Science and Technology of Advanced Materials

ISSN: 1468-6996 (Print) 1878-5514 (Online) Journal homepage: https://www.tandfonline.com/loi/tsta20

ChemTS: an efficient python library for *de novo* molecular generation

Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama & Koji Tsuda

To cite this article: Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama & Koji Tsuda (2017) ChemTS: an efficient python library for *de novo* molecular generation, Science and Technology of Advanced Materials, 18:1, 972-976, DOI: 10.1080/14686996.2017.1401424

To link to this article: https://doi.org/10.1080/14686996.2017.1401424

© 2017 The Author(s). Published by National Institute for Materials Science in partnership with Taylor & Francis

Published online: 24 Nov 2017.

Submit your article to this journal

Article views: 4443

View related articles

View Crossmark data

Citing articles: 9 View citing articles

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=tsta20

SCIENCE AND TECHNOLOGY OF ADVANCED MATERIALS, 2017
VOL. 18, NO. 1, 972–976
https://doi.org/10.1080/14686996.2017.1401424

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

ChemTS: an efficient python library for *de novo* molecular generation

Xiufeng Yang[a], Jinzhe Zhang[b], Kazuki Yoshizoe[c], Kei Terayama[a] and Koji Tsuda[a,c,d]

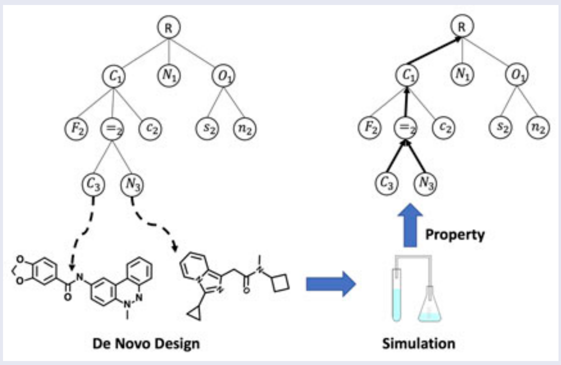
[a]Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Japan;
[b]Department of Biosciences, INSA Lyon, Villeurbanne Cedex, France;
[c]RIKEN, Center for Advanced Intelligence Project, Tokyo, Japan;
[d]National Institute for Materials Science, Tsukuba, Japan

ABSTRACT
Automatic design of organic materials requires black-box optimization in a vast chemical space. In conventional molecular design algorithms, a molecule is built as a combination of predetermined fragments. Recently, deep neural network models such as variational autoencoders and recurrent neural networks (RNNs) are shown to be effective in *de novo* design of molecules without any predetermined fragments. This paper presents a novel Python library ChemTS that explores the chemical space by combining Monte Carlo tree search and an RNN. In a benchmarking problem of optimizing the octanol-water partition coefficient and synthesizability, our algorithm showed superior efficiency in finding high-scoring molecules. ChemTS is available at https://github.com/tsudalab/ChemTS.

ARTICLE HISTORY
Received 29 September 2017
Revised 1 November 2017
Accepted 2 November 2017

KEYWORDS
Molecular design; Monte Carlo tree search; recurrent neural network; python library

CLASSIFICATION
60 New topics/Others; 404 Materials informatics / Genomics



1. Introduction

In modern society, a variety of organic molecules are used as important materials such as solar cells [1], organic light-emitting diodes [2], conductors [3], sensors [4] and ferroelectrics [5]. At the highest level of abstraction, design of organic molecules is formulated as a combinatorial optimization problem to find the best solutions in a vast chemical space. Most computer-aided methods for molecular design build a molecule by a combination of predefined fragments (e.g. [6]). Recently, Ikebata et al. [7] succeeded *de novo* molecular design using an engineered language model of SMILES representation of molecules [8]. It is increasingly evident, however, that engineered models often perform worse than neural networks in text and image generation [9,10]. Gomez-Bombarelli et al. [11] were the first to employ a neural network called variational autoencoder (VAE) to generate molecules. Later Kusner et al. enhanced it to grammar variational autoencoder (GVAE) [12]. SMILES strings created by VAEs are mostly invalid (i.e. they do not translate to chemical structures); so, generation steps have to be repeated many times to obtain a molecule. Segler et al. [13] showed that a recurrent neural network (RNN) using long short-term memory (LSTM) [14] achieves a high probability of valid SMILES generation. In their algorithm, a large number of candidates are generated randomly and a black-box optimization algorithm is employed to choose high-scoring molecules. It is required to generate a very large number of candidates to ensure that desirable molecules are included in the candidate set. Optimization in a too large candidate space can be inhibitively slow.

In this paper, we present a novel Python library ChemTS to offer material scientists a versatile tool of *de novo* molecular design. The space of SMILES strings is represented as a search tree where the *i*th level corresponds to the *i*th symbol. A path from the root to

CONTACT Koji Tsuda tsuda@k.u-tokyo.ac.jp

© 2017 The Author(s). Published by National Institute for Materials Science in partnership with Taylor & Francis.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

a terminal node corresponds to a complete SMILES string. Initially, only the root node exists and the search tree is gradually generated by Monte Carlo tree search (MCTS) [15]. MCTS is a randomized best-first search method that showed exceptional performance in computer Go [16]. Recently, it has been successfully applied to alloy design [17]. MCTS constructs only a shallow tree and downstream paths are generated by a rollout procedure. In ChemTS, an RNN trained by a large database of SMILES strings is used as the rollout procedure. In a benchmarking experiment, ChemTS showed better efficiency in comparison to VAEs, creating about 40 molecules per minute. As a result, high-scoring molecules were generated within several hours.

2. Method

ChemTS requires a database of SMILES strings and a reward function $r(S)$ where $S = \{s_1, \ldots, s_T\}$ is an input SMILES string. Our definition of SMILES strings contains the following symbols representing atoms, bonds, ring numbers and branches: $s_t \in$ {C, c, o, O, N, F, [C@@H], n, -, S,Cl, [O-],[C@H], [NH+],[C@], s, Br, [nH], [NH3+], [NH2+], [C@@], [N+], [nH+], [S@], [N-], [n+],[S@@], [S-], I, [n-], P, [OH+],[NH-], [P@@H], [P@@], [PH2], [P@], [P+], [S+],[o+], [CH2-], [CH-], [SH+], [O+], [s+], [PH+], [PH], [S@@+], /,=, #, 1,2,3,4,5,6,7,8,9,(,),}. In addition, we have a terminal symbol \$. The reward function involves first principle or semi-empirical calculations and describes the quality of the molecule described by S. If S does not correspond to a valid molecule, $r(S)$ is set to an exceptionally small value. We employ *rdkit* (www.rdkit.org) to check if S is valid or not. Before starting the search, an RNN is trained by the database and we obtain the conditional probability $P(s_t|s_1, \ldots, s_{t-1})$ as a result. The architecture of our RNN is similar to that in [13] and will be detailed in Section 2.1.

MCTS creates a search tree, where each node corresponds to one symbol. Nodes with the terminal symbol are called terminal nodes. Starting with the root node, the search tree grows gradually by repeating the four steps, selection, expansion, simulation and backpropagation (Figure 1). Each intermediate node has an upper confidence bound (UCB) score that evaluates the merit of the node [15]. The distinct feature of MCTS is the use of *rollout* in the simulation step. Whenever a new node is added, paths from the node to terminal nodes are built by a random process. In computer games, it is known that uniformly random rollout does not perform well, and designing a better rollout procedure based on available knowledge is essential in achieving high performance [15]. Our idea is to employ a trained RNN for rollout. A node at level $t-1$ has a partial SMILES string $s_1, \ldots, s_{t-1}$ corresponding to the path from the root to the node. Given the partial string, RNN allows us to compute the distribution of the next letter s_t. Sampling from the distribution, the string is elongated by one. Elongation by RNN is repeated until the terminal symbol occurs. After elongation is done, the reward of the generated string is computed. In the backpropagation step, the reward is propagated backwards and the UCB scores of traversed nodes are updated. See [17] for details about MCTS.

2.1. Recurrent neural network

Our RNN has a non-deterministic output: an input string $s_1, \ldots, s_T$ is mapped to probability distributions of output symbols $P(y_1), \ldots, P(y_T)$. The RNN represents the function $\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$, where $\boldsymbol{h}_t \in \Re^{512}$ is a hidden state at position t and $\boldsymbol{x}_t \in \Re^{64}$ is the one-hot coded vector of input symbol s_t. The function f is implemented by two stacked gated recurrent units (GRUs) [14], each with 256 dimensional hidden states. The input vector $\boldsymbol{x}_t$ is fed to the lower GRU, and the hidden state of the lower GRU is fed to the upper GRU. The distribution of output symbol is computed as $P(y_t = j) = g_j(\boldsymbol{h}_t)$, where g_j is a softmax activation function depending only on the hidden state of the upper GRU.

Given N strings in the training set, we train the network such that it outputs a right-shifted version of the input. Denoted by $\boldsymbol{x}_{it}$, the one-hot coded vector of the tth symbol in the ith training string. The parameters in the network $\boldsymbol{\theta}$ are trained to minimize the following loss function,

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sum_{t=1}^{T-1} D(\boldsymbol{x}_{it+1}, P(y_t)),$$

where D denotes the relative entropy. Our RNN was implemented using Keras library (github.com/fchollet/keras), and trained with ADAM [18] using a batch size of 256. After the training is finished, one can compute $P(y_t)$ from $s_1, \ldots, s_{t-1}$. It allows us to perform rollout by sampling the next symbol repeatedly.

3. Experiments

Following [11], we generate molecules that jointly optimize the octanol-water partition coefficient logP and the other two properties: synthetic accessibility [19] and ring penalty that penalizes unrealistically large rings. The score of molecule S is described as

$$J(S) = logP(S) - SA(S) - RingPenalty(S). \quad (1)$$

The reward function of ChemTS is defined as

$$r(S) = \begin{cases} \frac{J(S)}{1+|J(S)|} & \text{Valid SMILES} \\ -1.0 & \text{otherwise.} \end{cases} \quad (2)$$

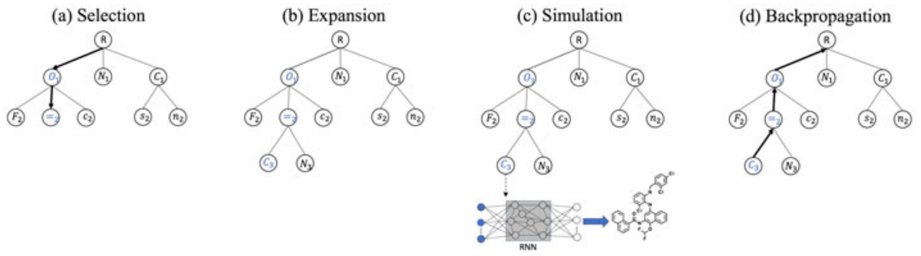


Figure 1. Monte Carlo tree search. (a) Selection step: the search tree is traversed from the root to a leaf by choosing the child with the largest UCB score. (b) Expansion step: 30 children nodes are created by sampling from RNN. (c) Simulation step: paths to terminal nodes are created by the rollout procedure using RNN. Rewards of the corresponding molecules are computed. (d) Backpropagation step: the internal parameters of upstream nodes are updated.

SMILES representation	*J*
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c2ccccc2c1OC(F)F)c1cccc2ccccc12	6.56
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)ccc1C1=CCCCC1)c1cc(F)cc(Cl)c1	6.43
O=C(Nc1cc(Nc2c(Cl)cccc2N=C(SC2CCCCC2)c2ccccc2)cc(Cl)c1Cl)c1ccc2ccccc2n1	6.34
O=C(Nc1cc(Oc2ccc(Cl)cc2Cl)ccc1Nc1cc(Cl)ccc1Cl)c1ccc(Cl)cc1	6.33
O=C(Nc1cc(Nc2c(Cl)cccc2Cl)c(Cl)cc1Br)N(c1ccccc1)c1ccc(Cl)cc1	6.26
O=C(Nc1cc(Oc2c(Cl)cccc2Oc2ccc(-c3ccccc3)cc2)ccc1Cl)c1ccccc1	6.19
O=C(Nc1cc(Nc2c(Cl)cccc2Cl)c(Cl)c(C(=O)N(Cc2ccccc2)c2ccccc2)c1Cl)c1ccccc1F	6.08
O=C(Nc1cc(Oc2ccc(Cl)cc2Cl)cc(Cl)c1Cl)c1ncoc1-c1ccc(Sc2ccccc2)cc1	6.007
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c2ncccc2c1Cl)c1ccc(Cl)cc1	6.0067
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2)c(Cl)cc1Cl)c1cc(F)ccc1Cl	6.0062
O=C(Nc1cc(Oc2c(Cl)cccc2Oc2ccccc2)nnc1-c1ccccc1)c1sc2ccccc2c1Cc1ccccc1	6.004
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c2ncccc2c1Cl)c1ccccc1Cl	5.97
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c(Cl)cc1Cl)c1ccc(F)cc1F	5.958
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccccc2)ccc1C(F)(F)F)c1ccc(Cl)c2ccccc12	5.952
O=C(Nc1cc(Nc2c(Cl)cccc2Cl)c(Cl)cc1OC(F)F)N(Cc1ccccc1)c1ccccc1C(F)(F)F	5.94
O=C(Nc1cc(Oc2c(Cl)cccc2Oc2ccccc2C2=CCCCC2)cc(Cl)c1)c1ccccc1	5.93
O=C(Nc1cc(Nc2c(Cl)cccc2[N+](=O)[O-])cs1)c1sc2ccc(Br)cc2c1N(c1ccccc1)c1ccccc1	5.92
O=C(Nc1cc(Nc2c(Cl)cccc2Cl)c(C(=O)c2ccc(Cl)cc2F)c(Cl)c1)Nc1cccc(Cl)c1	5.87
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)cc(F)c1F)c1cccc2ccccc12	5.84
O=C(Nc1cc(Nc2c(Cl)cccc2NCc2ccc(Cl)cc2Cl)c(Cl)cc1Cl)c1cccs1	5.82

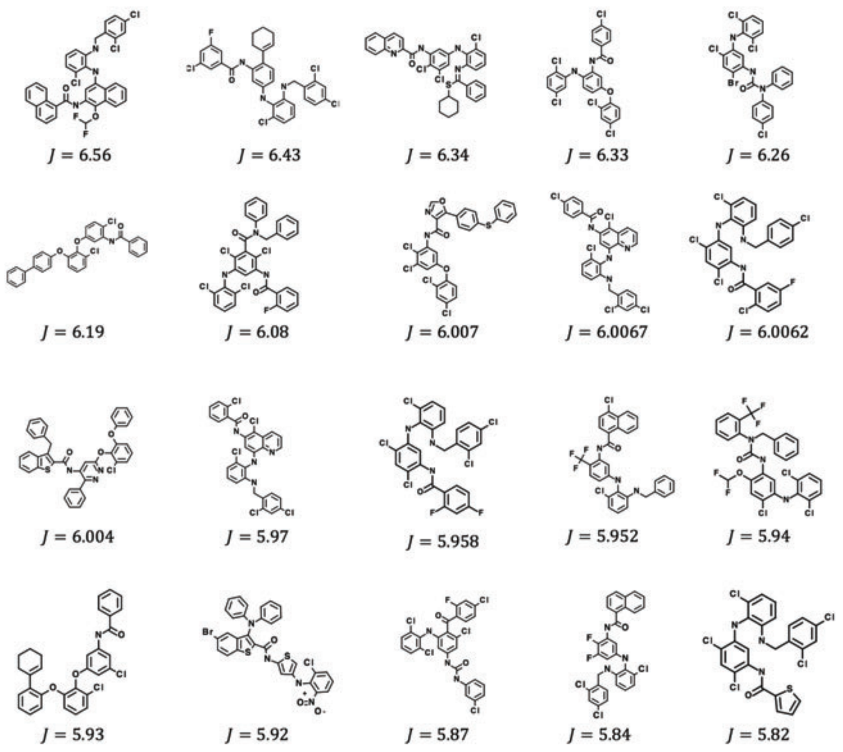


Figure 2. Best 20 molecules by ChemTS. Blue parts in SMILES strings indicate prefixes made in the search tree. The remaining parts are made by the rollout procedure.

Table 1. Maximum score *J* at time points 2,4,6 and 8 h achieved by different molecular generation methods.

Method	2 h	4 h	6 h	8 h	Molecules/Min
ChemTS	4.9 ± 0.4	5.4 ± 0.5	5.5 ± 0.4	5.6 ± 0.5	41 ± 1.6
RNN+BO	3.5 ± 0.3	4.5 ± 0.2	4.5 ± 0.2	4.5 ± 0.2	8.3 ± 0.0
Only RNN	4.5 ± 0.3	4.6 ± 0.3	4.8 ± 0.3	4.8 ± 0.3	41 ± 1.4
CVAE+BO	-30 ± 27	-1.4 ± 2.2	-0.6 ± 1.1	-0.0 ± 0.9	0.1 ± 0.1
GVAE+BO	-4.3 ± 3.1	-1.3 ± 1.7	-0.2 ± 1.0	0.3 ± 1.3	1.4 ± 0.9

Notes: The rightmost column shows the number of generated molecules per minute. The average values and standard deviations over 10 trials are shown.

ChemTS was compared with two existing methods CVAE [11] and GVAE [12] based on variational autoencoders. Their implementation is available at https://github.com/mkusner/grammarVAE. Both methods perform molecular generation by Bayesian optimization (BO) in a latent space of VAE. RNN, CVAE and GVAE were trained with approximately 250,000 molecules in ZINC database [20]. All methods were trained for 100 epochs. Training took 3.8, 9.4 and 33.5 h, respectively, on a CentOS 6.7 server with a GeForce GTX Titan X GPU. To evaluate the efficiency of MCTS, we prepared two alternative methods using RNN. One is simple random sampling using RNN, where the first symbol is made randomly and it is elongated until the terminal symbol occurs. The other is the combination of RNN and Bayesian optimization [21], where 4000 molecules are made a priori and Bayesian optimization is applied to find the best scoring molecule.

As shown in Table 1, effectiveness of each method is quantified by the maximum score *J* among all generated molecules at 2, 4, 6 and 8 h and the speed of molecules generation (i.e. the number of generated molecules per minute). VAE methods performed substantially slower than RNN-based methods, which reflects the low probability of generating valid SMILES strings. ChemTS performed best in finding high-scoring molecules, while the speed of molecular generation (40.89 molecules per minute) was only slightly worse than random generation by RNN (41.33 molecules per minute). The combination of RNN and BO could not find high-scoring molecules. Preparing more candidate molecules may improve the best score, but it would further slow down the molecular generation. In general, it is difficult to design a correct reward function when there are multiple objectives. So, it is important to generate many good molecules in a given time frame to allow the user to browse and select favourite molecules afterwards. See Figure 2 for the best molecules generated by ChemTS.

4. Conclusion

In this paper, we presented a new Python package for molecular generation. It will be further extended to include more sophisticated tree search methods and neural networks. Use of additional packages for computational physics such as *pymatgen* [22] allows the users to easily implement their own reward function. We look forward to see ChemTS as a part of the open-source ecosystem for organic materials development.

Acknowledgements

We would like to thank Hou Zhufeng, Diptesh Das, Masato Sumita and Thaer M. Dieb for their fruitful discussions.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the 'Materials research by Information Integration' Initiative (MI2I) project and Core Research for Evolutional Science and Technology (CREST) [grant number JPMJCR1502] from Japan Science and Technology Agency (JST). It was also supported by Grant-in-Aid for Scientific Research on Innovative Areas 'Nano Informatics' [grant number 25106005] from the Japan Society for the Promotion of Science (JSPS). In addition, it was supported by Ministry of Education, Culture, Sports, Science and Technology (MEXT) as 'Priority Issue on Post-K computer' (Building Innovative Drug Discovery Infrastructure Through Functional Control of Biomolecular Systems).

References

[1] Niu G, Guo X, Wang L. Review of recent progress in chemical stability of perovskite solar cells. J Mater Chem A. 2015;3(17):8970–8980.
[2] Kaji H, Suzuki H, Fukushima T, et al. Purely organic electroluminescent material realizing 100% conversion from electricity to light. Nat Commun. 2015;6:8476.
[3] Ueda A, Yamada S, Isono T, et al. Hydrogen-bond-dynamics-based switching of conductivity and magnetism: a phase transition caused by deuterium and electron transfer in a hydrogen-bonded purely organic conductor crystal. J Am Chem Soc. 2014;136(34):12184–12192.
[4] Yeung MCL, Yam VWW. Luminescent cation sensors: from host-guest chemistry, supramolecular chemistry to reaction-based mechanisms. Chem Soc Rev. 2015;44(13):4192–4202.
[5] Horiuchi S, Tokura Y. Organic ferroelectrics. Nat Mater. 2008;7(5):357–366.
[6] Podlewska S, Czarnecki WM, Kafel R, et al. Creating the new from the old: Combinatorial libraries generation with machine-learning-based compound structure optimization. J Chem Inf Model. 2017;57(2):133–147.
[7] Ikebata H, Hongo K, Isomura T, et al. Bayesian molecular design with a chemical language model. J Comput Aided Mol Des. 2017;31(4):379–391.

[8] Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. J Chem Inf Comput Sci. 1988;28(1):31–36.

[9] Bowman SR, Vilnis L, Vinyals O, et al. Generating sentences from a continuous space. In: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL); Berlin; 2016. p. 10–21.

[10] Oord Avd, Kalchbrenner N, Kavukcuoglu K. Pixel recurrent neural networks. In: Proceedings of 33rd International Conference on Machine Learning (ICML); New York City; 2016. p. 1747–1756.

[11] Gómez-Bombarelli R, Duvenaud D, Hernández-Lobato JM, et al. Automatic chemical design using a data-driven continuous representation of molecules. arXiv preprint arXiv:161002415; 2016.

[12] Kusner MJ, Paige B, Hernández-Lobato JM. Grammar variational autoencoder. In: Proceedings of 34th International Conference on Machine Learning (ICML); Sydney; 2017. p. 1945–1954.

[13] Segler MH, Kogej T, Tyrchan C, et al. Generating focussed molecule libraries for drug discovery with recurrent neural networks. arXiv preprint arXiv:170101329; 2017.

[14] Cho K, van Merrienboer B, Gülçehre Ç, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); Doha; 2014. p. 1724–1734.

[15] Browne C, Powley E, Whitehouse D, et al. A survey of Monte Carlo tree search methods. IEEE Trans Comput Intell AI Games. 2012;4(1):1–43.

[16] Silver D, Huang A, Maddison CJ, et al. Mastering the game of go with deep neural networks and tree search. Nature. 2016;529(7587):484–489.

[17] Dieb TM, Ju S, Yoshizoe K, et al. MDTS: automatic complex materials design using Monte Carlo tree search. Sci Tech Adv Mater. 2017;18(1):498–503.

[18] Kingma D, Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:14126980; 2014.

[19] Ertl P, Schuffenhauer A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. J Cheminf. 2009;1(1):8.

[20] Irwin JJ, Sterling T, Mysinger MM, et al. ZINC: a free tool to discover chemistry for biology. J Chem Inf Model. 2012;52(7):1757–1768.

[21] Ueno T, Rhone T, Hou Z, et al. COMBO: an efficient Bayesian optimization library for materials science. Mater Discov. 2016;4:18–21.

[22] Ong SP, Richards WD, Jain A, et al. Python materials genomics (pymatgen): a robust, open-source python library for materials analysis. Comp Mater Sci. 2013;68:314–319.