MERMAID: An Open Source Automated Hit-To-Lead method based on Deep Reinforcement Learning

Daiki Erikawa,[†] Nobuaki Yasuo,[‡] and Masakazu Sekijima[*,†]

[†]*Department of Computer Science, Tokyo Institute of Technology, 4259–J3–23, Nagatsuta-cho, Midori-ku, Yokohama, Japan*

[‡]*Academy for Convergence of Materials and Informatics (TAC-MI), Tokyo Institute of Technology, S6–23, 2–12–1, Ookayama, Meguro-ku, Tokyo, Japan*

E-mail: sekijima@c.titech.ac.jp

Abstract

The hit-to-lead process makes the physicochemical properties of the hit compounds that show the desired type of activity obtained in the screening assay more drug-like. Deep learning-based molecular generative models are expected to contribute to the hit-to-lead process. The simplified molecular input line entry system (SMILES), which is a string of alphanumeric characters representing the chemical structure of a molecule, is one of the most commonly used representations of molecules, and molecular generative models based on SMILES have achieved significant success. However, in contrast to molecular graphs, during the process of generation, SMILES are not considered as valid SMILES. Further, it is quite difficult to generate molecules starting from a certain molecule, thus making it difficult to apply SMILES to the hit-to-lead process. In this study, we have developed a SMILES-based generative model that can be generated

starting from a certain compound. This method generates partial SMILES and inserts it into the original SMILES using Monte Carlo Tree Search and a Recurrent Neural Network. We validated our method using a molecule dataset obtained from the ZINC database and successfully generated molecules that were both well optimized for the objectives of the quantitative estimate of drug-likeness (QED) and penalized octanol-water partition coefficient (PLogP) optimization. The source code is available at `https://github.com/sekijima-lab/mermaid`.

Introduction

Approximately 8,000 drugs are currently being developed worldwide.[1] From drug discovery to launch, a new drug takes an average of 10 to 15 years to be developed and costs $2.6 billion.[1,2] Among the drug candidates that enter Phase I clinical trials, less than 12 % are approved by the Food and Drug Administration (FDA).[1]

After the target protein of a therapeutic drug for a disease has been determined, high-throughput screening (HTS) is used to exhaustively test the binding affinity of thousands to hundreds of thousands of compounds to the target protein in the search for hit compounds. Although the number of possible structures of a compound is 10^{60} and depends on the quality of the compound library to be tested, the hit rate of HTS is approximately 0.1 %,[3] which provides an opportunity to discover unexpected hit compounds but also highlights the problem of high experimental cost. To reduce the number of compounds to be tested, virtual screening, a computer-aided drug design (CADD) method for selecting new drug candidates, was proposed in the late 1990s.[4] In virtual screening, compounds that have a high potential to bind to a target protein are ranked in order from a database of thousands to millions of compounds using an evaluation function that expresses the binding affinity calculated by a computer. The compounds narrowed down by the virtual screening are verified by biochemical experiments,[5–7] and those that are actually determined to be active proceed to the hit-to-lead. Hit-to-Lead is a stage in early drug discovery where small molecule compounds

hit by high-throughput screening (HTS) are processed through certain optimizations to identify promising lead compounds.[8] In addition to simulation, machine learning (ML) methods such as random forests and deep learning have been used in virtual screening;[9–12] however, molecular design methods using generative models are expected to be used in hit-to-lead.[13]

The significant progress made in ML in recent years, especially in terms of deep learning, has led to a breakthrough in image processing and natural language processing.[14] Subsequently, various ML models have been applied in the field of molecular design and have shown impressive results.[15] Gomez-Bombarelli et al.[16] used a variational autoencoder (VAE) for molecular design. Representing the molecule as a continuous variable enables us to perform gradient-based optimization in latent space. Considering that simplified molecular input line entry system (SMILES) is a string, which is one of the representations of molecules, it is natural to employ recurrent neural networks (RNNs) for molecular design. Segler et al.[17] used long short-term memory (LSTM), which is an RNN, for molecule generation.

Although this method showed high validity, it is not suitable for the purpose of generating molecules with desirable properties. This is because LSTM training is only optimized to satisfy SMILES grammar, and the generation process does not consider the properties of molecules. Therefore, we have to repeat the generation process incessantly until we generate the desired molecules.

Xiufeng Yang et al.[18] used Monte Carlo tree search (MCTS) to generate desirable molecules with better efficiency than random sampling in RNN-based molecule generation. The aforementioned methods are SMILES-based molecular generative models, and they cannot take a specific molecule as a starting point during optimization tasks. This is because, unlike the case of molecular graphs, sub-SMILES cannot be considered as valid molecules owing to the nature of SMILES grammar.

Graph representation, which is called a molecular graph, is also a useful representation of molecules. Graph representation is easier to understand visually, and checking the valence allows all generated molecules to be valid. Various ML models such as generative

adversarial network (GAN)[19] and VAE[20] have been applied to the generation of molecular graphs.[21–23] These methods often outperformed SMILES-based methods in terms of optimization for certain chemical properties, as well as for metrics, such as validity and novelty, for generated molecules. However, handling molecular graphs on a computer is more difficult than SMILES. Because VAE deals with likelihoods explicitly, graph matching is necessary to calculate the loss function, which is a huge computational cost.[24] Although GAN do not deal with likelihoods explicitly, the GNN used in the discriminator and generator require a significant computational cost.[25] For these reasons, molecular graph-based approach can only deal with small molecules.[26] In addition, for both SMILES-based and molecular graph-based approaches, reinforcement learning is used to optimize specific chemical properties; however, the reward model, which is represented by neural networks and not always accurate(especially in the case of extrapolation),[27] needs to be retrained for each evaluation function.

To address these issues, we have introduced *MERMAID*, a generative model based on SMILES using MCTS and RNN. Our model can take a specific molecule as a starting point, and because we adopt SMILES representation, the restrictions on the size of molecules is not as stringent as the molecular graph-based approach, and our model does not require retraining of the model for each chemical properties.

Methods

MCTS for Molecular Generation

MCTS[28,29] is a model-based reinforcement learning approach used to solve large space planning problems by sampling episodes and constructing search trees. A node corresponds to a state s_i and has the value $Q(s)$, which represents the evaluation of itself and the number of visits $N(s)$. While sampling episodes, MCTS selects a node from a search tree using tree policies and evaluates a new node by rollout. Rollout is the default policy for simulations

without adding new nodes to search tree. The details of Rollout are described later.

In molecular generation, a node corresponds to a character of SMILES; therefore, hence, a path from the root node to leaf node corresponds to a SMILES. The MCTS algorithm includes the following four steps and iterates until some convergence condition is satisfied.

Selection

The purpose of this step is to select a node from a search tree for the expansion of nodes. Starting from the root node, the child node of current node is selected based on a tree policy. The tree policy is discussed later. Node selection is repeated recursively until the leaf node is selected.

Expansion

The purpose of this step is to expand the current node(the node selected in Selection step). Some SMILES characters following the current node's SMILES character is selected from predefined vocabulary.

Simulation

The purpose of this step is to evaluate added nodes. The evaluation of non-terminal nodes is a difficult task in reinforcement learning. Therefore, MCTS evaluates non-terminal nodes using the *Rollout* procedure. The *Rollout* procedure expands recursively from evaluating the nodes until the terminal node appears. When the terminal node appears, the path from the root node to the terminal node corresponds to a complete SMILES. Therefore, We can evaluate the path easily using some metrics such as QED, LogP. We used the score as the non-terminal node.

Backpropagation

The purpose of this step is to update the value $Q(s)$ and the number of visit $N(s)$ of traversed nodes in this episode. Starting from the evaluated node in the Simulation step, the parent node of the current node is updated using the calculated score r

recursively until the root node appears. The update formulas are defined as follows.

$$Q(s) \leftarrow \frac{Q(s)N(s) + r}{N(s) + 1} \tag{1}$$

$$N(s) \leftarrow N(s) + 1 \tag{2}$$

Tree policies are important for the performance of MCTS. The upper confidence bound (UCB) score, which is proposed for the multi-armed bandit problem, is often used as the tree policy. Node selection based on the UCB score is as follows.

$$\pi(s) = \arg\max_i \left\{ Q(s_i) + 2C_p \sqrt{\frac{\ln N(s_p)}{N(s_i)}} \right\} \tag{3}$$

where, $Q(s)$ is the mean estimated value of state s for its child nodes, and $N(s)$ is the number of visits to the state s. s_i and s_p are the states of each node i and the parent node, respectively. C_p is the hyperparameter of the bias term. The first term corresponds to exploitation, and the second term corresponds to exploration.

The advantage of using UCB score as the tree policy is that the probability of selecting sub-optimal actions converges to zero as the number of iteration tends to infinity under specific conditions (appropriate C_p and the value of reward ranges between 0 and 1). However, it is not possible for the number of iterations to tend toward infinity. Furthermore, the performance of rollout-based algorithms degrades similar to that of other algorithms.[30] Since a character following the SMILES string needs to satisfy SMILES grammar, the number of suitable characters that follow the SMILES string is smaller than the number of vocabularies, which is action space. Therefore, a few characters selected by RNN is considered as actions in our approach. C_p is chosen to be an upper bound of the accumulated reward in practice. However, in this case, C_p is large to an extent that the state space is restricted, i.e., exploration is considered more valuable than exploitation. Therefore, we set $C_p = \frac{1}{\sqrt{2}}$ like other studies[18] that using MCTS.

Inference of SMILES using RNN

RNN is a type of neural network and propagates information not only in the direction of layers but also that of time series. In this study, LSTM,[31] which is a type of RNN and superior to normal RNN in terms of longer dependency, is used to capture the features of SMILES grammar.

The role of RNN in MCTS is to select SMILES characters following an incomplete SMILES string in the expansion step and the default policy in the simulation step. In the expansion step, the incomplete SMILES string $s_1 s_2 ... s_t$ (encoded to $\boldsymbol{x_0 x_1 ... x_t}$, e.g., one-hot vector) corresponding to path from root to the selected node is the input for RNN. RNN receives the encoded sequence as input and outputs the probability of selecting characters following the input sequence, which is incomplete SMILES. The probability of selecting a character s^i based on output of RNN $\boldsymbol{y_t}$ is as follows.

$$P(s^i_{t+1}|s_1 s_2 ... s_t) = \frac{\exp(y^i_t)}{\sum_j \exp(y^j_t)} \tag{4}$$

Several characters are selected through a fixed number of samplings from the probability and are added as child nodes to the selected node in the selection step. In the simulation step, a SMILES character that follows the current node is selected recursively until the terminal character is selected in the same way as the expansion step.

Optimization of Specific Molecules

The purpose of this study is to generate the derivatives of specific molecules. The aforementioned MCTS-based generative model cannot start from specific molecules simply, because MCTS only adds nodes to the tail of a search tree. Our proposed method is as follows.

1. Extract partial SMILES strings from the initial point of SMILES.

2. Use MCTS to generate a series of characters representing the partial SMILES strings

3. Replace the extracted partial SMILES strings with the generated partial SMILES

This method is capable of generating molecules that are obtained by removing or adding a series of zeros or more SMILES characters from SMILES regarded as the starting point.

RNN Training

The role of MCTS is to generate partial SMILES string as described in above section. Therefore, RNN combined with MCTS should be trained with partial SMILES string rather than full SMILES. In this study, a dataset of partial SMILES strings was generated from a dataset of full SMILES. Preprocessing was done as follows.

1. Partial SMILES strings were extracted exhaustively from full SMILES of the original dataset.

2. "Invalid" partial SMILES strings were filtered. Partial SMILES strings are regarded as valid if the SMILES generated by inserting partial SMILES strings into C*C or C(*)C is valid.

Approximately 250,000 molecules were obtained from the ZINC database to train the RNN model and evaluate the proposed generative model. 90 % of the molecules were used for training, and the remaining were used to evaluate molecule generation.

The RNN model consists of two layers of LSTM and receives encoded SMILES strings as input and outputs sequences of probability that represents the suitability of a SMILES character following the current input sequence for each position and each SMILES character in vocabulary. This model was trained on the preprocessed training set for 20 epochs using the Adam optimizer[32] to minimize cross entropy loss.

Replacement of partial SMILES string

In the proposed method, the selection of partial SMILES strings removed from the initial point of SMILES is done in MCTS using the "Replacement" node. Fig. 2 shows how to

select removed partial SMILES strings. Specifically, the "Replacement" node, which is a child node of the root node, has the values of the starting position of a removed SMILES string and its length of that. A partial SMILES string is generated from the grandchild nodes of the root node, and the generated string is replaced with the part of the initial point of SMILES corresponding to the "Replacement" node.

Initial Point of Molecule

This approach is capable of generating new molecules from the original molecule. However, this approach has problem with generated molecules. Because this approach replaces the substructure generated by MCTS with a portion of the original molecule, the generated molecule is a molecule in which only one part of the original molecule has been changed on SMILES. In other words, the generated molecule is not modified in more than one place. Therefore, we propose a method that applies this approach multiple times. Specifically, for every fixed number of steps, the initial point of SMILES is replaced with the SMILES generated up to that point, and MCTS is performed from the beginning.

For efficient performance, it is important to know how the next initial point of SMILES is selected. In this study, we preferred to optimize SMILES with the maximum reward score, for example QED, LogP, etc. is selected as next initial point. The effect of the difference between fixed or changed initial points is investigated in the Experiment section.

This policy is simple and easy to understand and implement; however, it is possible that the generated molecules flow into a local solution. In addition, it is not necessary that future molecules that have desirable and better properties need to be generated from the best molecule among the generated molecules. Using this policy, good results are obtained in this study ; however, the selection policy for next initial SMILES must be further considered to generate better molecules. An overview of our entire methods is given in the Figure 3.

Experiment

We conducted two experiments to demonstrate the performance of our method. The first experiment is normal optimization, which modifies a molecule to maximize a single evaluation function. The optimization targets in this experiment were the QED[33] score and penalized LogP. QED is a measure of drug-likeness, and the more drug-like it is, the closer it is to 1 in a range between 0 and 1. Penalized LogP is defined as follows.

$$PLogP(mol) = LogP(mol) + SAscore(mol) + RingPenalty(mol) \tag{5}$$

The Penalized LogP consists of three terms: the normal LogP (octanol-water partition coefficient), the SA score that penalizes complex structures, and the penalties for large rings. Note that any other target property metrics that can be calculated from SMILES can be used in this model. The 200 lowest PLogP/QED molecules in the validation data set ware selected as the starting point in this experiment.

Molecule generation was done in 10,000 steps for each of the test molecules. We analyzed two cases for the proposed approach, depending on whether the starting point molecule is fixed. The model in which the starting point molecule is fixed is called "Single", and the other model is called "Multi". The starting point of the "Multi" model is replaced every 2000 steps with the highest scoring(PLogP/QED) molecule generated up to that point.

The second experiment is constrained optimization, which modifies a molecule to maximize a single evaluation function while satisfying some conditions. In this case, PLogP was optimized with the condition of using the Tanimoto coefficient based on ECFP4 fingerprint. Models perform $4\times50=200$ steps of optimization for each of the 800 molecules with the lowest PLogP in the ZINC dataset. Mol-CycleGAN[34] and GCPN[27] ware used for comparison.

Results and Discussion

Normal Optimization

We evaluate the performance of the optimization task using the best molecular property score and the distribution of generated molecules using validity, uniqueness and novelty. Validity rate is defined as the ratio of SMILES that can be parsed by RDKit to all generated molecules. Uniqueness is the ratio of duplicate molecules to valid molecules, and Novelty is the ratio of molecules that are not included in the training dataset to those included.

Optimization results are shown in Table 1. Our model shows sufficient results in both the QED and PLogP optimization tasks. In particular, the "Multi" model produces molecules with better scores than the "Signle" model. This result is also shown in Figure 5. The distribution of the QED/PLogP score of molecules at the starting point is shifts towards a higher score. This can be confirmed from Figure 6(b), which shows that the "Multi" model generates molecules with higher scores as the number of steps increases. The distribution of similarity between the generated molecules and the starting point molecules in Figure 6(a) shows that the "Multi" model also generates molecules in regions where the "Single" model does not generate. Both the "Single" and "Multi" models generate molecules with relatively high similarity Additionally, the "Multi" model seeks higher-scoring molecules and expands the chemical space of generated molecules to a lower similarity region. The reason for this is that the "Multi" model can generate a molecule with changes occurring at multiple positions in SMILES as shown in Figure 4(c) because the starting point is updated, however, the "Single" model cannot generate such a molecule because it inserts partial SMILES at only one position.

The results for the group of all generated molecules are shown in Table 2. Although validity is low, both uniqueness and novelty are high. The specific structures of the generated molecules are shown in Figure 7. For each stating point molecule, molecules with high scores and different structures are generated.

It is observed that the properties of molecules generated by the "Multi" model improve with each step. The number of steps for each iteration is 10,000, and the number of iterations is 5. We obtained good results from this experiment; however, these parameters are not optimum. Therefore, we need to investigate the relation between molecules and the hyperparameters of the "Multi" model, such as the number of steps and selection policy of the next starting point, in order to generate better molecules.

Constrained Optimization

The results of constrained optimization are shown in Table 3. The left column shows the difference in PLogP between the original molecule and the generated molecule with the highest PLogP as Improvement. Our method outperforms others in terms of the properties of molecules. However, success rate, which is the percentage of molecules with similarity above a threshold and improved PLogP, is worse than GCPN. Note that in GCPN, the policy is updated sequentially, while our method has a fixed policy. Thus, our method shows consistent results even when the number of evaluation steps is less (i.e., when optimizing a property that takes a long time to evaluate).

Conclusion

In this paper, we developed a generative model based on MCTS and RNN to generate derivative molecules starting from a specific molecule. This model generates molecules by generating partial SMILES using MCTS and RNN and replacing it with part of the starting point's SMILES. In addition, we propose "Single" and "Multi" models. Unlike the "Single" model, the "Multi" model replaces the starting point molecule with one of the generated molecules at a certain number of steps. As a result, it was demonstrated that the "Multi" model is superior to the "Single" model in terms of optimizing the QED score. Additionally, molecules generated by our model have high uniqueness and novelty, and the chemical space

consisting of generated molecules is large in terms of similarity and molecular weight.

We conducted an experiment for 200 molecules from the ZINC database and obtained good results; however, we do not assumed that this is enough to ensure the reliable performance of our model and We have to further investigate the relation between the starting point molecules and generated molecules.

Acknowledgement

This work was partially supported by the Platform Project for Supporting Drug Discovery and Life Science Research (Basis for Supporting Innovative Drug Discovery and Life Science Research (BINDS)) from AMED under Grant Number JP20am0101112 and the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Numbers 20H00620 (To M.S.).

AUTHOR INFORMATION

Corresponding Author

*E-mail: sekijima@c.titech.ac.jp

Notes

The authors declare no competing interests.

Supporting Information Available

The following files are available.

- Supporting_information.docx: ROC curves and EF 1%, EF 10%, and AUC values for the 102 target proteins of the DUD-E dataset as well as a full description of the DUD-E dataset.

References

(1) PhRMA, BIOPHARMACEUTICALS IN PERSPECTIVE SUMMER 2019. 2019; https://www.phrma.org/-/media/Project/PhRMA/PhRMA-Org/PhRMA-Org/PDF/P-R/PhRMA_2019_ChartPack_Final.pdf (visited: 2021-3-22).

(2) Mullard, A. New drugs cost US $2.6 billion to develop. *Nature Reviews Drug Discovery* **2014**, *13*, 877.

(3) Varma, H.; Lo, D.; Stockwell, B. *Neurobiology of Huntington's Disease*; CRC Press, 2010; pp 121–145.

(4) Schneider, G. Virtual screening: an endless staircase? *Nature Reviews Drug Discovery* **2010**, *9*, 273–276.

(5) Chiba, S. et al. Identification of potential inhibitors based on compound proposal contest: Tyrosine-protein kinase Yes as a target. *Scientific reports* **2015**, *5*, 17209.

(6) Chiba, S. et al. An iterative compound screening contest method for identifying target protein inhibitors using the tyrosine-protein kinase Yes. *Scientific Reports* **2017**, *7*, 12038.

(7) Chiba, S. et al. A prospective compound screening contest identified broader inhibitors for Sirtuin 1. *Scientific Reports* **2019**, *9*.

(8) Rao, V.; Srinivas, K. Modern drug discovery process: An in silico approach. **2011**, *2*.

(9) Li, H.; Leung, K.-S.; Wong, M.-H.; Ballester, P. J. Improving AutoDock Vina Using Random Forest: The Growing Accuracy of Binding Affinity Prediction by the Effective Exploitation of Larger Data Sets. *Molecular Informatics* **2015**, *34*, 115–126.

(10) Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D. R. Protein–Ligand Scoring with Convolutional Neural Networks. *Journal of Chemical Information and Modeling* **2017**, *57*, 942–957.

(11) Yasuo, N.; Sekijima, M. Improved Method of Structure-Based Virtual Screening via Interaction-Energy-Based Learning. *Journal of Chemical Information and Modeling* **2019**, *59*, 1050–1061.

(12) Yasuo, N.; Nakashima, Y.; Sekijima, M. CoDe-DTI: Collaborative Deep Learning-based Drug-Target Interaction Predictior. 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). 2018; pp 792–797.

(13) Elton, D. C.; Boukouvalas, Z.; Fuge, M. D.; Chung, P. W. Deep learning for molecular design—a review of the state of the art. *Molecular Systems Design & Engineering* **2019**, *4*, 828–849.

(14) Elton, D.; Boukouvalas, Z.; Fuge, M.; Chung, P. Deep learning for molecular design - a review of the state of the art. *Molecular Systems Design & Engineering* **2019**, *4*.

(15) Sanchez-Lengeling, B.; Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* **2018**, *361*, 360.

(16) 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.

(17) 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.

(18) Yang, X.; Zhang, J.; Yoshizoe, K.; Terayama, K.; Tsuda, K. ChemTS: an efficient python library for de novo molecular generation. *Science and Technology of Advanced Materials* **2017**, *18*, 972–976.

(19) Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. Cambridge, MA, USA, 2014; p 2672–2680.

(20) Kingma, D. P.; Welling, M. Auto-Encoding Variational Bayes. 2013; `http://arxiv.org/abs/1312.6114`, cite arxiv:1312.6114.

(21) Jin, W.; Barzilay, R.; Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. Stockholmsmässan, Stockholm Sweden, 2018; pp 2323–2332.

(22) Zhou, Z.; Kearnes, S.; Li, L.; Zare, R. N.; Riley, P. Optimization of Molecules via Deep Reinforcement Learning. *Scientific Reports* **2019**, *9*, 10752.

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

(24) Simonovsky, M.; Komodakis, N. *GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I*; 2018; pp 412–422.

(25) De Cao, N.; Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models* **2018**,

(26) Jin, W.; Barzilay, R.; Jaakkola, T. Hierarchical Generation of Molecular Graphs using Structural Motifs. **2020**,

(27) You, J.; Liu, B.; Ying, R.; 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. Red Hook, NY, USA, 2018; p 6412–6422.

(28) Coulom, R. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. *Proceedings of the 5th international conference on Computers and games* **2006**, 72–83.

(29) 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.

(30) Kocsis, L.; Szepesvári, C. Bandit Based Monte-Carlo Planning. Machine Learning: ECML 2006. pp 282–293.

(31) Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780.

(32) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. 2017.

(33) Bickerton, R.; Paolini, G.; Besnard, J.; Muresan, S.; Hopkins, A. Quantifying the chemical beauty of drugs. *Nature chemistry* **2012**, *4*, 90–8.

(34) Maziarka, L.; Pocha, A.; Kaczmarczyk, J.; Rataj, K.; Danel, T.; Warchol, M. Mol-CycleGAN: a generative model for molecular optimization. *Journal of Cheminformatics* **2020**, *12*, 2.

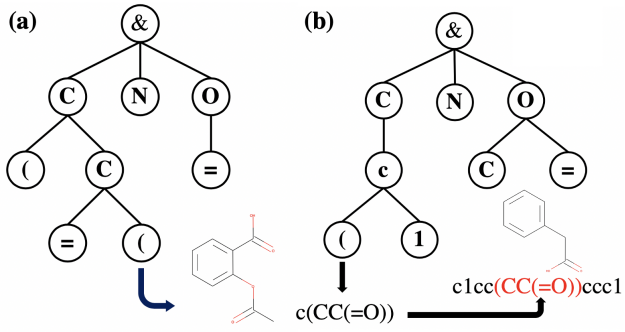


Figure 1: (a) Existing MCTS-based molecular generative model such as ChemTS.[18] This model generates full SMILES strings through MCTS. (b) Our model starting from a specific molecule. It generates partial SMILES and replaces a part of the starting point SMILES with the generated one.

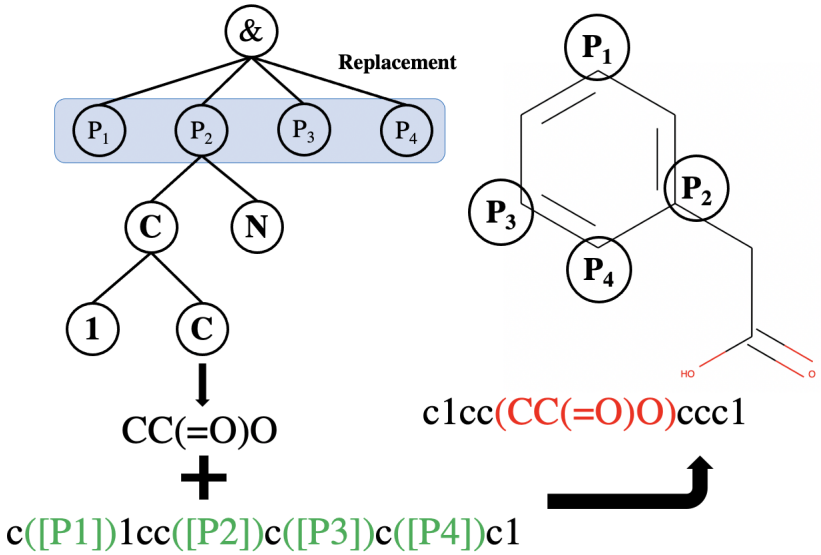


Figure 2: Selecting starting point SMILES that is replaced with the generated one. Nodes under the root node have the information of the part deleted from starting point SMILES. The generation of partial SMILES begins from the grandchild nodes of the root node.

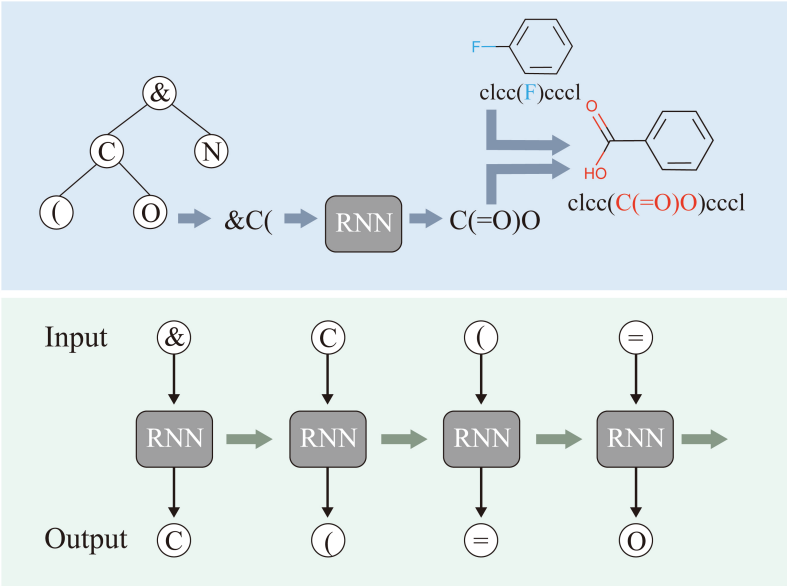
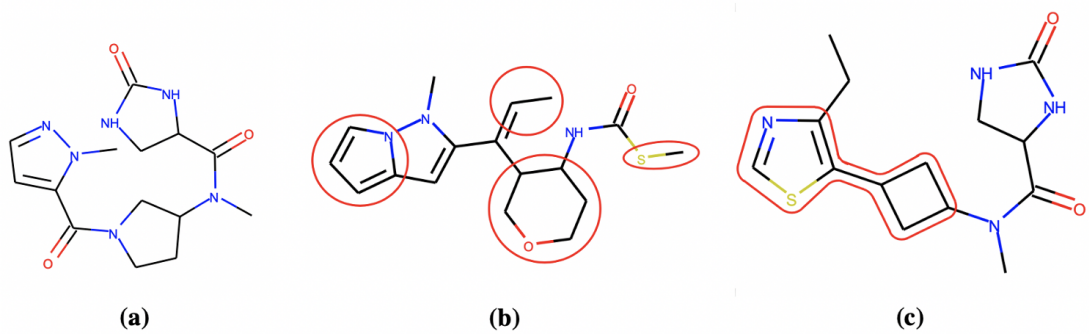


Figure 3: The whole flow of our method.



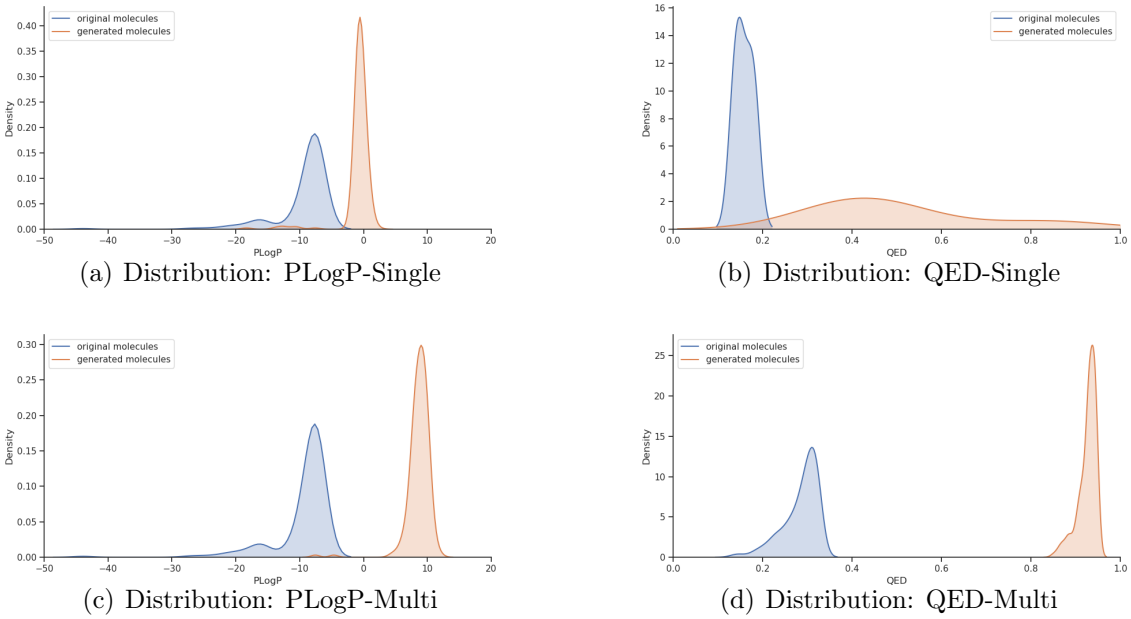
(a) (b) (c)

(a) CN(C(=O)C1CNC(=O)N1)C1CCN(C(=O)c2ccnn2C)C1

(b) CSC(=O)NC1CCOCC(C(=C(/C))c2cc3cccn3n2C)1

(c) CN(C(=O)C1CNC(=O)N1)C1CC(c2scnc2C)C1

Figure 4: Molecules generated by two models, showing the difference in terms of structure and SMILES. (a) Starting point molecule. (b) Molecules generated by the "Multi" model and (c) molecules generated by the "Single" model. The red outlines in the molecular structures and the red character string in SMILES are changes from molecule (a). Note that deleted parts from the starting point molecule during molecule generation are not highlighted in this figure.



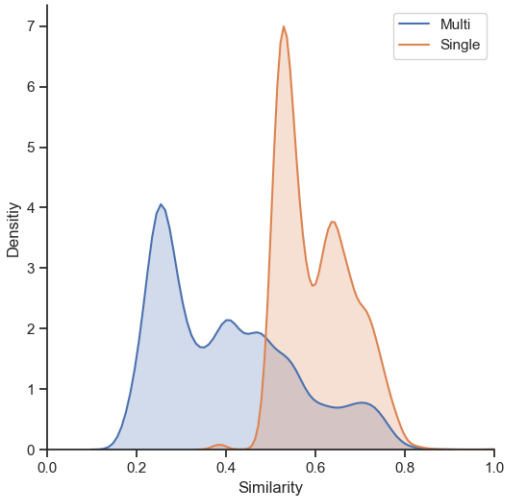
(a) Distribution: PLogP-Single

(b) Distribution: QED-Single

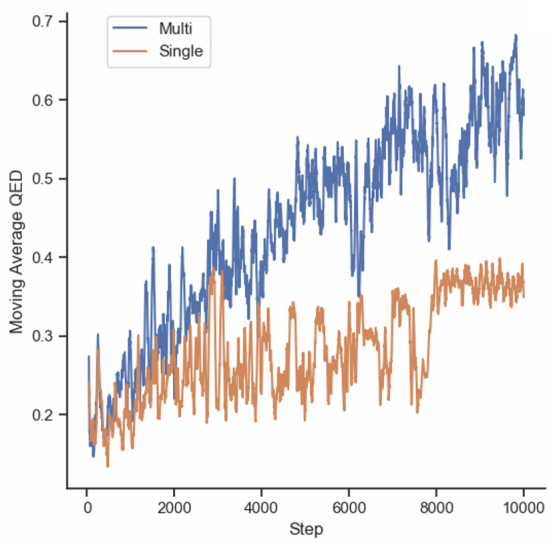
(c) Distribution: PLogP-Multi

(d) Distribution: QED-Multi

Figure 5: The distribution of the initial and generated molecules for each model and metrics. Blue: original molecules, red: generated molecules.



(a) Similarity distribution of both model



(b) Moving Average of QED for both model

Figure 6: The optimization result from one randomly chosen compound. (a)The distribution of similarity to the starting point molecule of generated molecules by our models. Blue: Multi model, red: Single model. (b) Moving average of QED with 20 steps. Blue: Multi model, red: Single model.

Table 1: The results of the optimization tasks. The mean of the top 3 highest scored Penalized logP and QED scores of the validation set and the validity rate are described.

Method	Penalized LogP				QED			
	1st	2nd	3rd	Validity	1st	2nd	3rd	Validity
ZINC	-9.41	-	-	-	0.285	-	-	-
Single	-2.20	-2.39	-2.53	29.8%	0.681	0.671	0.666	38.5%
Multi	11.33	11.20	11.10	31.8%	0.920	0.915	0.912	77.0 %

Table 2: Mean property of all generated molecules for 200 validation compounds. Validity, uniqueness and novelty of all generated molecules are shown for four models. Validity is the ratio of valid SMILES to all generated SMILES. Uniqueness is the ratio of non-duplicate molecules to all valid molecules. Novelty is the ratio of molecules that are not included in the training dataset to all valid molecules.

	Validity	Uniqueness	Novelty
PLogP-Single	0.298	0.981	1.0
PLogP-Multi	0.318	0.991	1.0
QED-Single	0.626	0.945	0.999
QED-Multi	0.770	0.958	0.999

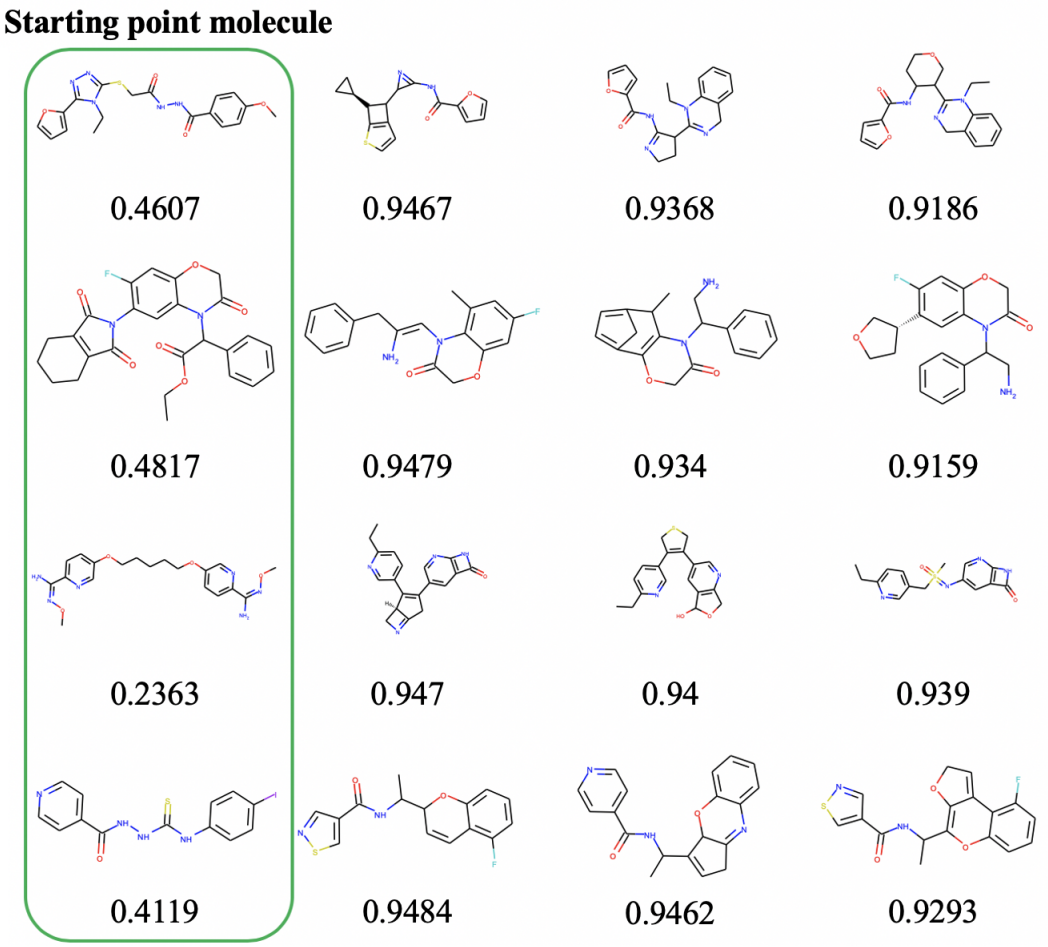


Figure 7: Structures of target molecules and generated molecules. In each row, molecule in the first column is the target (starting point) molecule, and generated molecules are shown in row. These molecules are generated by the "Multi" model that replaces starting point molecules with generated molecules, with changing replace point at a certain number of steps. In this experiment, 2000 steps is used. The QED score for each molecule is shown under the molecule.

Table 3: Results of constrained optimization for 800 validation molecules. The results of GCPN and Mol-CycleGAN are cited from L. Maziarka et al. (2020).[34] The mean and standard deviation of improvement, similarity, and success rate of generated molecules are shown.

δ	GCPN			Mol-CycleGAN			Ours		
	Improvement	Similarity	Success	Improvement	Similarity	Success	Improvement	Similarity	Success
0.2	4.12 ± 1.19	0.34 ± 0.11	100%	5.79 ± 2.35	0.30 ± 0.11	93.8%	9.94 ± 2.74	0.23 ± 0.04	100.0%
0.4	2.49 ± 1.30	0.47 ± 0.08	100%	2.89 ± 2.08	0.52 ± 0.10	58.8%	6.04 ± 2.29	0.42 ± 0.02	100.0%
0.6	0.79 ± 0.63	0.68 ± 0.08	100%	1.22 ± 1.48	0.69 ± 0.07	19.3%	1.99 ± 1.74	0.62 ± 0.02	85.3%

Algorithm 1 Molecule optimization algorithm

procedure OPTIMIZE($mol_s, n_{step}, n_{iter}$)
    set mol_s as initial molecule m_0
    for $i = 1, 2, ..., n_{iter}$ **do**
        $products$ = MCTS(m_0, n_{step})
        set $\arg\max_{m \in products} reward(m)$ as initial molecule m_0

procedure MCTS(m_0, n_{step})
    create root node with m_0
    create child nodes with possible replacement partial SMILES
    for $j = 1, 2, ..., n_{step}$ **do**
        $leaf\ node$ = Select()
        $new\ node$ = Expand($leaf\ node$)
        $reward$ = Rollout($new\ node$)
        Backup($leaf\ node, reward$)

procedure SELECT
    $node = rootnode$
    while $node$ is not leaf **do**
        node = $\arg\max_{v \in child(node)} UCB(v)$
    return $node$

procedure EXPAND($node$)
    $new\ node$ = RNN($node$)
    return $new\ node$

procedure ROLLOUT($node$)
    while $node$ is not terminal **do**
        $node$ = RNN($node$)
    return Reward($node$)

procedure BACKUP($node, reward$)
    while $node$ is not root **do**
        $node.visit += 1$
        $node.value += reward$
        $node = parent(node)$