}}(\text{MLP}(\mathbf{h}_{a_{\text{node}}}))$$

where n_{node} is the number of nodes in $\tilde{\mathcal{T}}$, n_{voc} is the size of vocabulary $\mathbb{X}$, and n_{site} is the maximum number of possible combinations of site information. The MLP function stands for a multilayer perceptron with two layers, including the ReLU activation function. The action (a_{node}, a_{word}, a_{site}, a_{stop}) taken by the agent is determined by sampling from this policy distribution as follows

$$a_{\text{node}} \in \{0, \cdots, n_{\text{node}} - 1\} \sim p_{\text{node}}(\tilde{\mathcal{T}})$$

$$a_{\text{word}} \in \{0, \cdots, n_{\text{voc}} - 1\} \sim p_{\text{word}}(\tilde{\mathcal{T}})$$

$$a_{\text{site}} = \{a_{\text{dir}}, a_{\text{site1}}, a_{\text{site2}}\}$$

$$a_{\text{dir}} \in \{-1, 0, 1\} \sim p_{\text{dir}}(\tilde{\mathcal{T}})$$

$$a_{\text{site1}} \in \{0, \cdots, n_{\text{site}} - 1\} \sim p_{\text{site1}}(\tilde{\mathcal{T}})$$

$$a_{\text{site2}} \in \{0, \cdots, n_{\text{site}} - 1\} \sim p_{\text{site2}}(\tilde{\mathcal{T}})$$

$$a_{\text{stop}} \in \{0, 1\} \sim p_{\text{stop}}(\tilde{\mathcal{T}})$$

The value function is also approximated using the shared RJTNN with the policy network as follows

$$V(s_t) = V(\tilde{\mathcal{T}}) = \text{MLP}\left(\sum_i \mathbf{W}^{\text{a}}\mathbf{h}_i\right)$$

Using these neural network functions, the PPO target loss function ($L_t^{\text{PPO}}(\theta)$) was minimized using the Adam optimizer during RL training. PyTorch[24] and PFRL[25] were used for the deep learning and RL frameworks, respectively, for the experiments.

Expert Learning. To guide the exploration space of RL for drug-like molecules, the policy network was pretrained using the given dataset ("expert dataset") similar to previous studies.[7,8] For pretraining, the dataset molecules were converted to RJTs, and the root node was randomly selected from the sampled $\tilde{\mathcal{T}}$ from this dataset. The path for traversing $\tilde{\mathcal{T}}$ from the root node was generated in the breadth (or depth)-first order, from which one edge (connecting N_i and N_j) was randomly selected. This edge represents the process of creating a new tree from a state comprising N_i and its ancestral nodes. Thus, the calculated state s_t and action a_t from the edge are used to minimize the negative log likelihood of the policy function, *i.e.*,

$$L_t^{\text{exp}}(\theta) = -\log \pi_\theta(a_t|s_t)$$

In the training phase of RL, the learned policy function is not expected to deviate significantly from the pretrained policy and generate non-drug-like molecules. The use of augmented likelihood in the REINVENT algorithm[7] prevents the learned policy from deviating significantly from the pretrained model. In this study, we jointly trained the policy using an expert dataset with the target function of the PPO by minimizing the following target function

$$L_t^{\text{All}}(\theta) = L_t^{\text{PPO}}(\theta) + c_{\text{exp}}L_t^{\text{exp}}(\theta)$$

where coefficient c_{exp} is a hyperparameter that controls for the effect of the expert dataset.

Experiment Settings. In all experiments, the valences of the atoms in the assembly process (conversion of RJT to a molecule) were checked. If excess valence was detected (*e.g.*, a carbon atom with more than five bonds), a small negative score (c_{invalid}) was given as the reward for step t to discourage the actions that produced invalid molecules. Next, in RJT-RL, any molecular property could be used as the reward function r_t, which is calculated at every RL step because RJTs corresponding to intermediate states are convertible to valid molecules. Thus, we examined the effectiveness of this "step reward" and compared it with the "final reward" case.

$$r_t = \begin{cases} c_{\text{invalid}} & \text{if } s_t \text{ is invalid} \\ R_{\text{step}}(t) - R_{\text{step}}(t-1) & t < t_{\max} \\ R_{\text{step}}(t) - R_{\text{step}}(t-1) + R_{\text{fin}} & t = t_{\max} \end{cases}$$

Here, R_{step} and R_{fin} are the step and final reward functions, respectively, defined according to the specific tasks, and $t_{\max}$ is the last time step in this episode. We also defined $R_{\text{step}}(-1) \equiv 0$. Finally, the effect of the "duplication count penalty" on reward function was examined. One problem of RL is the balance between exploration and exploitation; poor exploration that maximizes short-term rewards results in trapping in the local minima. To avoid this problem, the entropy bonus term $S[\pi_\theta](s_t)$ was introduced into the loss function of the PPO.[23] To further facilitate the exploration of the chemical space, we introduced a duplication count penalty function, which modifies the final reward R_{fin} calculated at the end of the episode as follows

$$r = \begin{cases} s & n_{\text{dup}} \le N_{\text{lim}} \\ -\max(0, s) & n_{\text{dup}} > N_{\text{lim}} \end{cases}$$

where n_{dup} is the duplication count of the generated molecule in this episode and s is the score calculated for the generated molecule. This function is intended to reset R (the total return of the episode) to zero if the duplication count exceeds a specified threshold. A threshold value of $N_{\text{lim}} = 2$ was utilized for all experiments in this study. This penalty term is inspired

Table 1. Summary of the Penalized Log *P* Experiments[a]

	method	reward	duplication penalty	Novel	Valid	Uniq	Frag	Scaf	IntDiv	Filt
P1	RJT-RL	step	off	1	1	0.991	0.078	0.016	0.593	0.602
P2	RJT-RL	final	on	1	1	1	0.132	0.024	0.521	0.807
P3	RJT-RL	step	on	1	1	0.996	0.077	0.021	0.593	0.553
P4	REINVENT			1	0.952	0.944	0.153	0.002	0.603	0.824

[a]The MOSES metrics[29] were calculated for the molecules generated in the last 1000 episodes (Novel: novelty, Valid: validity, Uniq: uniqueness, Frag: fragment similarity, Scaf: scaffold similarity, IntDiv: internal diversity (p = 1), and Filt: fraction of molecules passing the unwanted structure filter).

Table 2. Summary of the Similarity Experiments[a]

	target	method	reward	duplication penalty	Novel	Valid	Uniq	Frag	Scaf	IntDiv	Filt
S1	vortioxetine	RJT-RL	step	off	1	1	0.069	0.169	0.027	0.348	0.543
S2	vortioxetine	RJT-RL	final	on	1	1	0.917	0.514	0.077	0.715	0.922
S3	vortioxetine	RJT-RL	step	on	1	1	0.849	0.245	0.002	0.641	0.829
S4	vortioxetine	REINVENT			1	0.970	0.970	0.970	0.689	0.837	0.693
S5	vortioxetine	CReM			1	1	1	0.608	0.034	0.724	0.488
C1	celecoxib	RJT-RL	step	off	1	1	0.100	0.119	0.098	0.387	0.982
C2	celecoxib	RJT-RL	final	on	1	1	0.951	0.183	0.011	0.776	0.809
C3	celecoxib	RJT-RL	step	on	1	1	0.926	0.357	0.004	0.745	0.777
C4	celecoxib	REINVENT			1	0.942	0.942	0.975	0.682	0.843	0.712
C5	celecoxib	CReM			1	1	1	0.608	0.034	0.724	0.488

[a]The metrics of the molecules generated during the last 4000 episodes were calculated, as shown in Table 1.

Table 3. Summary of the Docking Experiments[a]

	target	method	reward	3D conf	Novel	Valid	Uniq	Frag	Scaf	IntDiv	Filt
D1	interaction	RJT-RL	final	random	1	1	0.934	0.062	0	0.643	0.135
D2	interaction	RJT-RL	step	random	1	1	0.875	0.374	0	0.682	0.096
D3	interaction	RJT-RL	final	Enum	1	1	0.845	0.627	0	0.731	0.333
D4	interaction	CReM		random	1	1	0.999	0.580	0	0.562	0.378
M1	multiobjective	RJT-RL	final/step	random	1	1	0.933	0.176	0.002	0.721	0.950
M2	multiobjective	CReM		random	1	1	1	0.803	0	0.640	0.586
F1	fine-tuning	RJT-RL	final/step	random	1	1	0.876	0.147	0	0.685	0.458

[a]The metrics of the molecules generated by the last 10,000, 8000, and 2000 episodes for D1−D4, M1−M2, and F1, respectively, were calculated, as shown in Table 1.

by previous studies,[26−28] wherein the count-base bonus term was shown to facilitate exploration in various RL tasks.

All of the experiments described in the Results and Discussion section utilized the ethane molecule ("CC" in SMILES representation), which is the simplest atom cluster in vocabulary $\mathbb{X}$, unless otherwise stated. The policy network with a hidden vector size of 128 dimensions was pretrained using a dataset derived from the ZINC250k dataset.[5] In the derived dataset, macrocyclic compounds or rings containing more than eight atoms were removed. Additionally, bridge compounds containing more than eight substitution sites were removed. The resulting dataset contains 246,416 molecules. We tested whether all RJT representations of the molecules in this dataset could be reversibly converted to the original molecules. For comparison, REINVENT,[7] which is a SMILES-based molecular generator that uses RL, was used. The policy network was pretrained using the same dataset, and RL training was performed using the same scoring function and hyperparameters of k = 1 and σ = 20. In addition, CReM,[10] a rule-based structure generation method using fragments as building blocks, was employed. The fragment library for CReM was generated using the ZINC250k dataset, and the optimization method described in this study was used for structure generation. Similar computational resources were used (CPU: one core of Intel Xeon Gold 6254 CPU@ 3.10GHz and GPU: NVIDIA Tesla V100) for all experiments.

RESULTS AND DISCUSSION

To examine the effectiveness of RJT-RL, we performed several experiments using molecular design as a benchmark. All of the experiments and their abbreviations are summarized in Tables 1, 2, and 3. We calculated several measures defined in the MOSES benchmark suite[29] for molecules generated in the last 1/5 episodes.

Penalized Log *P* Optimization. First, we verified the effectiveness of our method using a single chemical property that is easily calculated using only chemical structure. In the first experimental setup, we evaluated the penalized Log *P* score,[30] which is the water−octanol partition coefficient (Log *P*) that also accounts for the ring size and synthetic accessibility (SA) score.[31] Although the maximization of the penalized Log *P* score itself has no actual application in drug discovery, we first attempted this task to assess whether our method can optimize the computationally easy problem beyond the example molecules in the training dataset. Furthermore, many previous studies that performed optimization in the chemical space have employed this target score, thereby allowing the comparison of our method with these

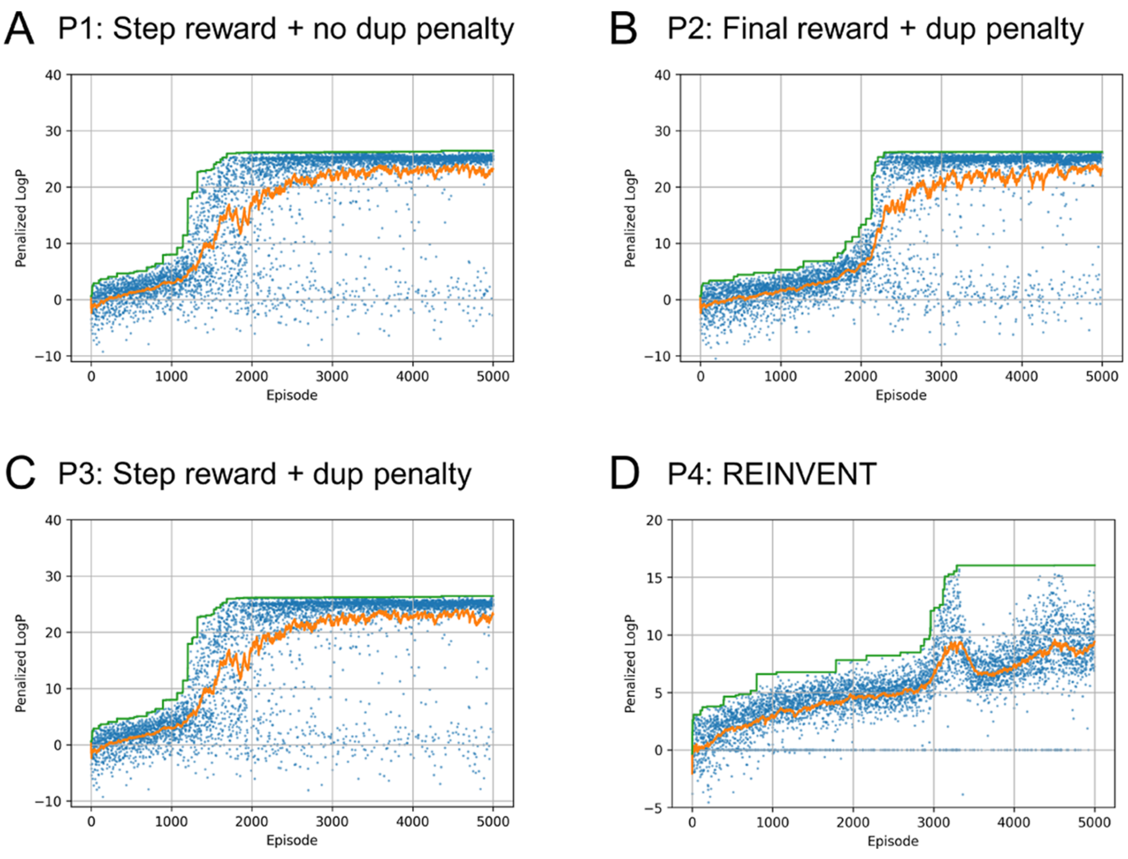


Figure 3. Results of penalized Log *P* optimization by (A−C) RJT-RL (P1−P3; see Table 1) and (D) REINVENT. The penalized Log *P* values for each episode are plotted in blue, whereas their moving average and maximum values are plotted in orange and green, respectively.

studies. For RJT-RL, we examined three cases (P1, P2, and P3 in Table 1) that differed in the settings of the reward evaluation (step or final) and the duplication penalty. In P1 and P3, the penalized Log *P* score was used as the step reward function, R_{step}, but the duplication penalty was turned off in P1. In P2, the penalized Log *P* score was used as R_{fin}, and no reward was given for the intermediate steps.

The results are shown in Figures 3, and S2 and Table 1. In all cases (P1, P2, and P3), molecules with high rewards were continuously generated, even in the absence of the duplication penalty (Figure 3A−C). Diverse molecules were generated without any penalty (uniqueness in Table 1); thus, the duplication penalty effect may be limited in this experimental setup. The comparison of the final and step reward cases (P2 *vs* P1/P3) showed that the reward functions tended to rise faster in P1/P3 than in P2 (Figure 3A−C). A similar tendency was observed in all other experiments, starting from different random states (Figure S3). These results suggest that the rewards of the RL intermediate states facilitated the optimization process. The REINVENT model (P4) also continuously generated high-scoring molecules, but the scores did not exceed those generated by our method (Figure 3D). The top three penalized Log *P* scores obtained in the experiments are summarized in Table 4, along with the results from previous studies. The comparison shows that our method performed comparably to the weighted retraining method,[6] which also updates the generative model during the optimization process according to the target function.

Similarity-Guided Molecule Generation. Next, we verified the effectiveness of our method using a simple task

Table 4. Summary of the Results of the Penalized Log *P* Optimization Experiment Performed in this and Previous Studies[a]

			best scores		
method	reward type	duplication penalty	1	2	3
RJT-RL (P1)	step	off	26.45	26.39	26.34
RJT-RL (P2)	final	on	26.23	26.22	26.21
RJT-RL (P3)	step	on	26.45	26.39	26.34
REINVENT (P4)			16.04	16.03	15.71
CharVAE			1.98	1.42	1.19
JT-VAE			5.3	4.93	4.49
GCPN			7.98	7.85	7.80
MolDQN			11.71	11.63	11.63
Weighted retraining			27.84	27.59	27.21

[a]Previous studies include the Following Methods: CharVAE,[5] JT-VAE,[16] GCPN,[8] MolDQN,[9] and Weighted Retraining[6]

to generate molecules similar to a given query structure. To measure the similarity between molecules *i* and *j*, we used the Jaccard index[32] $J_{i,j}$ of the RDKit implementation of the FCFP4 fingerprints.[33] Although the similarity-guided optimization problem could seem trivial to human intuition, this task is significant for computational optimization using fingerprints as the similarity measure. For example, when attempting to generate an ether group, the alcohol moiety must be generated before reaching the final state. The fingerprint bits of alcohol and ether oxygens are different in the definition of FCFP4;[33]

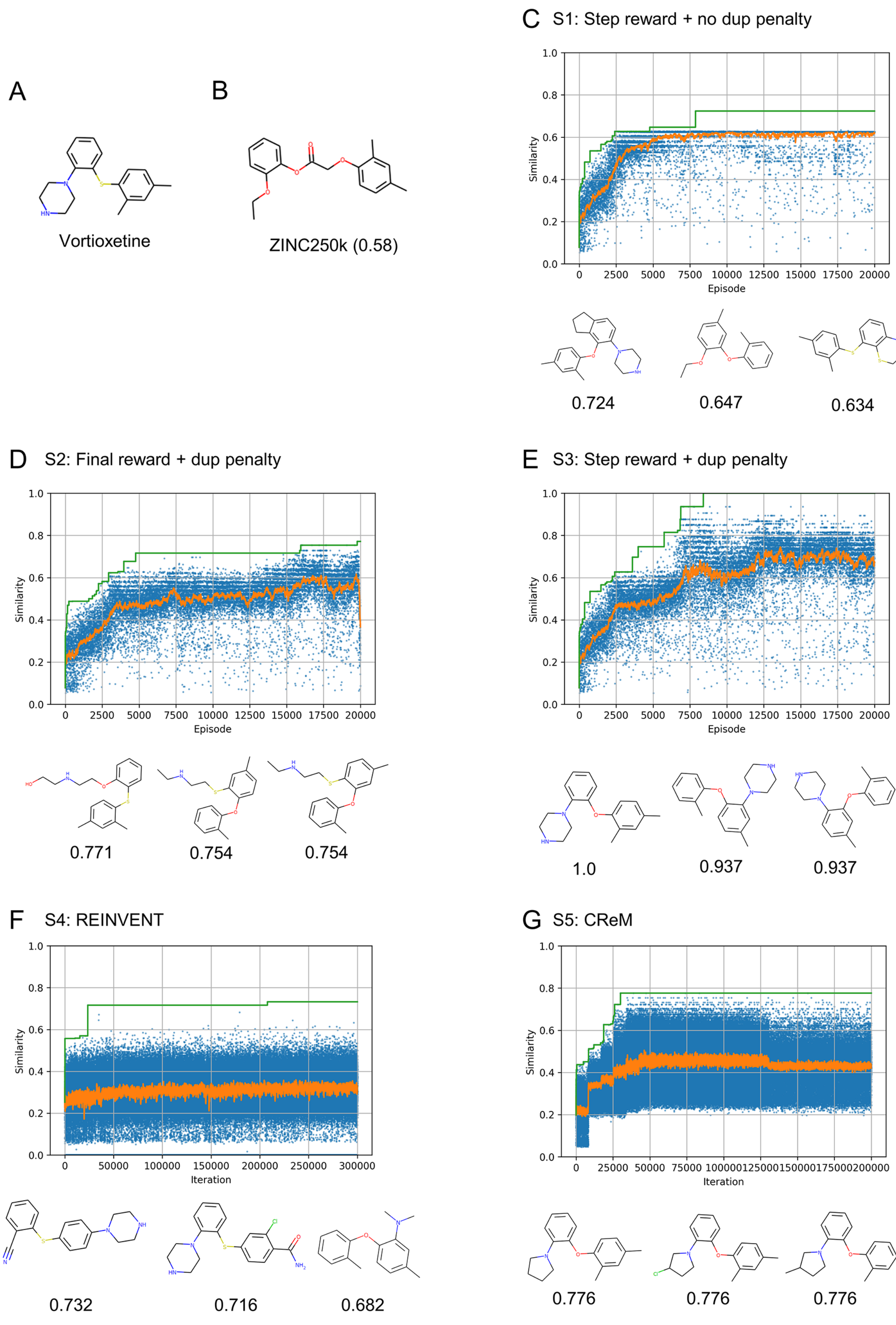


Figure 4. Compounds generated by the similarity optimization experiment targeting vortioxetine. (A) Chemical structure of vortioxetine used in the query structure in the experiment. (B) Compound most similar to vortioxetine in the ZINC250k dataset with its similarity score in parentheses. (C−G) Results of similarity optimization experiments, S1−S5 (see Table 2). The similarity scores for each episode (or iteration) are plotted in blue, and the moving average and maximum values are plotted in orange and green, respectively. The chemical structures of compounds with the highest scores (top three) are shown. Similarity scores are noted below the chemical structures.

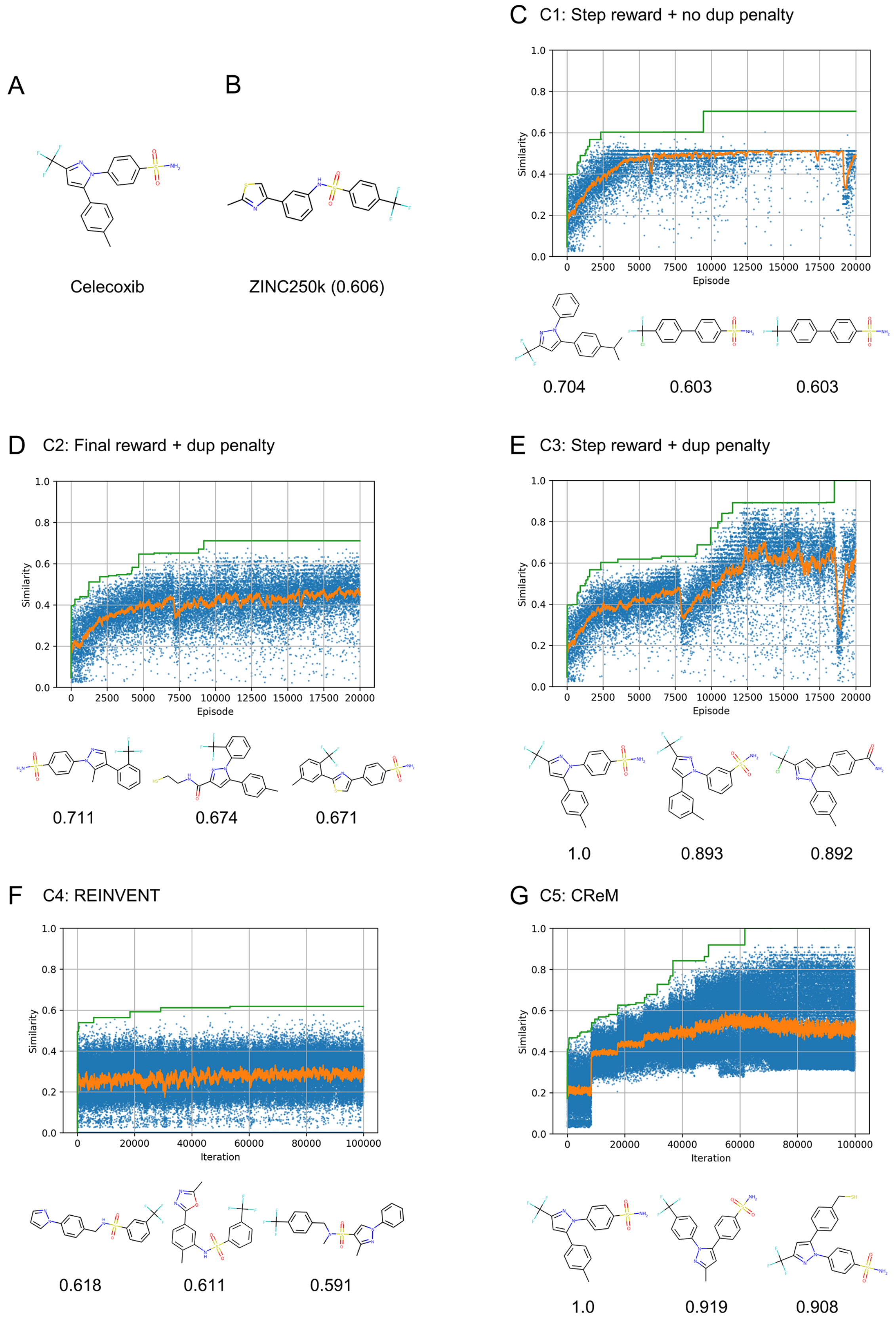


Figure 5. Compounds generated by the similarity optimization experiment targeting celecoxib. (A) Chemical structure of celecoxib used in the query structure of the experiment. (B) Compound most similar to celecoxib in the ZINC250k dataset with its similarity score in parentheses. (C–G) Results of the similarity optimization experiments, C1–C5 (see Table 2). The similarity scores for each episode (or iteration) are plotted in blue, and the moving average and maximum values are plotted in orange and green, respectively. The chemical structures of compounds with the highest scores (top three) are shown. Similarity scores are noted below the chemical structures.

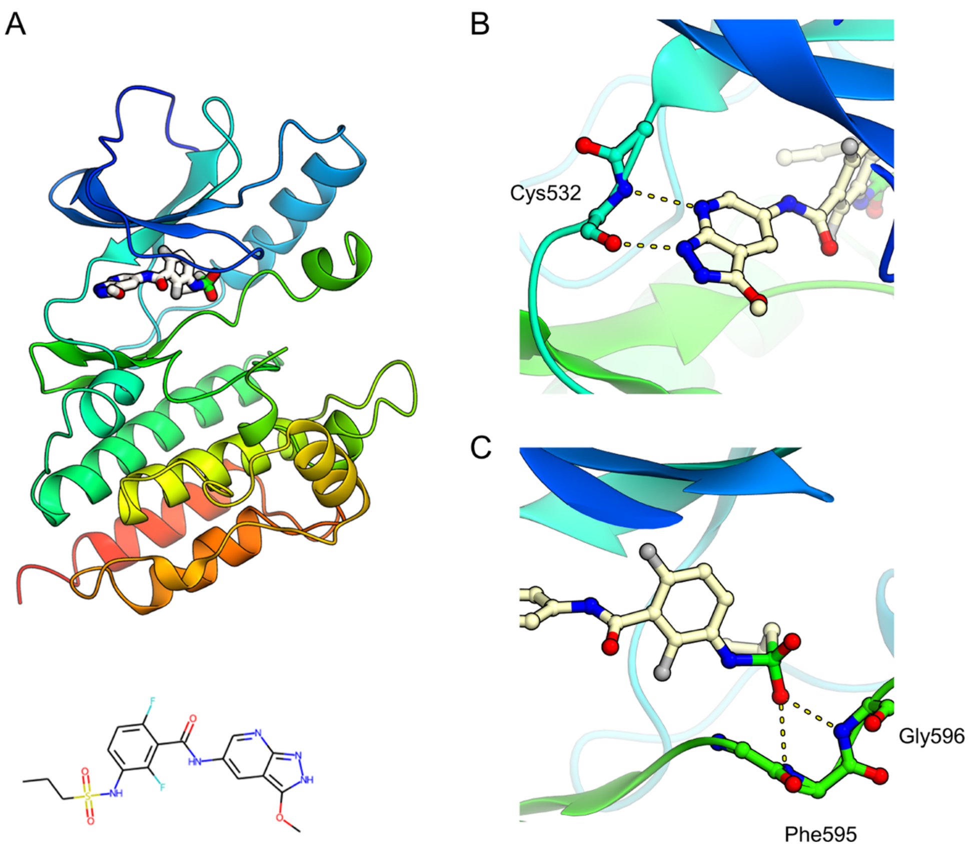


Figure 6. B-Raf kinase with its inhibitor complex[35] (PDB ID: 3TV6) was used as the target structure in the structure-based scaffold-hopping experiments. (A) Overall structure, (B) interactions between the kinase hinge motif and the inhibitor, and (C) interactions around the sulfonamide moiety of the inhibitor. The protein backbone and bound inhibitor are presented using ribbon and ball-and-stick models, respectively. Molecular graphics were prepared using CueMol (http://www.cuemol.org/).

the optimization target score decreases in the step generating the intermediate alcohol moiety. Therefore, the landscape of the fingerprint-based similarity score is not smooth but contains many local minima wherein simple optimization algorithms can get trapped. This local-minima problem of fingerprint similarity could be avoided using the final reward (*i.e.*, evaluating only the final molecule in the episodes). However, this makes it impossible to evaluate each RL state value in the episodes. To evaluate the effectiveness of different reward settings, we examined three different cases of RJT-RL (Table 2).

Vortioxetine Rediscovery Experiment. In the first experiment (S1–S5; Table 2), vortioxetine was used as the query structure (Figure 4A). The most similar molecule in the pretraining dataset is shown in Figure 4B, with a similarity of 0.58. The results of the vortioxetine rediscovery experiment are summarized in Figures 4C–G and S4. In case S1, optimization was performed with a step reward and without a duplication penalty (Table 2), thereby achieving a similarity score of 0.724 (Figure 4C). However, after convergence to a local minimum, the exploration efficiency suddenly deteriorated, and similar or the same molecules were repeatedly generated (uniqueness in Table 2 and Figure S4A). In case S2, when only the final reward and duplication penalty were applied (Table 2), molecules with a similarity score of approximately 0.5 were continuously generated (Figure 4D), and a highest similarity of 0.771 was achieved (Figure 4D). In case S3, with the application of step reward and duplication penalty, diverse molecules with high similarity scores were continuously generated (Figure 4E). The average similarity score between the last 1000 episodes was 0.681, and a structure that was almost identical to the query structure was generated (Figure 4E). The best and average performances for S3 outperformed those for S2, demonstrating that the step-by-step evaluation of the molecular properties successfully guided the generation of optimum molecules. Additionally, several runs were conducted with different random seeds for S1–S3 (Figure S5), and it was observed that the duplication penalty strongly facilitated optimization in all cases. The step reward (S3) tends to produce better results than the final reward (S2), but it is influenced by random states to some extent. These results suggest that the advantage of step reward outweighs the disadvantage arising from the local minima in this problem.

In S4, REINVENT also generated structures that were similar, but not identical, to the query structure, with the highest similarity score of 0.732 (Figure 4F). However, structures with high rewards were generated infrequently, and an average similarity score of approximately 0.32 was obtained at the end of training (Figure 4F). This suggested that the agent failed to adequately explore the chemical space around the query structure. In S5, CReM also generated similar structures, with the highest similarity score of 0.776 (Figure 4G). The average similarity score at the end of the optimization was approximately 0.5 (Figure 4G). However, it requires many evaluations (approximately 50,000) of the score function as compared to the other experiments (approximately 10,000).

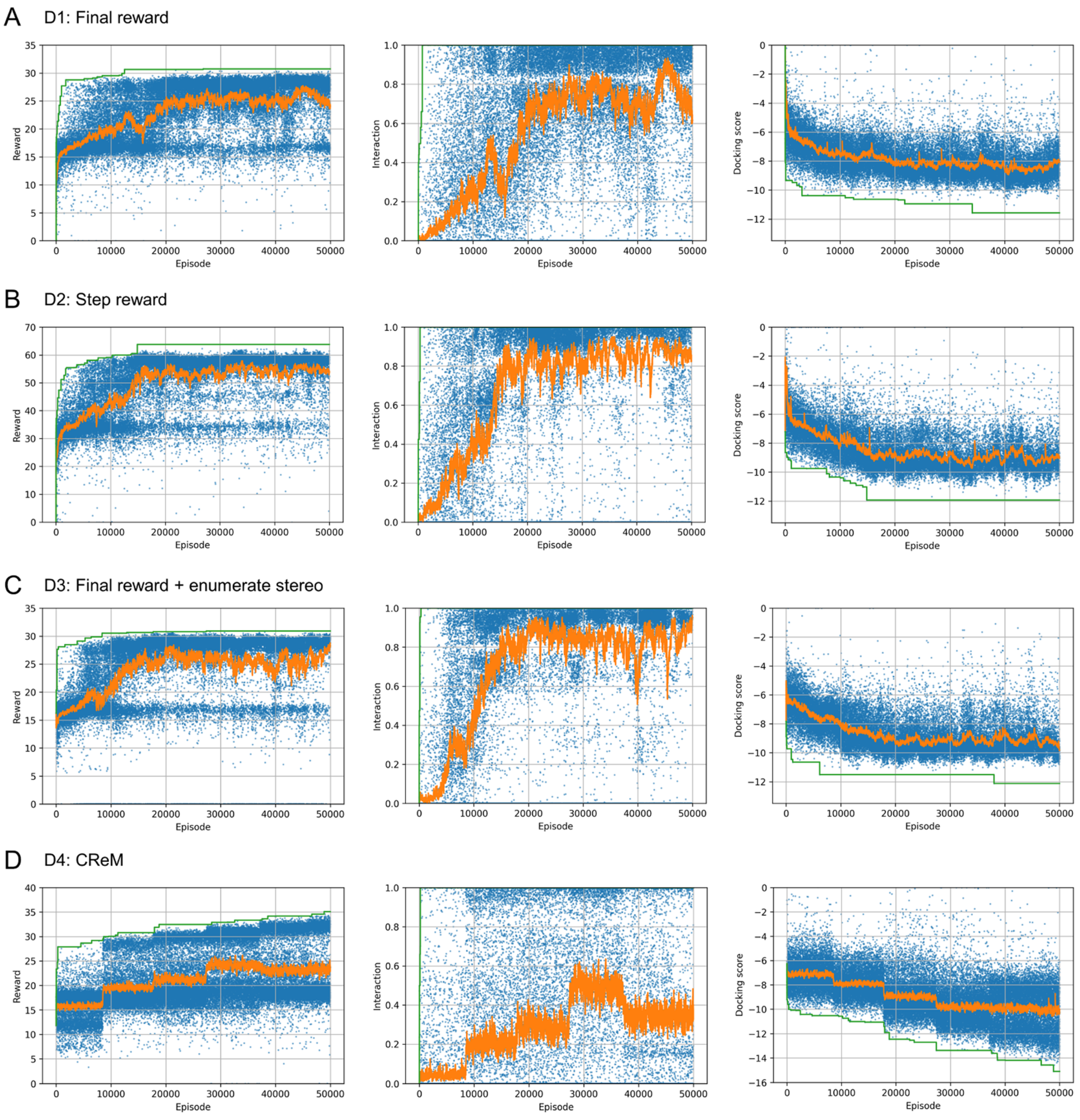


Figure 7. Results of the structure-based scaffold-hopping experiments, D1−D4 (see Table 3). The reward (left panel), interaction (middle panel), and docking (right panel) scores for each episode are plotted in blue, and the moving average and maximum values are plotted in orange and green, respectively.

Celecoxib Rediscovery Experiment. Next, similarity-guided optimization of celecoxib, which is also included in the GuacaMol benchmark suite,[34] was performed to observe the effects of the query structures (Figure 5A). For the experiment, RJT-RL, REINVENT, and CReM were used under conditions similar to those for vortioxetine (C1−C5; Table 3). The molecule most similar to celecoxib in the pretraining dataset is shown in Figure 5B, with a similarity of 0.606.

The results of the experiment are summarized in Figures 5C−G and S6. In RJT-RL, C3 (step reward) performed better than C1 and C2, as observed in the vortioxetine experiments (S1−S3). C3 generated a structure identical to celecoxib with a similarity of 1.0, but some effect of the random state on the generated structures was observed (Figure S7). CReM (C5) also successfully generated a structure identical to celecoxib after optimization of the 8th generation with an evaluation of approximately 60,000 compounds (Figure 5G). This result suggests that CReM is also a powerful method but requires a large number of score evaluations. In contrast, the best molecule generated by REINVENT (C4) has a similarity of 0.62 (Figure 5F), which is slightly better than the most similar structure in the dataset (Figure 5B). The original paper on

REINVENT[7] stated that it generated an identical structure to celecoxib under similar conditions but with a larger size (1.5 million) of the pretraining dataset. REINVENT may require a larger pretraining dataset to achieve the best performance.

Structure-Based Scaffold Hopping. We further examined the efficiency of RJT-RL in the realistic task of structure-based scaffold hopping, wherein known drug candidate compounds with structural information (*i.e.*, important interactions between the target protein and compounds) are available. B-Raf kinase with its inhibitor complex structure[35] (PDB ID: 3TV6, Figure 6A) was used as an example. In this structure, the pyrimidine nitrogen of the inhibitor hydrogen bonds with the main-chain nitrogen atom of Gly 596 (Figure 6B) in the kinase hinge region. In addition to this hinge interaction, this inhibitor interacts with the main-chain nitrogen atom of Cys 532 through its sulfonamide group (Figure 6C). In practical structure-based drug discovery applications, it is crucial to search for molecules with different scaffolds that retain known important interactions with target proteins. Thus, we attempted to design novel compounds that preserved the interaction between the hinge region of the enzyme and the pyrimidine group in the compounds while changing the scaffold in the compound interacting with Cys 532.

In this task (D1–D4; Table 3), we used the docking score and the interaction score pertaining to the hydrogen-bond acceptor of the compound and the main-chain nitrogen atoms of Gly 596 and Cys 532. The ETKDG algorithm implemented in RDKit was used to generate three-dimensional (3D) conformations from two-dimensional (2D) structures.[36] In this process, we randomly selected one stereoisomer from all possible stereoisomers except for D3, wherein we enumerated all possible stereoisomers up to 16, attempted docking simulations, and then selected the isomer with the best docking score. The interaction score is defined as follows

$$\mathrm{Intr}(p,\ c) = \begin{cases} \dfrac{d(p,\ c) - d_{\min}}{d_{\max} - d_{\min}}, & d_{\min} < d(p,\ c) \le d_{\max} \\ 1, & d_{\max} < d(p,\ c) \\ 0, & d(p,\ c) \le d_{\min} \end{cases}$$

where $d(p, c)$ is the distance between the specified atoms of the protein and the compound. In this experiment, $d_{\max}$ = 6.0 Å and $d_{\min}$ = 3.0 Å were used. The total reward function is defined as follows

$$\mathrm{Intr} = \mathrm{Intr}(\mathrm{G596} - \mathrm{N},\ \mathrm{HBA}) + \mathrm{Intr}(\mathrm{C532} - \mathrm{N},\ \mathrm{HBA})$$

$$R_{\mathrm{fin}} = \mathrm{Intr} \times c_{\mathrm{intr}} - \Delta G$$

where HBA is the hydrogen-bond acceptor atom in the compound, ΔG is the AutoDock Vina score[37] of the docking pose of the compound, and c_{intr} is a hyperparameter. Given the scale of the typical docking score (around −10) and interaction score (0–1) terms, c_{intr} = 10.0 was used in this experiment. The docking score was clipped by 500 because the PPO algorithm was destabilized by an extremely high value of the docking score. The pyrimidine group was used as the initial state for the RL training. The initial binding pose of the pyrimidine group to the protein was restrained to the same as that in the crystal structure. Two types of reward functions were compared as in the other experiments (Table 3). In cases D1 and D3, the aforementioned reward function was used for the final step, while $R_{\mathrm{step}}(t)$ = 0 was maintained for all other intermediate steps. In case D2, the same function was used for $R_{\mathrm{step}}(t)$. Furthermore, for comparison with the rule-based methods, CReM was examined using the same scoring function, including the docking and interaction scores (D4; Table 3).

The results for D1–D3 demonstrated that RJT-RL successfully generated optimized molecules with the desired interactions (Figure 7A–C). Both the interaction and docking scores were gradually optimized during the RL training episodes, and high-score compounds were continuously sampled during the later training episode stages. However, the distributions of the Log *P*, SA, and QED[38] scores of the generated compounds tended to deviate from those of the training dataset (Figure S8A–C). Compounds with high docking scores that satisfied the desired interactions are shown in Figure S9. Next, we compared the results of the final and step rewards (D1 and D2, respectively). The step reward was marginally better than the final reward. D2 obtained a higher reward than D1 for the same number of episodes and converged to the best score faster than D1 (Figure 7A,B). This result also demonstrates the potential of the step-by-step evaluation of the molecular properties. However, the speed of D2 was much lower than that of D1 because the docking simulation was performed for each RL step in D2. Notably, D2 took approximately seven times more time than D1 to reach the same number of episodes (Table 5).

Table 5. Average Computational Time Required for Generating One Molecule

	reward	3D conf	average time (s)
D1	final	random	1.74
D2	step	random	12.9
D3	final	enum	8.96

In D3, we enumerated all stereoisomers, selected the one with the best docking score, and used the final reward function. This result demonstrates that the enumeration of stereoisomer accelerated the speed of optimization compared to that of D1 (Figure 7C). However, the speed of D3 was approximately four times lower than that of D1, as it performed up to 16 docking simulations per episode (Table 5). In D1 and D2, the randomly determined stereoisomers in the 3D-embedding process may result in different docking scores even though the agent performed the same actions. This random selection of stereoisomers may affect the optimization speed.

Next, we compared the results for RJT-RL (D1) and CReM (D4). Because the score evaluation, including the docking simulation, is a bottleneck in this task, we considered one score evaluation in CReM corresponding to one episode in RJT-RL with a final reward. The results show that the generated molecules are optimized against the target score function at a speed comparable to that of D1 (Figure 7D). However, the distribution of each score term is quite different from that of RJT-RL (D1; Figure 7A). In D3, the docking score term is well optimized, while the interaction term is less optimized than in D1 (Figure 7D). One possible reason for this unbalanced optimization is the tendency of CReM to generate large molecules (Figures S8D and S9D). This tendency may increase the van der Waals interaction terms in the Vina scoring function,[37] thereby causing an unbalanced optimization. This could be avoided by optimizing c_{intr}, the balancing

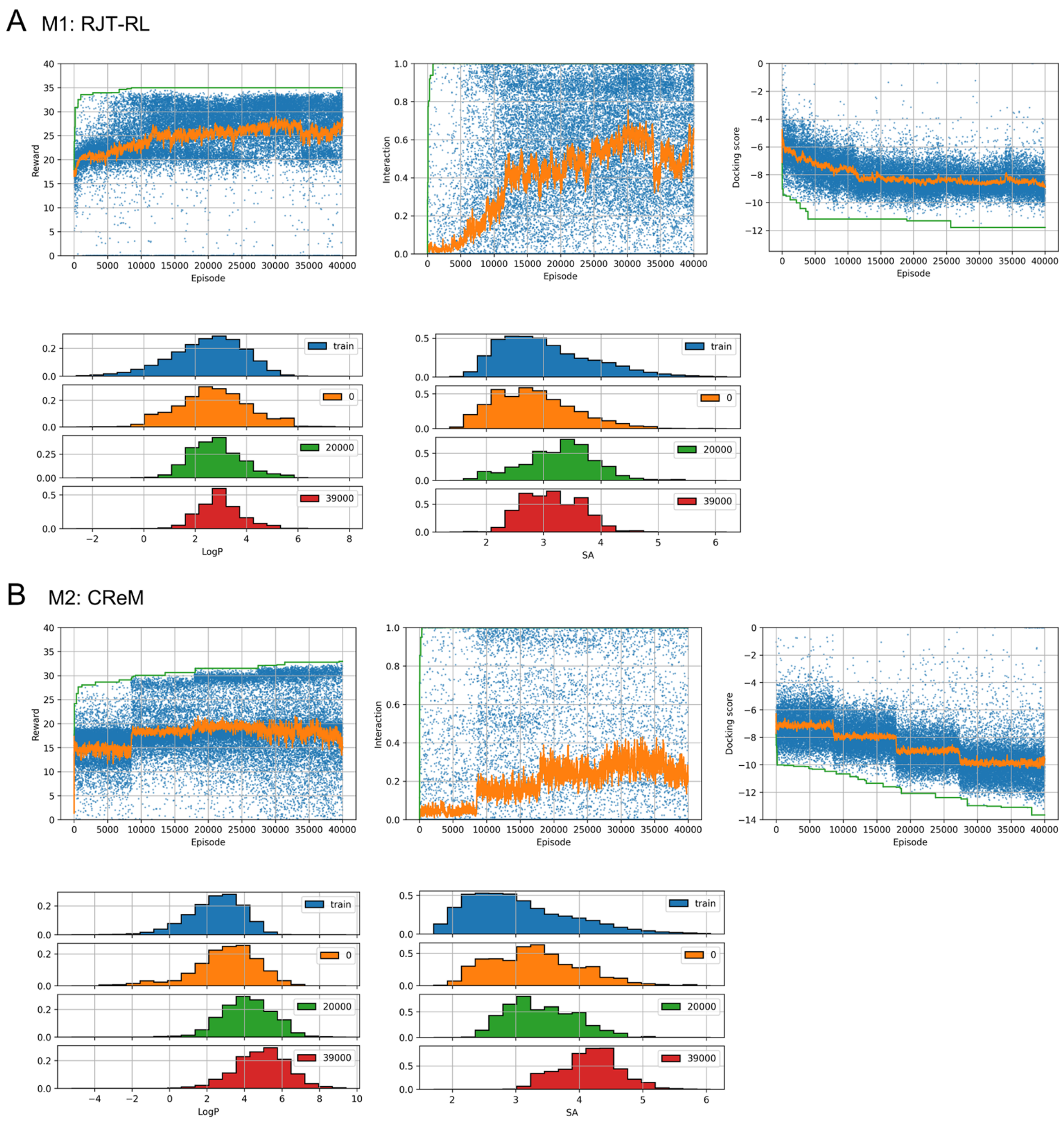
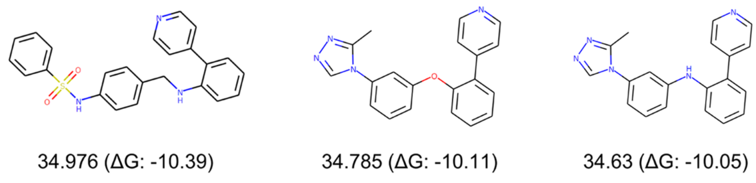


Figure 8. Results of the multiobjective reward experiments (A) M1 and (B) M2 (see Table 3). In the upper panel, the reward, interaction, and docking scores for each episode are plotted in blue, and the moving average and maximum values are plotted in orange and green, respectively. In the lower panel, histograms of the Log *P* and SA score distributions of the training dataset (blue) and episodes 0−1000 (orange), 20,000−21,000 (green), and 39,000−40,000 (red) are plotted.

hyperparameter of the score terms, or using the ligand efficiency[39] instead of the docking score.

Finally, we examined several runs with different random seeds for D1, D2, and D3 (Figure S10). In all cases, the optimization proceeds similarly, but the convergence speeds differ depending on the runs in D1 and D2. In particular, in another run of D2, the speed of convergence is observed to be similar to that of D1 (Figure S10B). Next, we examined the differences in the molecules generated from different runs. We calculated the pairwise similarities of the top 100 molecules from the two runs and plotted their distributions (Figure S10D). The results demonstrate that diverse molecules are generated from different runs with the same hyperparameters. This could be an advantage of RJT-RL because it is important to obtain a variety of compounds with high target scores in actual drug design tasks. Consequently, given the level of improvement and deterioration by the step reward (D2) and



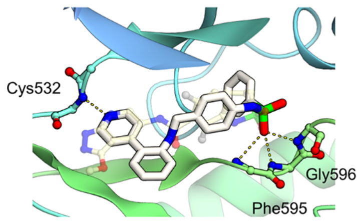


Figure 9. Compounds were generated in a multiobjective reward experiment, M1 (see Table 3). The chemical structures of the top three compounds, with their rewards and docking scores, are shown in the top panels. The binding poses of the best compounds are shown in the bottom panel. The protein backbone and bound inhibitor are shown using the ribbon and transparent ball-and-stick models, respectively. The original compound bound to the crystal structure is shown as semitransparent. The possible hydrogen-bonding interactions are indicated by yellow dashed lines.

stereoisomer enumeration (D3), we conclude that the final reward (D1) is best under the current experimental settings.

Multiobjective Optimization. While the above cases targeted simple rewards with one or two objectives, a typical drug design task requires the improvement of multiple objective functions. In particular, the compounds generated in cases D1−D3 tended to contain highly hydrophobic and/or possibly unstable structures (Figures S8A−D and S9), as also evident from the low "filter" score of D1−D3 in Table 3. These unfeasible structures can be suppressed using multiobjective target functions, including penalties for unwanted structures. In M1, we introduced SA[31] and Log *P* scores to penalize such unfeasible structures and examined the performance of our method in multiobjective optimization tasks. We define two penalty terms as follows.

$$p_{\mathrm{SA}} = \max(\mathrm{SA} - 4,\ 0)$$

$$p_{\mathrm{Log}\,P} = \max(|\mathrm{Log}\,P - 2.5| - 2.5,\ 0)^2$$

where SA and Log *P* are the SA and Log *P* scores, respectively, of the target molecule. Thus, these terms are intended to restrict the SA score to less than 4 and the Log *P* score to the range of 0−5. The total reward function is defined as follows

$$R_{\mathrm{fin}} = \mathrm{Intr}\cdot c_{\mathrm{intr}} - \Delta G - p_{\mathrm{SA}}\cdot c_{\mathrm{SA}} - p_{\mathrm{Log}\,P}\cdot c_{\mathrm{Log}\,P}$$

$$R_{\mathrm{step}} = (7 - \mathrm{SA})\cdot c_{\mathrm{SA}} - p_{\mathrm{Log}\,P}\cdot c_{\mathrm{Log}\,P}$$

where $c_{\mathrm{intr}} = 10$, $c_{\mathrm{SA}} = 1$, and $c_{\mathrm{LogP}} = 5$ were used. We standardized and neutralized the compounds using the functions implemented in RDKit[18] before generating the 3D conformations. Additionally, we examined the CReM generator with similar settings, except for the target score function, which is defined as R_{fin} above.

The result of M1 demonstrated that the multiobjective reward function increased similar to the cases D1−D3, and the distributions of SA and Log *P* values were restricted to the specified ranges (Figures 8A and S11). In D1, molecules with high SA scores (SA > 4) were frequently generated (Figure S8A), whereas the generation of such molecules was suppressed in M1 (Figure 8A). The Log *P* value of the generated molecules was also restricted to the specified range compared to those in the D1 experiments (Figures 8A *vs* S8A). Consequently, the "filter score" in M1 was drastically improved compared to that of D1 (Table 3). The top three compounds in terms of reward and docking pose of the best compound are shown in Figure 9. Although the second and third compounds in Figure 9 are similar, the diversity of the generated structures of M1 can be observed from their statistics and plots (Table 3 and Figure S11). Collectively, these results demonstrate that RJT-RL can be successfully applied to multiobjective optimization. Moreover, another run of M1 was performed, and a similar tendency in the property distributions was observed (Figure S12). A similarity plot between these runs (Figure S12D) demonstrated that diverse molecules were generated from the different runs in this case.

Next, the results for RJT-RL (M1) and CReM (M2) were compared. CReM also successfully improved the overall score (Figure 8B). However, an unbalanced optimization of the score terms was observed, similar to the case of D4 (Figure 7D). Although the docking score was improved by optimization, the other terms (*i.e.*, interaction, SA, and Log *P* terms) were less improved than those of M1 (Figure 8). Even though the increasing tendency of the SA score was suppressed, a large number of compounds with SA > 4.0 were still generated (Figure 8B). As for the Log *P* property, D4 generated more molecules with Log *P* in the specified range (0 < Log *P* < 5) than M2 (Figures S8D and 8B). To perform multiobjective optimization with CReM, we may have to perform an intensive hyperparameter search for the best weighting terms (c_{intr}, c_{SA}, and c_{LogP}) in the scoring function and/or to introduce another penalty term for large molecules.

Fine-Tuning to Different Reward Functions. One of the advantages of learning-based methods is their transferability to various tasks. To assess the transferability of the RJT-RL models (policy and value function), a fine-tuning experiment, F1, using different reward functions (Table 3) was performed. In this experiment, we attempted fine-tuning the model trained in D1 to the multiobjective reward function defined in M1.

The results of F1 are shown in Figures 10 and S13. From the episodes in the early training stage, the agent generated molecules with high docking and interaction scores, which were maintained during the training (Figure 10). The SA and

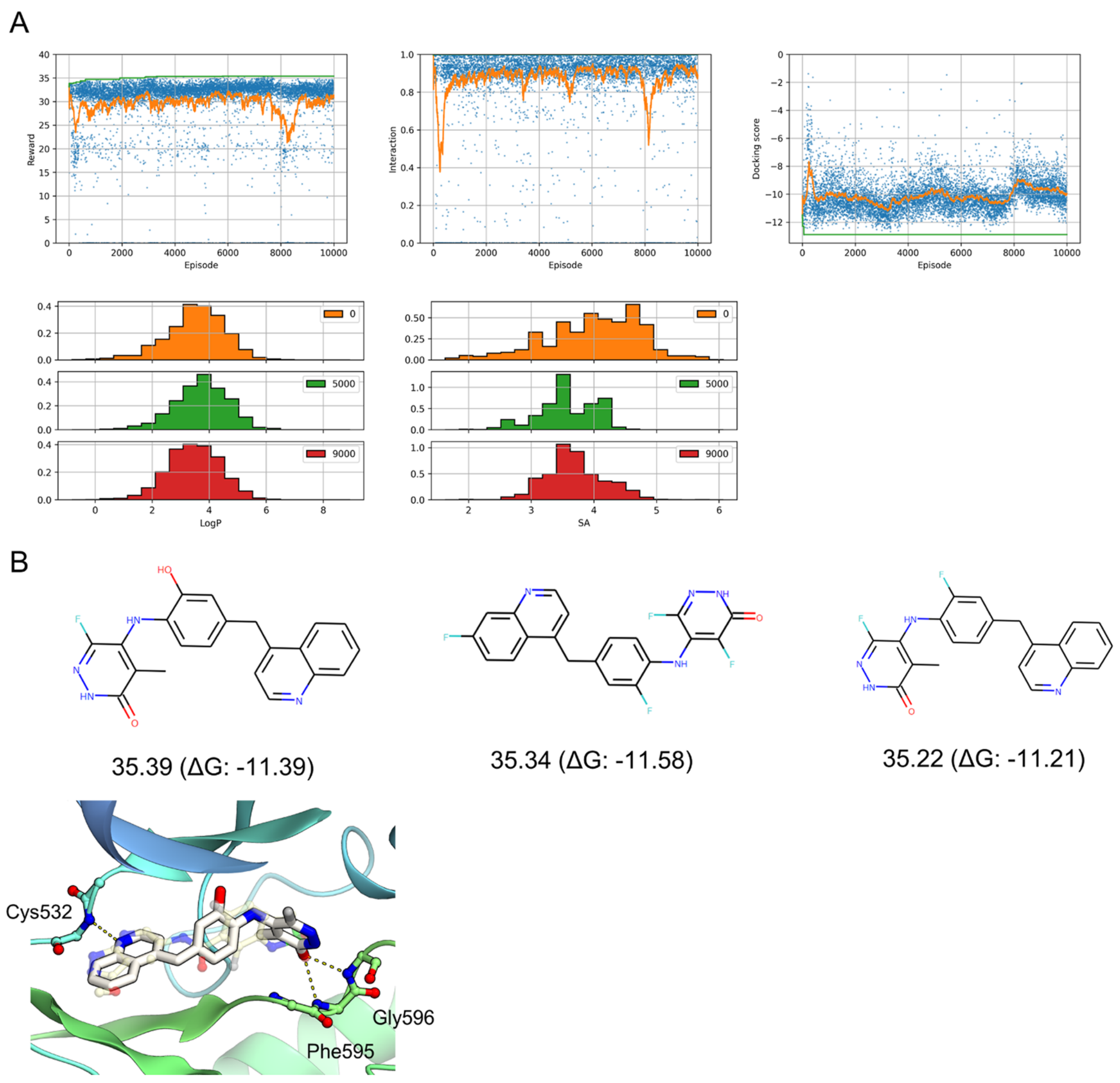


Figure 10. Results of fine-tuning experiment F1 (see Table 3). (A) In the upper panel, the reward, interaction, and docking scores for each episode are plotted in blue, and the moving average and maximum values are plotted in orange and green, respectively. In the lower panel, histograms of the Log *P* and SA score distributions of episodes 0−1000 (orange), 5000−6000 (green), and 9000−10,000 (red) are plotted. (B) Chemical structures of the top three compounds with their rewards and docking scores are shown in the top panels. The binding poses of the best compounds are shown in the bottom panel as in Figure 9.

Log *P* scores also gradually improved as the training proceeded, and their distributions fell within the range specified by the penalty score (*i.e.*, SA < 4 and 0 < Log *P* < 5) (Figure 10). The filter score of F1 was improved from that of D1 (Table 3), suggesting that the distribution of the generated compounds contained more feasible structures than before fine-tuning. In contrast, the diversity of molecules generated by F1 was slightly lower than that of M1, as evident from their statistics (uniqueness and IntDev in Table 3). The top three compounds in terms of reward and the docking pose of the best compound are shown in Figure 10. This result demonstrates that the agent adapted to the different reward functions containing the penalty terms after fewer training steps than those in M1.

■ CONCLUSIONS

In this study, we introduced RJT, a coarse-grained representation of molecules, which is directly convertible to the original chemical structure. We leveraged this representation for drug discovery, formulating a molecule design task as reinforcement learning to generate an RJT. The proposed method exhibited a better or comparable performance to other state-of-the-art methods in simple benchmark tasks. The results indicate the potential of the step-by-step evaluation of molecular properties, which is only possible in our framework using RJT. The step reward was shown to be advantageous when the computational cost of the score calculation was low because the score required multiple evaluations to generate a single compound. We further demonstrated that our method is applicable to real-world tasks such as structure-based scaffold hopping with a multiobjective reward function. Another advantage of RTJ-RL is the fine-tuning of the policy and value function models, which is not possible with rule-based fragment growth methods. This feature may be useful for tuning scoring functions in the compound design process.

The results of the experiments also suggest several problems with the proposed method. For example, considering the 3D generation involving docking simulation, efficient training of the RJT-RL model requires proper handling of the stereoisomers of the generated compounds. In this study, we enumerated the possible stereoisomers and searched for the best stereoisomers using the brute-force method. However, the search was performed on a single CPU, which reduced the overall performance (Table 5). Parallelization of the 3D conformer generation and docking simulation could improve performance because this calculation is embarrassingly parallelizable. Another solution to this problem is to extend the site information in the RJT representation to properly handle the chirality flag of the atom so that the agent can generate the molecule, including the stereoisomers. Finally, in this study, we only considered the de novo design of compounds from a relatively small starting fragment size. To apply the proposed method for the optimization of lead compounds with larger and more complex structures, it is necessary to extend the action to include node deletion and mutation to enable the modification of the starting scaffolds.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.2c00366.

Supplementary Figures S1−S13 (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Ryuichiro Ishitani − *Preferred Networks, Inc., Tokyo 100-0004, Japan;* orcid.org/0000-0002-4136-5685; Email: ishitani@preferred.jp

Authors

Toshiki Kataoka − *Preferred Networks, Inc., Tokyo 100-0004, Japan*

Kentaro Rikimaru − *Preferred Networks, Inc., Tokyo 100-0004, Japan*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.2c00366

Author Contributions

R.I., T.K., and K.R. designed the study, performed computational experiments, analyzed the data, and wrote the manuscript. R.I. directed and supervised this study.

Funding

The authors received no funding sources for this study.

Notes

The authors declare no competing financial interest.

The materials supporting this article, including the source code, sampled molecules, and pretrained models, are available at github (https://github.com/pfnet-research/RJT-RL) under the MIT license.

■ ACKNOWLEDGMENTS

The authors thank Shin-ichi Maeda and Yasuhiro Fujita (Preferred Networks, Inc.) for their helpful discussions regarding RL. They also thank Kazuya Ujihara (Preferred Networks, Inc.) for helpful discussions regarding evaluation methods for the generated compounds.

■ REFERENCES

(1) LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436−444.

(2) Bilodeau, C.; Jin, W.; Jaakkola, T.; Barzilay, R.; Jensen, K. F. Generative Models for Molecular Discovery: Recent Advances and Challenges. *WIREs Comput. Mol. Sci.* **2022**, No. e1608.

(3) Meyers, J.; Fabian, B.; Brown, N. De Novo Molecular Design and Generative Models. *Drug Discovery Today* **2021**, *26*, 2707−2715.

(4) Kingma, D. P.; Welling, M. Auto-Encoding Variational Bayes, arXiv: 1312.6114. arXiv.org e-Print archive. https://arxiv.org/abs/1312.6114 (submitted Dec 20, 2013).

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

(6) Tripp, A.; Daxberger, E.; Hernández-Lobato, J. M. Sample-Efficient Optimization in the Latent Space of Deep Generative Models via Weighted Retraining, arXiv: 2006.09191. arXiv.org e-Print archive. https://arxiv.org/abs/2006.09191 (submitted Jun 16, 2020).

(7) Olivecrona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular De-Novo Design through Deep Reinforcement Learning. *J. Cheminf.* **2017**, *9*, No. 48.

(8) You, J.; Liu, B.; Ying, R.; Pande, V.; Leskovec, J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, arXiv:1806.02473. arXiv.org e-Print archive. https://arxiv.org/abs/1806.02473 (submitted Jun 7, 2018).

(9) Zhou, Z.; Kearnes, S.; Li, L.; Zare, R. N.; Riley, P. Optimization of Molecules via Deep Reinforcement Learning. *Sci. Rep.* **2019**, *9*, No. 10752.

(10) Polishchuk, P. CReM: Chemically Reasonable Mutations Framework for Structure Generation. *J. Cheminf.* **2020**, *12*, No. 28.

(11) Chéron, N.; Jasty, N.; Shakhnovich, E. I. OpenGrowth: An Automated and Rational Algorithm for Finding New Protein Ligands. *J. Med. Chem.* **2016**, *59*, 4171−4188.

(12) Clark, D. E.; Frenkel, D.; Levy, S. A.; Li, J.; Murray, C. W.; Robson, B.; Waszkowycz, B.; Westhead, D. R. PRO-LIGAND: An Approach to de Novo Molecular Design 1. Application to the Design of Organic Molecules. *J. Comput.-Aided Mol. Des.* **1995**, *9*, 13−32.

(13) Stahl, M.; Todorov, N. P.; James, T.; Mauser, H.; Boehm, H.-J.; Dean, P. M. A Validation Study on the Practical Use of Automated de Novo Design. *J. Comput.-Aided Mol. Des.* **2002**, *16*, 459−478.

(14) Böhm, H.-J. LUDI: Rule-Based Automatic Design of New Substituents for Enzyme Inhibitor Leads. *J. Comput.-Aided Mol. Des.* **1992**, *6*, 593−606.

(15) Böhm, H.-J. The Computer Program LUDI: A New Method for the de Novo Design of Enzyme Inhibitors. *J. Comput.-Aided Mol. Des.* **1992**, *6*, 61−78.

(16) Jin, W.; Barzilay, R.; Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation, arXiv: 1802.04364. arXiv.org e-Print archive. https://arxiv.org/abs/1802.04364 (submitted Feb 12, 2018).

(17) Jin, W.; Barzilay, R.; Jaakkola, T. Hierarchical Graph-to-Graph Translation for Molecules, arXiv:1907.11223. arXiv.org e-Print archive. https://arxiv.org/abs/1907.11223 (submitted Jun 11, 2019).

(18) Landrum, G. *RDKit: Open-Source Cheminformatics Software*. http://rdkit.org/.

(19) Sutton, R. S.; Barto, A. G. *Reinforcement Learning: An Introduction*; The MIT Press, 2012.

(20) Sutton, R. S.; McAllester, D.; Singh, S.; Mansour, Y. In *Policy Gradient Methods for Reinforcement Learning with Function Approximation*, Adv Neural Inf Process Syst (NIPS 11), 1999.

(21) Williams, R. J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* **1992**, *8*, 229−256.

(22) Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation, arXiv:1506.02438. arXiv.org e-Print archive. https://arxiv.org/abs/1506.02438 (submitted Jun 8, 2015).

(23) Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms, arXiv:1707.06347. arXi-

v.org e-Print archive. https://arxiv.org/abs/1707.06347 (submitted Jul 20, 2017).

(24) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S. PyTorch: An Imperative Style High-Performance Deep Learning Library, arXiv:1912.01703. arXiv.org e-Print archive. https://arxiv.org/abs/1912.01703 (submitted Dec 3, 2019).

(25) Fujita, Y.; Nagarajan, P.; Kataoka, T.; Ishikawa, T. ChainerRL: A Deep Reinforcement Learning Library, arXiv:1912.03905. arXiv.org e-Print archive. https://arxiv.org/abs/1912.03905 (submitted Dec 9, 2019).

(26) Ostrovski, G.; Bellemare, M. G.; Oord, A.; van den Munos, R. Count-Based Exploration with Neural Density Models, arXiv:1703.01310. arXiv.org e-Print archive. https://arxiv.org/abs/1703.01310 (submitted Mar 3, 2017).

(27) Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; Munos, R. Unifying Count-Based Exploration and Intrinsic Motivation, arXiv:1606.01868. arXiv.org e-Print archive. https://arxiv.org/abs/1606.01868 (submitted Jun 6, 2016).

(28) Tang, H.; Houthooft, R.; Foote, D.; Stooke, A.; Chen, X.; Duan, Y.; Schulman, J.; de Turck, F.; Abbeel, P. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning, arXiv 1611.04717. arXiv.org e-Print archive. https://arxiv.org/abs/1611.04717v2?utm_campaign=Revue%20newsletter&utm_medium=Newsletter&utm_source=revue (submitted Nov 15, 2016).

(29) Polykovskiy, D.; Zhebrak, A.; Sanchez-Lengeling, B.; Golovanov, S.; Tatanov, O.; Belyaev, S.; Kurbanov, R.; Artamonov, A.; Aladinskiy, V.; Veselov, M.; Kadurin, A.; Johansson, S.; Chen, H.; Nikolenko, S.; Aspuru-Guzik, A.; Zhavoronkov, A. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Front. Pharmacol.* **2020**, *11*, No. 565644.

(30) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder, arXiv:1703.01925. arXiv.org e-Print archive. https://arxiv.org/abs/1703.01925 (submitted Mar 6, 2017).

(31) Ertl, P.; Schuffenhauer, A. Estimation of Synthetic Accessibility Score of Drug-like Molecules Based on Molecular Complexity and Fragment Contributions. *J. Cheminf.* **2009**, *1*, No. 8.

(32) Jaccard, P. Étude Comparative de La Distribution Florale Dans Une Portion Des Alpes et Des Jura. *Bull. Soc. Vaudoise Sci. Nat.* **1901**, *37*, 547−579.

(33) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50*, 742−754.

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

(35) Wenglowsky, S.; Ren, L.; Ahrendt, K. A.; Laird, E. R.; Aliagas, I.; Alicke, B.; Buckmelter, A. J.; Choo, E. F.; Dinkel, V.; Feng, B.; Gloor, S. L.; Gould, S. E.; Gross, S.; Gunzner-Toste, J.; Hansen, J. D.; Hatzivassiliou, G.; Liu, B.; Malesky, K.; Mathieu, S.; Newhouse, B.; Raddatz, N. J.; Ran, Y.; Rana, S.; Randolph, N.; Risom, T.; Rudolph, J.; Savage, S.; Selby, L. T.; Shrag, M.; Song, K.; Sturgis, H. L.; Voegtli, W. C.; Wen, Z.; Willis, B. S.; Woessner, R. D.; Wu, W.-I.; Young, W. B.; Grina, J. Pyrazolopyridine Inhibitors of B-Raf(V600E) Part 1: The Development of Selective Orally Bioavailable and Efficacious Inhibitors. *ACS Med. Chem. Lett.* **2011**, *2*, 342−347.

(36) Riniker, S.; Landrum, G. Better Informed Distance Geometry: Using What We Know To Improve Conformation Generation. *J. Chem. Inf. Model.* **2015**, *55*, 2562−2574.

(37) Trott, O.; Olson, A. AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function Efficient Optimization and Multithreading. *J. Comput. Chem.* **2009**, *30*, 2562−2574.

(38) Bickerton, G. R.; Paolini, G.; Besnard, J.; Muresan, S.; Hopkins, A. L. Quantifying the Chemical Beauty of Drugs. *Nat. Chem.* **2012**, *4*, 90−98.

(39) Hopkins, A. L.; Groom, C. R.; Alex, A. Ligand Efficiency: A Useful Metric for Lead Selection. *Drug Discovery Today* **2004**, *9*, 430−431.