ChemTSv2: Democratizing Functional Molecular Design Using de novo Molecule Generator

Shoichi Ishida,*† Tanuj Aasawat,‡ Masato Sumita,*‡,§ Michio Katouda,∥⊥
Tatsuya Yoshizawa,† Kazuki Yoshizoe,# Koji Tsuda,@ and Kei Terayama*†

†Graduate School of Medical Life Science, Yokohama City University, 1-7-29, Suchiro-cho,
Tsurumi-ku, Yokohama 230-0045 Kanagawa, Japan.
‡Parallel Computing Lab - India, Intel Labs
¶RIKEN Center for Advanced Intelligence Project, Tokyo, 103-0027, Japan
§International Center for Materials Nanoarchitectonics(WPI-MANA), National Institute
for Materials Science, Tsukuba, 305-0044, Japan
∥Research Organization for Information Science and Technology, 1-18-16, Hamamatsucho
Minato-ku, Tokyo, 105-0013 Japan
⊥Waseda Research Institute for Science and Engineering, Waseda University, 3-4-1,
Ookubo, Sinjuku-ku, Tokyo 169-8555, Japan
#Research Institute for Information Technology, Kyushu University, 744 Motooka,
Nishi-ku, Fukuoka 819-0395, Japan.
@Graduate School of Frontier Sciences, The University of Tokyo, Chiba, Japan

E-mail: ishida.sho.nm@yokohama-cu.ac.jp terayama@yokohama-cu.ac.jp

Abstract

Designing functional molecules is the prerogative of experts who have advanced
knowledge and experience in their fields. To democratize automatic molecular design
for both experts and non-experts, we introduce a generic open-sourced framework,
ChemTSv2, to design molecules based on a de novo molecule generator equipped with an easy-to-use interface. Besides, ChemTSv2 can easily be integrated with various simulation packages, such as Gaussian 16 package, and supports a massively parallel exploration that accelerates molecular designs. We exhibit the potential of molecular design with ChemTSv2, including previous work, such as chromophores, fluorophores, drugs, and so forth. ChemTSv2 contributes to democratizing inverse molecule design in various disciplines relevant to chemistry.

Introduction

Controlling phenomena and properties of molecules at the atomic level is a challenging task. In chemical laboratories, exploring and optimizing chemical compounds are commonly performed for functionalizing and novelizing them by experts with professional knowledge. A vast amount of knowledge has been accumulated to facilitate the exploration and optimization processes in the long history of chemistry. Meanwhile, the vast amount of knowledge complicates and perplexes these processes, which causes numerous trial-and-error routines even for the experts. To alleviate real-world experiments, molecular and quantum mechanics simulation techniques have been developed as the segregate models of chemical experiments. These simulations have also been employed to find the expected chemical compounds within the possibility of the simulation prior to the real-world experiments. However, the diversity arising from the internal freedom of small molecules and the vast chemical space makes molecular designs by a single chemist puzzling. Moreover, this diversity leads to unintentionally inefficient processes (stagnating near local minima).

To deal with the diversity of chemical compounds and prevent the unintentional inefficiency, several methods for solving inverse problems have been proposed. Recently, various molecule generators, which use the inverse molecular designs based on the prediction models, have been developed and applied in several fields, such as materials and drug designs. However, the performance of these molecule generators depends on the prediction
models, which require a sufficient amount of known data about target properties. Meanwhile, molecule generators coupled with simulation techniques, such as quantum chemistry (QC), can overcome the known data dependency because these techniques basically require only molecule structure information. These molecule generators, however, require extreme computational costs, which can be reduced by introducing parallelized molecule generators to effectively explore chemical space.

Prior to actual experiments, the molecule generators are ultimately expected to automatically search the space to find optimal molecules with desired properties within a realistic time. We have previously shown the practical potential of a de novo molecule generator combined with QC simulations or machine learning (ML) models using our developed one, ChemTS, which is based on two algorithms: a recurrent neural network (RNN) as a molecule generator and a Monte Carlo tree search (MCTS) as an exploration system. Additionally, to explore the space depicted by QC, a parallelized version of ChemTS was introduced. Despite the high computational cost of QC, we have shown the molecular design potential of ChemTS in materials discovery through various experiments, such as light absorption/emission and electric charges retention. In addition, by switching the QC to a docking simulation for a protein or an ML prediction model, ChemTS can potentially be applied for drug discovery by designing potential inhibitors. Furthermore, we have demonstrated that ChemTS can be applied in both molecular design and molecular structure analyses. Although ChemTS has successfully designed experimentally validated molecules with various desired properties, ChemTS still has limitations for third-party users because the existing software packages are not user-friendly and are distributed independently according to research topics.

Here, we propose ChemTSv2, a Python-based framework that allows users to generate molecules through ChemTS with the ease of lunch. To achieve this goal, ChemTSv2 provides four main functions: (1) an easy-to-run interface that requires only a configuration file, (2) an easy-to-define framework of any reward function and molecular filter that can be
used during the molecular design process, (3) massive parallelization mode to incorporate a computationally expensive reward, and (4) various usage examples. As the massive parallelization mode, ChemTSv2 incorporates the most recent parallel MCTS algorithm\(^{38}\) of ChemTS, which increases molecular design speed from days to minutes. In this paper, we present the ChemTSv2 framework and review the ChemTS-based molecular design with some experimental validation, such as chromophores, fluorophores, electret polymers, and drugs. The ChemTSv2 package is publicly available on GitHub at https://github.com/molecule-generator-collection/ChemTSv2.

**Implementation**

**Usage workflow**

The usage workflow of ChemTSv2 consists of four steps: preparing reward and configuration files, executing ChemTSv2, and analyzing the generated molecules, as shown in Fig. 1. Detailed explanations are described in the following sections.

**Python environment preparation**

ChemTSv2 is written in Python3 and available on GitHub under the MIT Licence. ChemTSv2 mainly depends on the RDKit, TensorFlow, NumPy, and pandas packages. ChemTSv2 can be installed from the Python Package Index (PyPI) with the following command:

```
$ pip install chemtsv2
```

The command automatically installs the following dependencies: TensorFlow, RDKit, pandas, Joblib, and PyYmal, and the `chemtsv2` and `chemtsv2-mp` commands are available after the installation. Users who want to use the most recent parallel MCTS algorithm (available using the `chemtsv2-mp` command) should prepare a server supporting parallel computing and install the `mpi4py` package as follows: `pip install mpi4py==3.0.3`. When preparing the
Python environment, virtual environment managers, such as Anaconda, pyenv-virtualenv, and pipenv, should be used to ensure that existing Python environments are not changed. Users should keep the versions of the installed packages when installing additional packages for use in the reward calculations.

Figure 1: Whole workflow of ChemTSv2. In Step 1, users prepare a reward file using software packages that determine the reward according to desired molecular properties. In Step 2, users choose the parameters of the molecule generation environment, including the number of generated molecules and the prioritization of exploration and exploitation, through the configuration file. In Step 3, users excuse ChemTSv2 via the `chemtsv2` or `chemtsv2-mp` command. In the final step, users obtain the designed molecules in CSV format and can analyze the generated molecules.

**Basic usage**

Various example files with auxiliary files can be downloaded from the GitHub repository with the following command:

```
$ git clone git@github.com:molecule-generator-collection/ChemTSv2.git
```

Then, users can generate molecules by using the `chemtsv2` command in a command-line interface, as follows:
$ chemtsv2 -c config/setting.yaml

When users want to generate molecules with the parallel MCTS algorithm, run the chemtsv2-mp command using the mpiexec command, as follows:

$ mpiexec -n <NUM_PROCESSES> chemtsv2-mp -c config/setting_mp.yaml

Users need to prepare only the configuration and reward files for typical use. The processes for preparing the two files are described in the following sections.

**Reward definition**

The reward definition file should be written in Python3. A user-defined class should be inherited from a Reward base class defined in ChemTSv2. The reward class contains two static methods: `get_objective_functions()` and `calc_reward_from_objective_values()`. The former method takes a configuration parameter object in a dictionary format, has at least one inner function to calculate an objective value from a Mol object of RDKit, and returns a list of inner functions. The latter method takes a list of calculated objective values and the parameter object and returns a float value. Example code for defining the reward to maximize the Jscore is described in List 1.

**Configuration setup**

The configuration file can be written in YAML format. The basic setting options are used to configure the MCTS parameters as follows.

`c_val` is an exploration parameter of the upper confidence bound (UCB1) score. If the parameter is set to a large value, such as 1.0, a higher priority is placed on exploration, whereas if the parameter is set to a small value, such as 0.1, the priority is placed on exploitation.
**threshold_type** specifies which indicator to use as a stop criterion for molecule generation.

ChemTSv2 supports two threshold types: the number of generated molecules and the elapsed time.

**model_setting** specifies the location of two files: the JSON file that contains the architecture of the Keras RNN model and a file in H5 format that contains the weight values.

**tokens** are simplified molecular input line entry system (SMILES) tokens that are generated when the RNN model is trained. Note that the list of tokens depends on the dataset used during model training.

To efficiently explore chemical space, users can set molecule filtering criteria via the config-

Listing 1: Example of a reward definition for the Jscore maximization task

```python
import sys
import numpy as np
from rdkit.Chem import Descriptors
import sascorer
from reward.reward import Reward

class Jscore_reward(Reward):
    def get_objective_functions(conf):
        def LogP(mol):
            return Descriptors.MolLogP(mol)
        def SAScore(mol):
            return sascorer.calculateScore(mol)
        def RingSizePenalty(mol):
            ri = mol.GetRingInfo()
            max_ring_size = max((len(r) for r in ri.AtomRings()), default=0)
            return max_ring_size - 6
        return [LogP, SAScore, RingSizePenalty]

    def calc_reward_from_objective_values(values, conf):
        logP, sascore, ring_size_penalty = values
        jscore = logP - sascore - ring_size_penalty
        return jscore / (1 + abs(jscore))
```

7
uration file to skip the reward calculation for generated molecules that are filtered by the criteria. ChemTSv2 supports various filtering criteria, such as the synthetic accessibility score and Lipinski’s rule of five. Moreover, ChemTSv2 allows users to incorporate their own filtering criteria. Additionally, the configuration file can be accessed via the reward file. Therefore, users who want to use their own parameters for the reward calculation can add these parameters to the configuration file as keys and values and refer to them in the reward file.

**Analysis of generated molecules**

To facilitate the analysis of the designed molecules, the output format of ChemTSv2 is comma-separated values (CSV), one of the most common file formats. ChemTSv2 records the following seven items: ID, molecule string representation (SMILES), reward value, search tree depth, elapsed time, molecule filtering results, and objective values. Users can analyze the output with various spreadsheet software, such as Microsoft Excel and pandas package because most applications support the CSV file format.

**Algorithm**

ChemTSv2 is based on two techniques: Monte Carlo tree search (MCTS) efficiently explores the optimal molecules and a recurrent neural network (RNN) generates the SMILES strings, as shown in Fig. 2. In ChemTSv2, each node in the MCTS corresponds to one symbol in SMILES format. In the following sections, non-parallel and parallel MCTS algorithms that are implemented in ChemTSv2 are described.

**Non-parallel MCTS algorithm in ChemTSv2**

MCTS is a probabilistic tree search algorithm that consists of four steps: selection, expansion, simulation, and backpropagation, as shown in Fig. 2.
1. **Selection**: Starting with the root node, the UCB1-based policy is recursively applied to select the most promising leaf node.

\[
\text{UCB1} = \frac{w_i}{v_i} + C\sqrt{\frac{2 \log V}{v_i}} \tag{1}
\]

Here, \(w_i\) represents the cumulative reward of each node; \(v_i\) and \(V\) denote the numbers of visits to the child and parent nodes, respectively; and \(C\) is an exploration parameter.
that balances the trade-off between exploration and exploitation.

2. **Expansion**: After selecting the leaf node, the probabilities of the next atom are obtained by the RNN, and child nodes are added to the leaf node based on the probabilities. In ChemTSv2, the threshold of the cumulative probability is set to 0.995 by default.

3. **Simulation**: The simulation generates a complete SMILES string according to the partial SMILES string produced by the expanded node using the RNN. Then, the reward, $r$, is evaluated using the specified reward function.

4. **Backpropagation**: The reward is backpropagated to the root node while updating the UCB1 values ($w_i = w_i + r$) and visit counts ($v_i = v_i + 1$) of the nodes in the path.

The MCTS algorithm uses the above four steps in one round and searches for optimal molecules through multiple rounds.

**Acceleration of the MCTS algorithm with massive parallelization**

Parallelization is important because the search process is often time-consuming due to the vast search space and slow reward functions. Unlike the inherent parallelism available in deep learning models, MCTS is rarely parallelized on a large scale because of its sequential nature. Yang et al.\textsuperscript{38} introduced a massive parallel MCTS (MP-MCTS) based on a virtual loss and a hash-driven approach. UCB$vl$ has two additional variables, $t_i$ and $T$, as shown in Eq. 1. Here, $t_i$ is the number of workers currently searching in the subtree of child node $i$, and $T$ is the sum of $t_i$ of the children, as shown in Fig. 2. UCB$vl$ penalizes the score based on the number of processes searching the subtree ($t_i$) and the sum ($T$). This approach assumes that the other processes obtain zero rewards (hence the name virtual loss) and ensures that the subsequent process avoids the same path. When the time needed to obtain the optimal molecules is compared between the MP-MCTS and non-parallel MCTS algorithms, the MP-MCTS algorithm obtained candidate molecules with similar scores faster.
than the non-parallel MCTS algorithm; moreover, this increase was essentially proportional to the number of CPUs. In detail, the MP-MCTS algorithm needed only 10 minutes and 256 CPU cores, while the non-parallel MCTS algorithm took 42 hours to obtain candidate molecules with similar scores. As shown above, MP-MCTS can maintain the search quality with an increase in parallelism (the number of CPU cores) because the workers communicate the most recent reward information which makes the tree as deep as non-parallel case using the same compute time.

Recurrent neural network model

MCTS uses an RNN model to prune and sample the search space by selecting children with higher probability. Referring to the original ChemTS paper, gated recurrent units (GRU)-based RNN model was adopted to construct the RNN model. The model architecture included two-stacked GRUs with 256 units, hyperbolic tangent activation, and a dropout ratio of 0.3, and a dense layer with softmax activation. To support various demands for molecular design conditions, ChemTSv2 provides four RNN models that were trained using the curated molecules from ZINC, ChEMBL, and PubChemQC databases.

Representative applications using ChemTS

This section introduces six exemplary applications of ChemTS in various fields, including materials design, drug discovery, and analytical chemistry. First, materials design applications of ChemTS are demonstrated through QC simulations to design and experimentally validate the proposed light absorption/emission molecules and polymer electrets. Then, the drug discovery applications of ChemTS are illustrated through docking simulations and ML prediction models, focusing on ligand-based (LBDD) and structure-based (SBDD) drug designs. Finally, the utility of the ChemTS concept in analytical chemistry is highlighted through its application in identifying unknown molecules using nuclear magnetic resonance
Material design

Recent developments in computer science have advanced the concept of materials informatics, encouraging researchers to apply ML techniques to materials design. Figure 3 shows three representative materials design applications of ChemTS combined with Gaussian 16 package. ChemTS has shown great potential as a de novo molecule generator for designing functional materials by proving that the designed materials have the desired functions through laboratory experiments. In the following sections, three representative ChemTS applications for designing chromophores, fluorophores, and electret polymers are described.

Chromophore design

In 2018, Sumita et al. successfully designed synthesizable functional molecules using ChemTS coupled with density functional theory (DFT) calculations. Photofunctional organic molecules with first excited states at 200, 300, 400, 500, and 600 nm were selected as functional molecules. In this study, ChemTS optimized molecules to have a target absorption wavelength (\( \lambda \)) computed by Gaussian 16 package with DFT at the B3LYP/3-21G* level. Eighty-six molecules were generated over 10 days, and six promising molecules were evaluated according to the following criteria: (i) at least one synthetic route is reported in SciFinder and (ii) the oscillator strength obtained with time-dependent DFT is strong enough to allow the transition from the ground state to the first excited state. According to the results, five of the six molecules were experimentally validated, and their first peaks in the experimental spectra were near the target absorption wavelengths (Figure 3A). In addition to controlling the molecular absorption wavelength, we experimentally validated that DFT-coupled ChemTS is useful for digging potential chromophores to absorb long wavelength light. These results clearly show that ChemTS coupled with DFT calculations could find synthesizable functional
molecules. Users can perform the molecule generation using ChemTSv2 with QCforever, and the following setting file is prepared:

```
$ chemtsv2 -c config/setting_chro.yaml
```

Figure 3: Applications of ChemTS in materials design. A. Experimental UV–vis absorption spectra (purple) and theoretical spectra (green) computed by Gaussian 16 package at the B3LYP/3-21G* level for four representative ChemTS-designed compounds. The computational spectra were smoothed by a Gaussian function and arbitrarily scaled for comparison with the experimental spectra. The black dashed lines indicate the target wavelengths. B. (a) Representative example of an unreported fluorescent molecule (PC; 3-[3-(pyridin-3-yl)pyrazolo[1,5-a]pyrimidin-6-yl]-2H-chromen-2-one) designed using ChemTS. (b) Images of PC dissolved in dichloromethane (DCM) and dimethyl sulfoxide (DMSO) under room light (upper panel) and UV light at 365 nm (lower panel). Adapted with permission from Sumita et al. under the terms of a Creative Commons Attribution Licence 4.0 (CC BY). C. Monomer structures of a fluorinated polymer CYTOP CTX series. The surface potential of a 15 µm-thick sample is maintained at -3.0 kV for more than 800 h after charging with a bias voltage of -3.5 kV. The curing temperature is 230°C. Adapted with permission from Zhang et al. under the term of [RightsLink](#).
Fluorophore design

Sumita et al. also designed fluorescent molecules, using parallel ChemTS with DFT calculations. Fluorescent molecules emit light and have attracted attention in various fields, such as organic light-emitting diodes, sensors, and bioimaging. The above study aimed to design fluorescent molecules with an upper absorption wavelength of 700 nm, an upper fluorescence wavelength of 1200 nm, and an oscillator strength of 0.01. The above properties were calculated by Gaussian 16 package with DFT at the B3LYP/3-21G* level. Since the DFT calculations to explore chemical space were computationally expensive, parallel ChemTS with 1024 cores was applied to generate thousands of molecules in less time. The parallel ChemTS designed 3643 molecules over five days, and 87 candidate molecules were selected for synthesis according to the following criteria: (i) the minimum oscillator strength for the transition from the singlet ground ($S_0$) state to the singlet first excited ($S_1$) state is greater than 0.1; (ii) the molecule emits fluorescence with a wavelength of over 400 nm and an oscillator strength of more than 0.01; and (iii) the difference between the absorption and fluorescence wavelengths in the $S_1$ states is greater than 100 nm to guarantee discriminability. All 87 molecules exhibited naked-eye-detectable fluorescence (an example is shown in Fig. 3B). Additionally, 80 molecules of them were not found in SciFinder and had no common skeletons of known fluorescent molecules. The result implied that parallel ChemTS could discover novel synthesizable molecules with previously unknown skeletons to chemists. Users can perform the molecule generation using ChemTSv2 with QCforever, and the setting file is prepared as follows:

```
$ chemtsv2 -c config/setting_fluor.yaml
```

Polymer electret design

Figure C shows polymer electrets design assisted by ChemTS. Polymer electrets are considered promising electromechanical materials due to their low weight, large quasi-piezoelectric sensitivity, and flexibility.\textsuperscript{53} Energy harvesting, which is a potential application of poly-
mer electrets, converts ambient environmental energy into electrical energy and is used in various devices, such as wearable devices and wireless electronics. To improve the power output of polymer electrets, polymer electrets with high surface charge density, durability, and thermal stability should be developed. Sakane et al. investigated fluorinated polymer electrets, perfluoro(1-butenyl vinyl ether) polymer (CYTOP), with different end groups (CF$_3$, carboxyl, and amidosilyl) and found that the amidosilyl end group improves the surface charge density. To explore end groups that improve the surface charge density of CYTOP, Zhang et al. utilized ChemTS to perform functional group enrichment analysis based on the generated molecules. In their study, ChemTS was used to optimize the end group of CYTOP to maximize the electron gain (EG) energy calculated by Gaussian 16 package with DFT at the B3LYP/STO-3G level based on the optimized structure at the UFF level. The functional group enrichment analysis revealed that the molecules with high EG energy were enriched with the hydroxyl group. Based on their findings, they chose N-(3-aminopropyl)diethanolamine (APDEA) as a component of the end group and synthesized a new polymer electret (CTX-A/APDEA). The surface potential of CYTOP CTX-A/APDEA was more stable than the surface potentials of the other three CYTOPs according to the experimental results (Figure 3C, right panel).

Drug design

Computer-aided drug design (CADD) techniques are essential in the preliminary stage of drug discovery and are generally categorized into LBDD and SBDD methods. Figure shows two representative applications of ChemTS in LBDD and SBDD, respectively. LBDD methods utilize structural and physicochemical information acquired from a set of ligands with potency against a relevant target, such as a receptor and enzyme. SBDD methods employ knowledge obtained from the binding poses and estimated interactions of known protein-ligand complexes to design drug candidates. In the following sections, the methods to perform LBDD and SBDD using ChemTS are described.
Figure 4: Applications of ChemTS in drug design. A. The molecule generation optimization process for EGFR-selective inhibitors (blue line) is shown. Each line shows moving averages that were calculated with a window size of 1000 using the average results of three runs. Adapted with permission from Yoshizawa et al. under the terms of a Creative Commons Attribution Licence 4.0 (CC BY-NC-ND). B. The left panel shows the molecule generation optimization process for c-Abl inhibitors targeting the c-Abl protein (PDB ID: 1IEP). The red line shows the AutoDock Vina-predicted binding affinity (-12.21 kcal/mol) obtained by redocking imatinib, which is included in the complex structure. An example binding mode of the generated molecule is shown in the right panel using an open-source PyMOL package. The molecule and protein structures are shown in light blue and grey, respectively. The four residues, Glu286, Thr315, Met318, and Asp381, reported as important in the paper are indicated in orange. Each blue dashed line indicates a hydrogen bond.

Ligand-based drug design

As an example of LBDD, Yoshizawa et al. designed selective inhibitors for kinase homologues using ChemTS combined with the light gradient boosting machine (LightGBM), an ML-based prediction model. In the study, ChemTS successfully proposed tyrosine kinase-
selective inhibitor candidates considering 18 objectives, including inhibitory activities against nine tyrosine kinases: epidermal growth factor receptor (EGFR), platelet-derived growth factor receptor beta (PDGFRβ), ephrin type-B receptor 4 (EPHB4), Abelson tyrosine-protein kinase (ABL), fibroblast growth factor receptor 1 (FGFR1), vascular endothelial growth factor receptor 2 (VEGFR2), lymphocyte-specific tyrosine-protein kinase (LCK), proto-oncogene tyrosine-protein kinase (SRC), and receptor protein-tyrosine kinase erbB-2 (ERBB2). The values of the inhibitory activities are based on LightGBM prediction models. Figure 4A shows the optimized EGFR-selective inhibitor design. In the optimization process, ChemTS successfully generated molecules with high inhibitory activity against EGFR while suppressing the inhibitory activity against the other eight kinase homologues. Users can perform the molecule generation using ChemTSv2 with LightGBM, and the setting file is prepared as follows:

```
$ chemtsv2 -c config/setting_dscore.yaml
```

**Structure-based drug design**

As an example of SBDD, Ma et al. performed structure-aware molecule generations using ChemTS combined with docking simulation software, namely, rDock. Inspired by their work, we generated molecules with ChemTSv2 to design c-Abl inhibitors using AutoDock Vina instead of rDock. The protein structure of c-ABL (PDB ID: 1IEP) in the docking simulation was obtained from AutoDock Vina’s GitHub repository (https://github.com/ccsb-scripps/AutoDock-Vina). A 3D grid space representing the potential binding site of a protein was defined as follows: center coordinates of 15.190, 53.903, and 16.917 angstrom and a size of 35 by 35 by 35 angstrom. To generate 3D structures of ligands from a SMILES string generated by the rollout, an ETKDG algorithm implemented in RDKit was used. Other experimental conditions were applied according to previous work.

Figure 4B shows the optimization of the AutoDock Vina-predicted binding affinity of the molecules generated by ChemTSv2 (left panel) and an example binding mode of a generated
molecule (right panel). After approximately 10,000 molecules were generated, molecules with binding affinities better than the affinity obtained by redocking imatinib (-12.21 kcal/mol), which is included in the complex structure (PDB ID: 1IEP), were frequently generated. To confirm that molecules with reasonable binding modes were generated, the molecules at the end of the optimization progress were visually inspected. As shown in the right panel of Figure 4B, the generated molecule (blue) formed four hydrogen bonds with important residues (orange), namely, Glu286, Thr315, and Asp381, as reported in c-Abl. These results indicate that ChemTSv2 combined with AutoDock Vina has the potential to design promising c-Abl inhibitors. Users can perform the molecule generation using ChemTSv2 with AutoDock Vina, and the setting file is prepared as follows:

```bash
$ chemtsv2 -c config/setting_vina_binary.yaml
```

**Analytical chemistry**

Figure 5 shows a representative application of a ChemTS derivative in analytical chemistry. Analytical chemistry contributes to various science fields by identifying what matter is and quantifying how much matter is present in a given sample. NMR spectroscopy is a common technique for identifying what matter is that is ubiquitously used in chemical laboratories. Although some techniques to automatically identify molecules according to their NMR spectra have been developed, the efficient identification of unknown molecules, i.e., molecules that are unknown to the public, still requires chemists’ intervention. Zhang et al. considered that 1H-NMR spectra are distinct characteristics of individual molecules and attempted to identify unknown molecules according to their spectra using a parallel ChemTS derivative. In their study, a score considering the similarity between the NMR spectrum of an unknown molecule and the simulated spectrum of a generated molecule was used as a reward. The similarity was evaluated according to the Wasserstein distance, which is a metric that is commonly used to describe the distance between two distributions. The 1H-NMR spectrum of the generated molecule was computed by Gaussian 16 package with DFT.
at the B3LYP/3-21G* level based on the optimized structure at the UFF level. Although the study was proof-of-concept work and requires further improvements and evaluations, the parallel ChemTS derivative successfully identified some unknown molecules.

Figure 5: Concept workflow of ChemTS identifying an unknown molecule according to its NMR spectrum. ChemTS attempts to design molecules with NMR spectra that are as similar as possible to the spectra of the unknown molecule using the Wasserstein distance-based score (WS). WS values closer to 1 indicate higher similarity between the unknown NMR spectrum and the simulated NMR spectrum of the generated molecule. The represented spectra are computational spectra smoothed by Lorentzian functions. The $^1$H-NMR spectrum of the generated molecule was computed by Gaussian 16 package with DFT at the B3LYP/3-21G* level.

Summary and Outlook

ChemTSv2 provides a generic Python-based platform for generating molecules with desired properties. ChemTSv2 employs the following functions: an easy-to-define reward file that can be prepared using any software package, an easy-to-switch single process and massive parallel modes, and an easy-to-run with only a configuration file. Various ChemTS applications in materials design, drug design, and analytical chemistry were summarized in this paper, and these results clearly indicate that ChemTS, the former version of ChemTSv2, has high potential in designing molecules with desired properties through simulation and ML techniques. Although these features have previously been provided as separate inde-
pendent packages, ChemTSv2 implements essentially all of these features, enabling users to
design desired molecules using parallel and non-parallel ChemTS by just focusing on the
reward setting.

To design highly reliable molecules, simulation techniques with high computational costs,
such as QC and molecular dynamics (MD) simulations, will become crucial factors more and
more. Although ChemTSv2 with the massive parallel mode can perform such simulations,
ChemTSv2 currently provides a few examples using simulation packages, such as Gaussian
16 and AutoDock Vina. Hence, we will implement other commonly used software, such as
Gromacs \(^{62}\) (MD simulation) and GAMESS \(^{63}\) (QC simulation), in ChemTSv2 and evaluate
their performance in generating the desired molecules. Additionally, we will investigate
difficult ChemTS applications, such as designing polymer electrets (Fig. 3C) and identifying
molecules using NMR (Fig. 5), in ChemTSv2. Moreover, developing a graphical user interface
(GUI) to use ChemTSv2 is essential to extend applicability to users with various levels of
programming skills. We plan to develop a user-friendly GUI with functions that allow users
to execute Step 1 through 4 (Fig. 1) via GUI.

Author Contributions

Shoichi Ishida: conceptualization (lead), software (lead), writing - original draft (lead),
writing - review & editing (lead). Tanuj Aasawat: software (lead), writing - original draft
(supporting), writing -review & editing (supporting). Masato Sumita: conceptualization
(lead), software (supporting), writing - original draft (supporting), writing -review & editing
(supporting). Michio Katouda: software (supporting). Tatsuya Yoshizawa: software
(supporting). Kazuki Yoshizoe: software (lead), writing -review & editing (supporting).
Koji Tsuda: supervision (lead), conceptualization (lead), writing -review & editing (support-
ing). Kei Terayama: supervision (lead), conceptualization (lead), software (support-
ing), writing - review & editing (lead).
Software availability

The ChemTSv2 application is publicly available on GitHub at https://github.com/molecule-generator-collection/ChemTSv2 under the MIT License. The README file in the GitHub repository provides information about how to setup and use the application.

Acknowledgement

The authors thank Dhiraj Kalamkar (Parallel Computing Lab - India, Intel Labs) for his valuable discussion and comments from the perspective of an expert in parallel computing. This work was conducted in “Development of a Next-generation Drug Discovery AI through Industry-Academia Collaboration (DAIIA)” and a Research Support Project for Life Science and Drug Discovery (project ID: JP22ama121023) supported by Japan Agency for Medical Research and Development (AMED) under grant no.JP22nk0101111. This work was also supported by MEXT as a “Program for Promoting Researches on the Supercomputer Fugaku (Application of Molecular Dynamics Simulation to Precision Medicine Using Big Data Integration System for Drug Discovery)” and Data Creation and Utilization Type Material Research and Development Project Grant Number JPMXP1122683430. M.K. and K.Y. were supported by JSPS KAKENHI Grant Number JP19K05443 and JP20H04251, respectively.

Supporting Information Available

References


fluorescent molecule based on quantum chemical computation and machine learning. 


25


(50) Frisch, M. J. et al. Gaussian®16 Revision C.01. 2016; Gaussian Inc. Wallingford CT.


