Taylor & Francis
Taylor & Francis Group

Science and Technology of Advanced Materials: Methods

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tstm20

NIMS-OS: an automation software to implement a closed loop between artificial intelligence and robotic experiments in materials science

Ryo Tamura, Koji Tsuda & Shoichi Matsuda

To cite this article: Ryo Tamura, Koji Tsuda & Shoichi Matsuda (2023) NIMS-OS: an automation software to implement a closed loop between artificial intelligence and robotic experiments in materials science, Science and Technology of Advanced Materials: Methods, 3:1, 2232297, DOI: 10.1080/27660400.2023.2232297

To link to this article: https://doi.org/10.1080/27660400.2023.2232297

SCIENCE AND TECHNOLOGY OF ADVANCED MATERIALS: METHODS
2023, VOL. 3, NO. 1, 2232297
https://doi.org/10.1080/27660400.2023.2232297

NIMS-OS: an automation software to implement a closed loop between artificial intelligence and robotic experiments in materials science

Ryo Tamura[a,b], Koji Tsuda[a,b] and Shoichi Matsuda[c,d]

[a]Center for Basic Research on Materials, National Institute for Materials Science, Tsukuba, Japan; [b]Graduate School of Frontier Sciences, The University of Tokyo, Chiba, Japan; [c]Research Center for Energy and Environmental Materials (GREEN), National Institute for Materials Science, Tsukuba, Japan; [d]Center for Advanced Battery Collaboration, Research Center for Energy and Environmental Materials (GREEN), National Institute for Materials Science, Tsukuba, Japan

ABSTRACT

NIMS-OS (NIMS Orchestration System) is a Python library created to realize a closed loop of robotic experiments and artificial intelligence (AI) without human intervention for automated materials exploration. It uses various combinations of modules to operate autonomously. Each module acts as an AI for materials exploration or a controller for a robotic experiments. As AI techniques, optimization tools for PHYSics based on Bayesian Optimization (PHYSBO), BoundLess Objective-free eXploration (BLOX), Phase Diagram Construction (PDC), and Random Exploration (RE) methods can be used. Moreover, a system called NIMS automated robotic electrochemical experiments (NAREE) is available as a set of robotic experimental equipment. Visualization tools for the results are also included, which allows users to check the optimization results in real time. Newly created modules for AI and robotic experiments can be added easily to extend the functionality of the system. In addition, we developed a GUI application to control NIMS-OS. To demonstrate the operation of NIMS-OS, we consider an automated exploration for new electrolytes. NIMS-OS is available at https://github.com/nimsos-dev/nimsos.

IMPACT STATEMENT

NIMS-OS can create a closed loop of robotic experiments and artificial intelligence for automated materials exploration. The GUI application to control NIMS-OS, as well as the Python package, were developed.

ARTICLE HISTORY
Received 26 April 2023
Revised 29 May 2023
Accepted 8 June 2023

KEYWORDS
NIMS-OS; robotic experiments; artificial intelligence; electrochemistry; materials informatics

1. Introduction

The integration of artificial intelligence (AI) and robotic experiments is essential to realize automated materials exploration. If an AI system can take on some information tasks conventionally performed by human researchers and robotic systems can then execute the required physical tasks, experiments for materials exploration can proceed automatically. Such a platform may be expected to discover many novel materials and lead to substantial innovation in materials science. In recent years, significant progress has been made in the development of AI techniques and robotic devices suitable for materials exploration.

Since the launch of the Materials Genome Initiative [1], AI techniques have been actively used for materials

CONTACT Ryo Tamura tamura.ryo@nims.go.jp Center for Basic Research on Materials, National Institute for Materials Science, 1-1 Namiki, Tsukuba, Ibaraki 305-0044, Japan; Koji Tsuda tsuda@k.u-tokyo.ac.jp Graduate School of Frontier Sciences, The University of Tokyo, 5-1-5 Kashiwa-no-ha, Kashiwa, Chiba 277-8561, Japan; Shoichi Matsuda matsuda.shoichi@nims.go.jp Research Center for Energy and Environmental Materials (GREEN), National Institute for Materials Science, 1-1 Namiki, Tsukuba, Ibaraki 305-0044, Japan

This article has been republished with minor changes. These changes do not impact the academic content of the article.
Supplemental data for this article can be accessed online at https://doi.org/10.1080/27660400.2023.2232297.

exploration [2–4]. In general, materials exploration can be regarded as the problem of finding optimal materials from among a materials search space. The elements to be used in the search space must be configured, along with its composition range, process parameter range, and so forth. To solve this problem, black-box optimization methods are useful [5], and various methods have been developed and applied to fit various needs. Bayesian optimization (BO) is among the most frequently used methods in materials science [6–8]. In this method, promising materials can be selected in the materials search space using the predictions of their properties and the uncertainty of these predictions evaluated by Gaussian process regression. Using BO, various real materials, such as Li-ion conductive materials [9], multilayered metamaterials [10], halide perovskite [11], superalloys [12], and electrolytes [13], have been explored. BO is also used for the automation techniques for analysis of materials [14,15]. In addition, many methods have been proposed for black-box optimization in materials exploration, such as genetic algorithms [16,17], Monte Carlo tree search [18], rare event sampling [19], and algorithms using an Ising machine [20–22]. In the future, many more innovative methods are expected to be developed.

Robotic experiments have progressed to realize laboratory automation of chemical analysis and high-throughput screening in the field of biology [23–26]. Various types of automated analyzers and pipetting devices have been developed, and robotic arms have been used as a transport system to connect these systems. Moreover, robotic technology has been used to explore novel materials, such as thin-film materials [27,28], battery electrolytes [13,29], and photocatalysts [30]. These studies used BO to automate the proposal of promising experimental conditions. This enables a closed loop of robotic experiments and AI that can perform automated materials exploration without human intervention. This approach involves some key advantages, such as the ability to generate materials data of uniform quality and the absence of human error. In contrast, at present, robotics systems are limited in their ability to perform complex material synthesis tasks that require the skills of experts. Thus, further innovation in robotic devices will be important.

In addition to AI and robotic technologies, the control systems and software used to interlink them are also an important element to realize a closed loop without human intervention. Generally, different AI algorithms should be used depending on the motivation of a materials exploration task. Furthermore, the procedure to control the devices should depends on the nature and characteristics of the robotic systems used. Therefore, control software has thus far been developed on a case-by-case basis for different AI algorithms and robotic systems.

In this study, we developed NIMS-OS (NIMS Orchestration System) to realize a closed loop between AI models and robotic experiments, with the aim of establishing a generic control software system. Although this software was written in the Python programming language, we also developed a GUI version to improve operability after installation. NIMS-OS treats each AI algorithm and each robotic system as separate modules (see Figure 1). This enables the implementation of a closed loop with any combination of these modules. If modules for new AI algorithms or robotic systems are prepared, new closed-loop systems can be easily controlled via NIMS-OS. One of the advantages of developing such generic control software is the establishment of technical standards for automated materials exploration. For AI algorithms, we determined standard formats for the input and output. Algorithms created according to the standard format can be immediately tested using any currently available robotic system. Specifically, we developed a standard format in which all the experimental conditions to be explored are listed in advance, and the appropriate experimental conditions that have not yet been tested are selected from the list by AI algorithms. The advantage of this approach is that it enables automated materials exploration utilizing materials databases. When utilizing materials databases, the compositional and structural information needs to be converted into materials descriptors, which serve as the materials search space. However, this search space generated from the materials databases cannot be solely defined by a continuous parameter space, and it requires a selection from the pre-listed descriptors. Of course, optimization of continuous parameters can still be handled approximately by preparing a list of grid points that discretize the

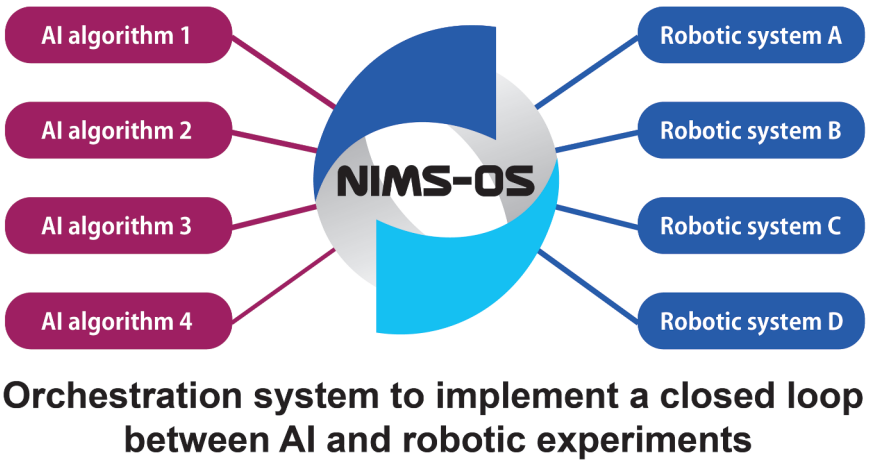

Figure 1. Image of the combinations of AI algorithms and robotic systems via NIMS-OS.

continuous parameters. Furthermore, we expect this work to contribute to the development of new AI algorithms for automated materials exploration. For robotic systems, we expect modules developed based on NIMS-OS to increase the commonality of operational procedures, leading to cost reductions as new robotic experimental devices are introduced. Note that ChemOS [31] is similar to NIMS-OS; it was developed as an automation system in the field of chemistry. ChemOS specializes in BO within a defined continuous or discretized parameter space and includes several default modules for various BO methods. On the other hand, NIMS-OS offers the capability to perform automated materials explorations not only within a defined parameter space but also utilizing materials databases. In addition to BO, NIMS-OS also incorporates several default implementations of black-box optimization methods to deal with different motivations in materials explorations.

Let us briefly introduce the specifications of NIMS-OS. First, a candidates file listing experimental conditions as a materials search space should be prepared in advance. A closed loop is formed according to the following three steps (see Figure 2):

Step 1: Select promising experimental conditions from the candidates file using an AI model.
Step 2: Create an input files for the robotic experiments and execute the experiments.
Step 3: Analyze the output from the experiments and update the candidates file based on the experimental results.

Currently, the following AI algorithms are used as modules, which are available for Step 1: (i) optimization tools for PHYSics based on Bayesian Optimization (PHYSBO) [32], (ii) BoundLess Objective-free eXploration (BLOX) [33], and (iii) Phase Diagram Construction (PDC) [34] methods and Random Exploration (RE) approach can be selected according to the purpose of materials exploration effort. For Steps 2 and 3, a STANdard module (STAN) is provided for robotic experiments, which enables operation checks even without devices, along with a module for NIMS Automated Robotic Electrochemical Experiments (NAREE) [13,35]. We plan to continue developing additional modules for this system.

The reminder of this study is organized as follows. Section 2 describes the preparation of a candidates file storing experimental conditions. In Section 3, we introduce the available modules for the AI and robotic experiments in NIMS-OS. Section 4 details the use of the Python code, and the usage of the GUI version is explained in Section 5. In Section 6, as a demonstration, the results of an autonomous electrolyte exploration via a closed-loop approach using PHYSBO and NAREE in NIMS-OS are described. Finally, Section 7 concludes this work with some discussion and suggest some important avenues for further research.

2. Preparation of candidates for experimental conditions

A major feature of NIMS-OS is that a data file listing candidate experimental conditions is prepared in advance (we refer to this data file a candidates file). In general, because there are many candidates, conducting experiments in all possible conditions is impractical. Thus, automated materials exploration proceeds by selecting promising experimental conditions from these listed candidates. This makes the closed-loop strategy more generalizable. That is, a variety of exploration motivations and robotic systems can be handled by NIMS-OS.

The experimental condition is expressed as a real-valued vector $\mathbf{x}_i \in \mathbb{R}^d$. This condition is prepared with information such as the compositions and structures of materials and the processes required to synthesize them. If the number of candidates for the experimental conditions is N, the dataset for candidates is defined as

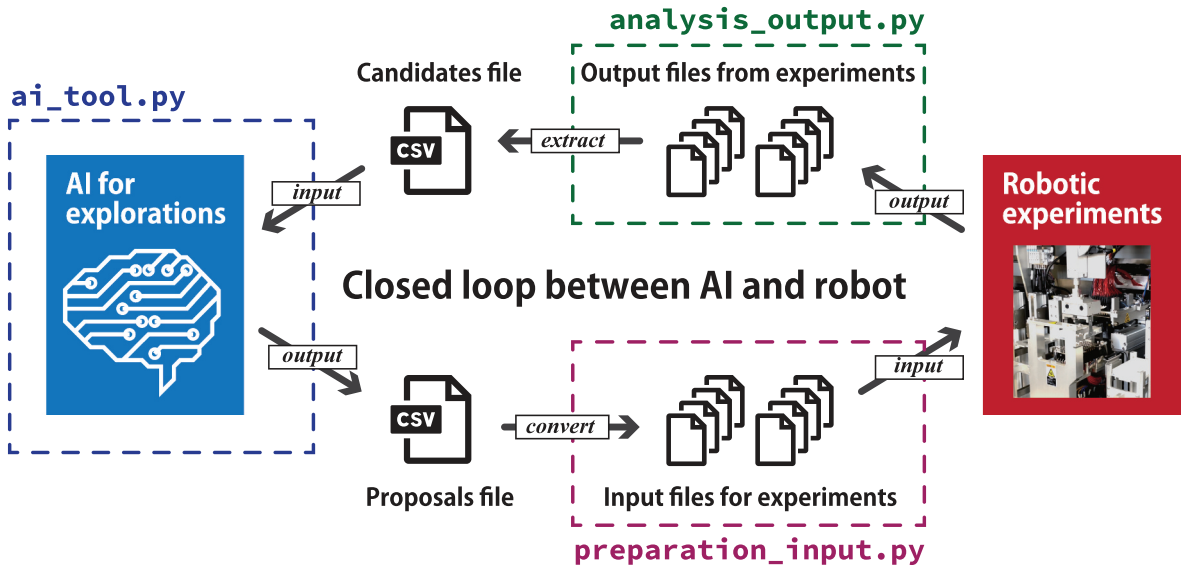

Figure 2. Procedures in NIMS-OS and roles of each Python scripts.

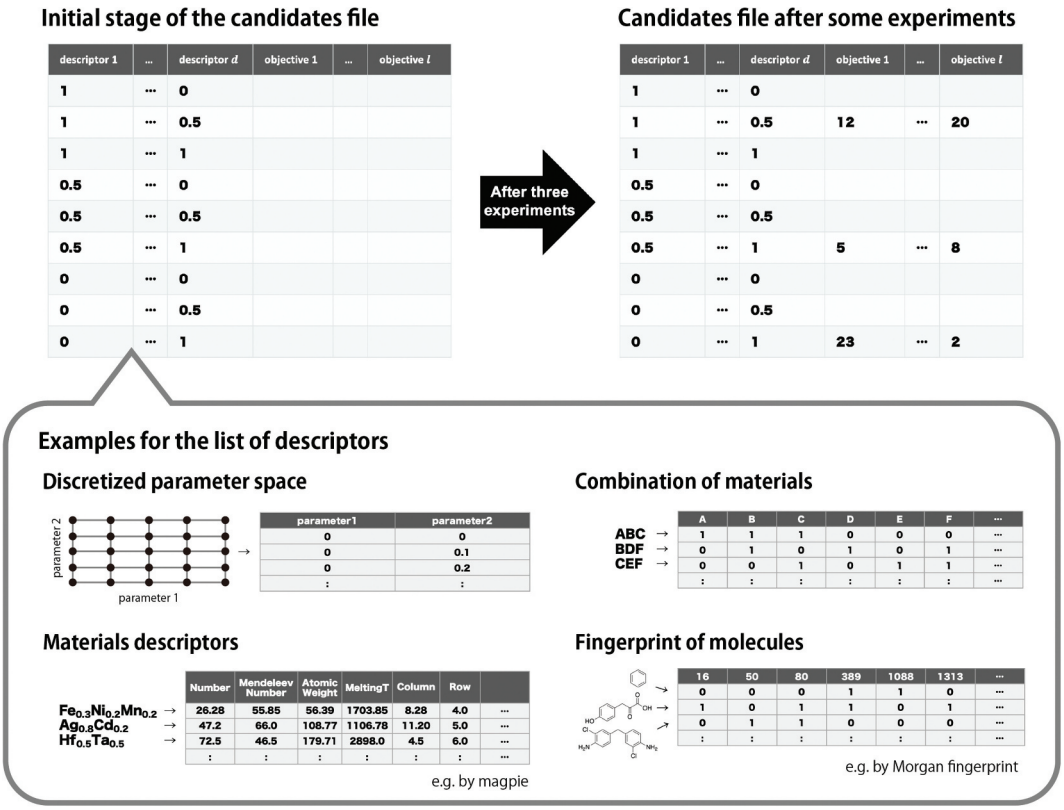

Figure 3. (Top panels) Examples of the candidates files of the initial stage and that after some experiments. Here, an example for the case that $N = 9$ is shown. (Bottom panels) Examples for the list of descriptors depending on the types of search space. If the continuous parameter space is considered, $D = \{\mathbf{x}_i\}_{i=1,\ldots,N}$ is the discretized parameters. When the combination of materials is the search space, the bit strings where the material used is represented by 1 and the material not used is represented by 0 are prepared in $D = \{\mathbf{x}_i\}_{i=1,\ldots,N}$. Furthermore, materials descriptors from compositions obtained by such as magpie [36,37] and fingerprint of molecules obtained by such as RDKit [38] would be used as $D = \{\mathbf{x}_i\}_{i=1,\ldots,N}$.

$D = \{\mathbf{x}_i\}_{i=1,\ldots,N}$. The initial candidates file is created by this dataset D. An example of a candidates file with l objective functions is presented in Figure 3. All the candidates of D are written in the first d columns. In this part, there should be no empty spaces. The next l columns are used for the objective function values. In this part, at the initial stage, all cells are empty because experiments have not been performed for all the experimental conditions.

In NIMS-OS, some promising conditions are selected from among those listed in the candidates file using AI models (available algorithms are described in Section 3.1). When the values of objective functions are obtained by performing experiments, the objective functions in the candidates file are updated, accordingly. That is, when the experiments are completed for M experimental conditions, only results for M conditions are entered at the l columns for the objective functions. Thus, at the next step, the experimental conditions are selected from among $N - M$ candidates.

3. Modules in NIMS-OS

In this section, we introduce the modules included in NIMS-OS for AI algorithms and robotic systems. In the present work, we prepared four and two types of modules as AI algorithms and robotic systems, respectively.

3.1. AI algorithms

To select promising experimental conditions, three types of AI algorithms are implemented as standard in NIMS-OS. In addition, random exploration can be selected. Each algorithm is briefly explained in this subsection. In the future, more algorithms will be made available.

3.1.1. Bayesian optimization: PHYSBO

Bayesian optimization (BO) is an optimization technique using machine learning prediction. In this method, by using Gaussian process regression, the value of an objective function is predicted when the experimental conditions are input. The next promising experimental conditions are then selected based on the prediction values. Here, because the Gaussian process can evaluate not only the mean value of the prediction but also its variance, an acquisition function defined by mean and variance can be used to make the selection. In NIMS-OS, BO can be performed using the Python package PHYSBO [32]. PHYSBO supports single- and multi-

objective optimizations, and multiple proposals are calculated. Note that the number of objective functions is recommended to be no more than three due to excessively large computational time with higher values. In NIMS-OS, Thompson sampling is used to define the acquisition function for rapid calculation. The key point in using PHYSBO is that the exploration is performed to maximize the objective functions. Thus, if a material with the smaller properties is explored, we need to add a negative value to the objective functions.

3.1.2. Boundless objective-free exploration: BLOX

BLOX is a Python package that performs boundless objective-free exploration. It is based on an algorithm designed to select the next experimental conditions, to perform uniform sampling in the space of the objective functions. For materials science, curious materials can be found using BLOX. Specifically, BLOX trains machine learning models to predict objective functions from experimental conditions. Experimental conditions that realize uniform sampling in the space of objective functions are found based on the Stein discrepancy evaluated using the prediction results. In NIMS-OS, a modified version of the BLOX algorithm that can propose multiple candidates is implemented. To select multiple candidates, after the experimental condition with the largest Stein discrepancy is selected, another condition is selected when the predicted values of the selected condition are regarded as a correct value. This procedure is iterated, and we obtain multiple proposals. In NIMS-OS, random forest regression is used as a prediction model. Although BLOX can handle any number of objective functions, it is recommended that the number of the objective functions be limited to three or four, because exploration in more dimensions requires more time. BLOX has been used to search chemical spaces [33] and to explore superhard materials [39].

3.1.3. Phase diagram construction: PDC

PDC is a Python package that can create a detailed phase diagram with a small number of experiments. To investigate a phase diagram, PDC proposes promising experimental conditions for the next experiment by using active learning. Specifically, uncertainty sampling based on the label propagation method finds uncertain points in the phase diagram, and these uncertain points are proposed for the next experiments. PDC was developed to propose multiple experimental conditions for batch experiments [40]. In NIMS-OS, the least confident score is used as an uncertainty score to evaluate uncertain points. Note that, for PDC, the objective function is the phase name or an index of phases, and thus only a one-dimensional objective function can be specified in the candidates file. PDC has been used to create new phase diagrams for the growth conditions of thin film [41] and to determine large and small areas of creep phenomena in polymer materials [42].

3.1.4. Random exploration: RE

In RE, the next candidate experimental condition is selected randomly. This approach can be used to generate initial data before executing AI algorithms when no experimental data have yet been recorded. Furthermore, it can also be used to generate data for comparison as new AI algorithms can be developed.

3.2. Robotic experiments

The module for robotic experiments comprises two Python scripts. The first script creates input files for robotic experiments according to the experimental conditions selected by the AI and commands a robot to begin the experiment. The second script analyzes the experimental results when the experiments are finished, and updates the candidates file. At present, two types of modules are implemented in NIMS-OS.

3.2.1. Standard module for robotic experiments: STAN

The STANdard module (STAN) is a virtual implementation of the procedure for conducting robotic experiments. Thus, NIMS-OS can be run virtually using this module, even without a robotic device. In this module, the following steps are executed:

(i) Create the input files for the robotic experiments in an appropriate folder according to the experimental conditions selected by the AI. In this standard module, we simply create a text file with a date as its name.
(ii) Send a signal to the robotic system to begin the experiments. Depending on the machine, various cases can be considered, such as sending a start signal via serial communication. In this standard module, we assume that the experiments are begun by storing the inputend.txt file in the specified folder.
(iii) Wait until the robotic experiments are completed. This step includes various operations, such as receiving signals from the robot when the experiment is finished. This standard module assumes that the robot outputs outputend.txt file to indicate that the experiment is finished, and NIMS-OS continues waiting until this file appears.
(iv) Read the files of experimental results and extract the values of objective functions. Here, the case of simply reading results.csv, which contains the objective function values, is implemented.
(v) Update the candidates file according to the values extracted in (iv).

Steps (i) and (ii) are performed by preparation_input.py, and analysis_output.py conducts steps (iii)-(v). In

practice, for use with actual robotic systems, new modules can be created according to this standard module.

Additionally, this module can also facilitate closed-loop materials exploration between AI and experiments for processes that are time-consuming and cannot be partially automated. The procedure is as follows: When the proposals.csv file is generated, NIMS-OS automatically enters a sleep mode until experimental results are obtained. Based on the information in proposals.csv, the corresponding manual experiments are conducted. Once the objective function values are obtained through the experiments, a results.csv file is created, containing the objective function values corresponding to each line in proposals.csv. The results.csv file, along with an empty file named outputend.txt, is stored in the specified folder where the experimental results are output. Subsequently, NIMS-OS restarts and generates a new proposals.csv file.

3.2.2. NIMS automated robotic electrochemical experiments (NAREE) system: NAREE

As a robotic system for materials science, the NIMS Automated Robotic Electrochemical Experiments (NAREE) system [13,35] can be used in NIMS-OS. NAREE comprises a liquid-handling dispenser, an electrochemical measurement unit, and a robotic arm. By using a microplate-based electrochemical cell equipped with electrodes, the performance of electrolytes prepared by mixing solution by a liquid handling dispenser is electrochemically evaluated in a high-throughput manner. This module was developed according to the procedures of standard module explained in Section 3.2.1.

4. Usage of the NIMS-OS Python version

4.1. Install

NIMS-OS is written in Python3 programming language (version 3.6 or higher is required), and it can be installed via PyPI as follows:

```
$ python3 -m pip install nimsos
```

If this installation is successful, the following packages are also installed or updated automatically:

- Cython
- matplotlib
- numpy
- physbo
- scikit-learn
- scipy

4.2. Basic usage

We show a small example program (Program 1) in which PHYSBO is performed. In this program, assuming no experimental results in the candidates file, random exploration is performed in the first cycle.

4.2.1. Assignment of parameters and candidates file

First, the parameters for closed-loop experiments are defined. For example, when the number of objective functions is two, the number of proposals for each cycle is two, and the number of cycles is three, we define this in the code as follows:

```
ObjectivesNum = 2
ProposalsNum = 2
CyclesNum = 3
```

Next, we specify a csv file containing the candidates of experimental conditions, which is prepared according to Section 2.

```
candidates_file = "./candidates.csv"
```

The name of the file that will contain the experimental conditions selected by the AI is as follows:

```
proposals_file = "./proposals.csv"
```

We specify the folder name where the input files for the robotic experiments are stored and the folder name where the results from the experiments are output, respectively, as follows:

```
input_folder = "./EXPInput"
output_folder = "./EXPOutput"
```

4.2.2. Execution of AI

`nimsos.selection` is a class to select the next experimental conditions with the help of the AI. For example, `nimsos.selection` is used as follows:

```
nimsos.selection(method = "PHYSBO",
    input_file = candidates_file,
    output_file = proposals_file,
    num_objectives = ObjectivesNum,
    num_proposals = ProposalsNum)
```

The parameters of the method in this class indicate the module for AI algorithms. For method, `'PHYSBO'` (Bayesian optimization), '`BLOX`' (objective free search), '`PDC`' (phase diagram construction), and '`RE`' (random exploration) are specified. The experimental conditions are selected from the data without the values of objective functions among `input_file`. In addition, selected conditions are outputted to `output_file`. For `num_objectives`, the number of objectives is input, and the number of proposals is specified as `num_proposals`. In general, although many hyperparameters should be considered to use the AI, they are determined automatically in NIMS-OS. Note that if there are no experimental results in the candidates file, only '`RE`' is used. For '`PHYSBO`', '`BLOX`', and '`PDC`', some values of objective functions must be stored in the candidates file.

Program 1 Small example of NIMS-OS for Bayesian optimization

```
import nimsos

ObjectivesNum = 2
ProposalsNum = 2
CyclesNum = 3

candidates_file = "./candidates.csv"
proposals_file = "./proposals.csv"

input_folder = "./EXPInput"
output_folder = "./EXPOutput"

for K in range(CyclesNum):

    if K==0:
        method = "RE"
    else:
        method = "PHYSBO"

    nimsos.selection(method = method,
                     input_file = candidates_file,
                     output_file = proposals_file,
                     num_objectives = ObjectivesNum,
                     num_proposals = ProposalsNum)

    nimsos.preparation_input(machine = "STAN",
                             input_file = proposals_file,
                             input_folder = input_folder)

    nimsos.analysis_output(machine = "STAN",
                           input_file = proposals_file,
                           output_file = candidates_file,
                           num_objectives = ObjectivesNum,
                           output_folder = output_folder)
```

4.2.3. Preparation of input files for robotic experiments and execution of experiments

`nimsos.preparation_input` is a class to prepare the input files for robotic experiments and send the start message to robot. For example, `nimsos.preparation_input` is used as follows.

```
nimsos.preparation_input
      (machine = "STAN",
      input_file = proposals_file,
      input_folder = input_folder)
```

The parameter of machine selects the module of robotic experiments. For machine, 'STAN' which is the standard module for this procedure and 'NAREE' (NIMS automated robotic electrochemical experiments) are used. For `input_file`, the experimental conditions selected by the AI are specified. In addition, the folder in the computer where the input files for robotic experiments are stored is referred to as the `input_folder`. In the `nimsos.preparation_input` module, the two functions `make_machine_file()` and `send_message_machine()` should be modified depending on the robotic systems used. The former creates the input files for robotic experiments from selected experimental conditions, whereas later sends the message to begin the robotic experiments.

4.2.4. Analysis of output files from experiments and update of candidates file

`nimsos.analysis_output` is a class used to analyze the experimental results and update the candidates file. For example, `nimsos.analysis_output` is used as follows.

```
nimsos.analysis_output
      (machine = "STAN",
      input_file = proposals_file,
      output_file = candidates_file,
      num_objectives = ObjectivesNum,
      output_folder = output_folder)
```

The parameter of machine is the same in `nimsos.preparation_input` module, which selects the module for robotic experiments. Here, 'STAN' and 'NAREE' can be selected. For `input_file`, the experimental conditions selected by the AI are specified, and `output_file` is the name of the candidates file. The file specified by `output_file` is updated by this module. In addition, for `num_objectives`, the number of objectives is input. For output_folder, the folder in the computer where the results from robotic experiments are output is specified. In the `nimsos.analysis_output` module, two functions `extract_objectives()` and `recieve_exit_message()` should be modified depending on the robot systems. The former extracts the values of objective functions from the output files of robotic experiments, and the later receives the message when the robotic experiments are finished. If 'NAREE' is selected, `objectives_info` should be specified as a dictionary indicating which objective function is extracted from the experimental results.

4.3. Visualization of the results

By using `nimsos.visualization`, the figures of the results are obtained. When this module is used, the new folder named fig is prepared in advance in the same folder where the main script is stored. The figures are output to this folder. `nimsos.visualization.plot_history` and `nimsos.visualization.plot_distribution.plot` create figures for the history and distributions of objective functions, respectively. These modules are useful when using AI algorithms other than PDC. In contrast, `nimsos.visualization.plot_phase_diagram.plot` creates the predicted phase diagram when PDC is used as an AI algorithm.

5. Usage of the NIMS-OS GUI version

A GUI version of NIMS-OS has been developed for easy execution, which is available at https://github.com/nimsos-dev/nimsos-gui. This can be used after installing the required Python version, as described in Section 4.1, and performing the installation is described in the manual (https://nimsos-dev.github.io/nimsos/docs/en/index.html). Figure 4 shows the operation screen of the NIMS-OS GUI version. In this GUI version, the name of the candidates file is fixed to candidates.csv, and the name of the proposals file is fixed to proposals.csv. The execution procedure is as follows:

(i) Specify the number of objectives, proposals, and cycles in the *Parameters* section.
(ii) Select the method to be used in the *AI algorithm* section. If we use a newly created module for AI method named ai_tool_original.py, click on Original.
(iii) Select the robotic system in the *Robotic system* section. If we use a newly created module for robotic systems named preparation_input_original.py and analysis_output_original.py, click on Original.
(iv) Press the '*Run*' button on the *Controller* section to begin the automated materials exploration.

When NIMS-OS is started, the *Cycle counter* begins to operate. Furthermore, in the *Time* section, the amount of time required to execute the AI algorithm and a single cycle are measured, and the remaining time is also output. The standard output

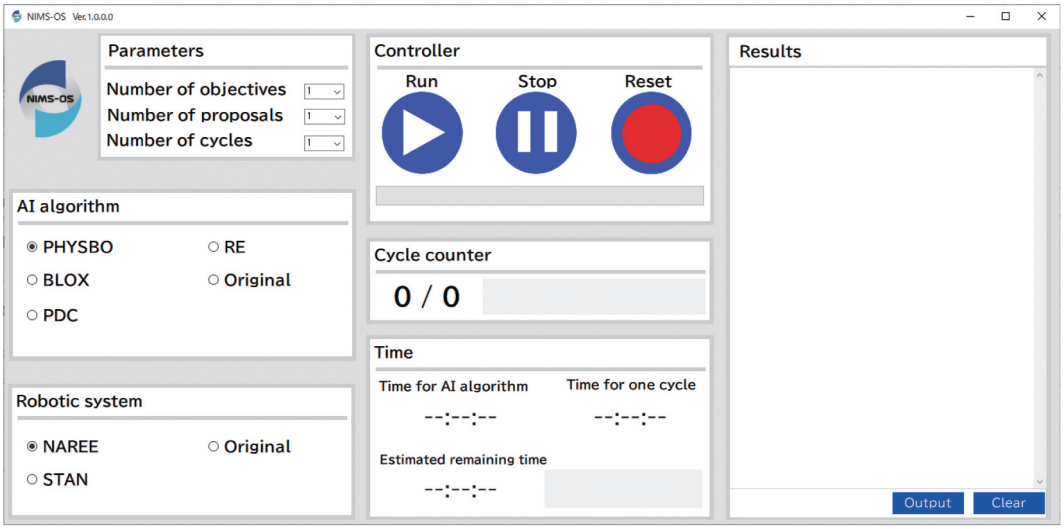

Figure 4. Operation screen of the NIMS-OS GUI version.

of the Python version is displayed in real time in the *Results* section, and these output results can be saved as a file by pressing the *Output* button. In addition, to pause the automated exploration, the user can press the '*Stop*' button of the *Controller* section. Note that pressing this button does not stop the process immediately, but when the candidates file is updated, NIMS-OS is stopped. To reset the settings, press the '*Reset*' button on the *Controller*. The operation with NAREE is shown as a video of Supplemental Movie 1. Note that even for processes that cannot be partially automated, the closed loop between AI and manual experiments can still be achieved by selecting STAN in the *Robotic system* section, as explained in Section 3.2.1.

6. Application

To demonstrate the effectiveness of NIMS-OS for the application of automated robotic experiments, we applied NIMS-OS for NAREE system and performed an exploration for multi-component electrolytes that maximize the performance of lithium metal electrode. The anode-free type microplate based electrochemical cells were fabricated using $LiFePO_4$ as positive electrode and Cu foil as negative electrode. The cells were subjected to charging process with capacity limitation of 0.05 mAh. After that, the cells were subjected to discharge process. Here, we defined the discharge time as one-dimensional objective function. In this case, the longer discharge time represents the better battery performance (higher capacity). Using such experimental setup, a combination of electrolyte additives was optimized to maximize the discharge time. Five different additives were selected from a list of 16 compounds (Table 1) and injected into electrochemical cell containing 1 M LiTFSI in TEGDME. In this case, the number of candidates for combination of electrolyte additives is ${}_{16}C_5 = 4{,}368$. The candidate files for this experiment were prepared in similar manner as shown in Figure 3 (combination of materials). In our experiment, 32 electrochemical cells were prepared in one microplate and 32 experiments were parallelly performed in 2 hours.

For the autonomous experiments for searching multi-component electrolytes using NAREE system operated by NIMS-OS, at first, 32 parallel experiments (one microplate) were performed by random exploration using RE because we do not have initial data at this stage. After obtaining the initial data by RE, next five cycles of experiments (five microplates) were performed by BO using PHYSBO. Notably, fully automated experiment was continuously conducted without any human intervention for 10 hours. After that, addition six cycles of experiments (six microplates) were also performed by BO. In total, 384 experiments were performed. The obtained results

Table 1. List of 16 types of additives used in an automated exploration for new electrolytes using the NAREE system. For all additives, the solvent is fixed as TEGDME.

ID	Additive	Concentration
1	lithium bis(pentafluoroethanesulfonyl)imide (LiBETI)	100 mM
2	$LiPF_6$	100 mM
3	$LiBF_4$	100 mM
4	lithium bis(trifluoro methanesulfonyl)imide (LiTFSI)	100 mM
5	LiTfO	100 mM
6	$LiClO_4$	100 mM
7	lithium bis(oxalate)borate (LiBOB)	10 mM
8	$LiAsF_6$	10 mM
9	LiF	10 mM
10	N-methyl-2-pyrrodione (NMP)	2 vol.%
11	sulfolane	2 vol.%
12	dimethyl sulfoxide (DMSO)	2 vol.%
13	propylene carbonate (PC)	2 vol.%
14	ethylene carbonate (EC)	2 vol.%
15	fluoroethylene carbonate (FEC)	2 vol.%
16	vinylene carbonate (VC)	2 vol.%

can be visualized by using nimsos.visualization in the Python version of NIMS-OS, as already mentioned in Section 4.3, and the time course of objective function and the histogram distribution of the results in the total 384 experiments were summarized in Figure 5. The results clearly revealed that the best electrolyte composition was discovered at 7th experimental cycle. In Table 2, the details of electrolyte composition for top 10 samples that enhanced the discharge time were summarized. The electrolyte containing, 100 mM $LiPF_6$, 100 mM LiTFSI, 2 vol.% PC, 2 vol.% FEC, and 2 vol.% VC, exhibits the highest discharge time of 1439.09 s. It should be noted that the possible maximum discharge time is 1800 s since the current density during discharge was set to 0.1 mA. Thus, there is still much room for improvement of battery performance. In addition, there can be seen that most of the top 10 samples contains VC and/or FEC. These results are essentially consistent with the knowledge in this filed that VC and FEC has positive effect for improving the performance of the lithium metal electrode [43,44].

7. Summary

In this study, we developed NIMS-OS to implement a closed loop of AI and robotic experiments for automated materials exploration. We anticipate that this software can serve as a generic control system. To use NIMS-OS, a candidates file listing experimental conditions as a materials search space should be prepared in advance. This allows various problems for automated materials exploration to be commonly performed in NIMS-OS. Establishing standards for automated materials exploration is a key advantages of such generic control software. Using NIMS-OS and the NAREE system, we also demonstrated an example of automatic exploration for electrolytes.

The compatibility with original robotic systems other than NAREE is discussed. We believe that the most crucial aspects of integrating other robotic systems lie in providing instructions to initiate the robot and determining the completion of the robotic experiment. Regarding the former, the current NAREE system is designed to automatically start an experiment when an input file is stored in a specified folder. Therefore, by making slight modifications to the existing Python script, original robotic systems with this functionality can be easily integrated into NIMS-OS. Even if the PC controlling the experimental system and the PC running the NIMS-OS are different, the robot can be started by sharing the specified folder using a file-sharing service or Network-Attached Storage (NAS). However, if the experimental systems require voltage signal control or Application Programming Interface (API) control, specific Python code needs to be developed. The development of Python code for voltage signal control or API control is considered a future prospect. Regarding the latter, the experimental results will always be output in the specified folder. Therefore, it is sufficient to determine whether the result files have been generated or not, even if the robotic systems are changed. Furthermore, there may be cases

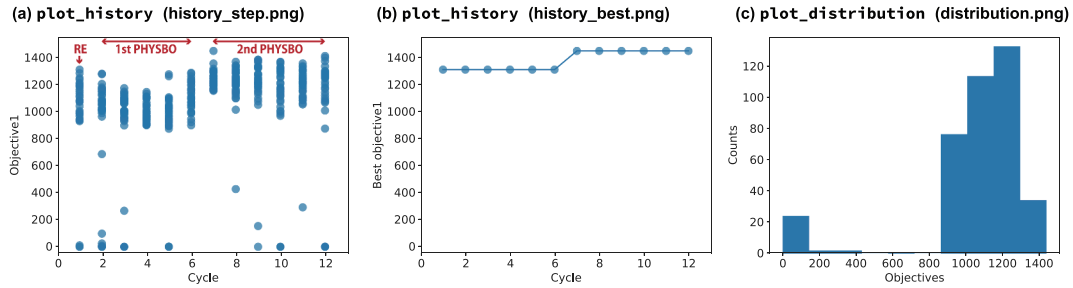

Figure 5. Output results from NIMS-OS for automated exploration for electrolytes using the NAREE system: (a) history_step.png and (b) history_best.png by nimsos.Visualization.plot_history and (c) distribution.png by nimsos.Visualization.plot_distribution.plot. The target property is the discharge time and its unit is seconds. In the first cycle, RE is used to generate initial states. After the second cycle, PHYSBO is used.

Table 2. Top 10 compositions that enhanced the discharge time. The found cycle number is also shown.

Ranking	Additive 1	Additive 2	Additive 3	Additive 4	Additive 5	Discharge time	Found cycle
1	100 mM $LiPF_6$	100 mM LiTFSI	2 vol.% PC	2 vol.% FEC	2 vol.% VC	1439.09 s	7th
2	100 mM LiBETI	100 mM LiTfO	10 mM LiBOB	2 vol.% EC	2 vol.% FEC	1401.97 s	12th
3	2 vol.% NMP	2 vol.% sulfolane	2 vol.% DMSO	2 vol.% PC	2 vol.% FEC	1374.86 s	9th
4	100 mM $LiBF_4$	100 mM LiTFSI	100 mM LiTfO	10 mM LiF	2 vol.% FEC	1365.57 s	9th
5	100 mM LiBETI	100 mM $LiBF_4$	10 mM LiBOB	2 vol.% PC	2 vol.% FEC	1364.32 s	12th
6	100 mM LiTFSI	100 mM LiTfO	10 mM $LiAsF_6$	2 vol.% FEC	2 vol.% VC	1358.99 s	10th
7	100 mM LiBETI	10 mM LiBOB	10 mM LiF	2 vol.% EC	2 vol.% VC	1357.43 s	10th
8	100 mM LiBETI	100 mM LiTFSI	10 mM LiBOB	10 mM LiF	2 vol.% FEC	1356.39 s	9th
9	100 mM $LiBF_4$	100 mM LiTFSI	100 mM $LiClO_4$	2 vol.% sulfolane	2 vol.% VC	1347.23 s	11th
10	100 mM $LiPF_6$	100 mM LiTfO	10 mM $LiAsF_6$	10 mM LiF	2 vol.% EC	1346.72 s	7th

where the experimental system can only be fully controlled by the GUI software. In such cases, it is necessary to manually press a button on the GUI control screen to initiate the robot. However, an automated closed loop can be achieved by installing Robotic Process Automation (RPA) on the PC that controls the experimental system [15]. For example, with RPA, the following operations can be performed: (i) Identifying the presence of an input file in the specified folder, (ii) Providing instructions to initiate the experiments on the GUI operation screen that controls the experimental system, and (iii) Deleting the input file once it is confirmed that the experimental results have been generated. Thus, we believe that the current NIMS-OS is designed to be easily adaptable to a variety of original robotic systems in materials science.

At present, NIMS-OS does not include a sufficient set of available AI algorithms and robotic experimental systems. For the further growth of this OS, developing and releasing more modules for various AI algorithms and robotic systems will be essential. The NAREE system used in this study can perform sequential operations since all evaluations of proposed experimental conditions by robotic experiments are completed within the same timeframe. However, in realistic experiments, the costs associated with synthesis, device fabrication, and evaluation strongly depend on the specific experimental conditions. In such cases, waiting for all experiments to be completed would be inefficient. To address this issue, BO introduces the concept of asynchronous parallel global optimization [45,46]. Therefore, it is necessary to develop a module within NIMS-OS that can facilitate asynchronous parallel optimization. Furthermore, in automated materials exploration, a greater amount of experimental data is generated compared to human experiments. Thus, the ability to store, share, and utilize experimental data for secondary purposes should be implemented as extensions in NIMS-OS. Specifically, a module that facilitates the automatic transfer of data to external storage or data repositories will be essential in enhancing data sharing and utilization. We will continue to enhance the extensions available for NIMS-OS to develop it as a game changer for digital transformation (DX) in materials science.

Acknowledgments

The authors thank Masahiko Demura, Hideki Yoshikawa, and Masanobu Naito for valuable discussions. The authors also thank Kazuha Nakamura for experimental contributions, and thank Satoshi Murata, Daisuke Ryuno, and Hiromichi Taketa for the development of NIMS-OS.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

The work was supported by the MEXT Program: Data Creation and Utilization-Type Material Research and Development Project [JPMXP1122712807].

References

[1] White A. The materials genome initiative: one year on. MRS Bull. 2012;37(8):715–716. doi: 10.1557/mrs.2012.194
[2] Ramprasad R, Batra R, Pilania G, et al. Machine learning in materials informatics: recent applications and prospects. npj Comput Mater. 2017;3(1):54. doi: 10.1038/s41524-017-0056-5
[3] Schmidt J, Marques MRG, Botti S, et al. Recent advances and applications of machine learning in solid-state materials science. npj Comput Mater. 2019;5(1):83. doi: 10.1038/s41524-019-0221-0
[4] Ling C. A review of the recent progress in battery informatics. npj Comput Mater. 2022;8(1):33. doi: 10.1038/s41524-022-00713-x
[5] Terayama K, Sumita M, Tamura R, et al. Black-box optimization for automated discovery. Acc Chem Res. 2021;54(6):1334–1346. doi: 10.1021/acs.accounts.0c00713
[6] Ueno T, Rhone TD, Hou Z, et al. COMBO: an efficient bayesian optimization library for materials science. Mater Discovery. 2016;4:18–21. doi: 10.1016/j.md.2016.04.001
[7] Packwood D. Bayesian optimization for materials science. Springer Singapore: 2017. doi: 10.1007/978-981-10-6781-5
[8] Kumar PV, Jin Y. Bayesian optimisation for efficient material discovery: a mini review. Nanoscale. 2023;15:10975–10984. doi: 10.1039/D2NR07147A
[9] Homma K, Liu Y, Sumita M, et al. Optimization of a heterogeneous ternary Li_3PO_4—Li_3BO_3—Li_2SO_4 mixture for Li-ion conductivity by machine learning. J Phys Chem C. 2020;124(24):12865–12870. doi: 10.1021/acs.jpcc.9b11654
[10] Sakurai A, Yada K, Simomura T, et al. Ultranarrow-band wavelength-selective thermal emission with aperiodic multilayered metamaterials designed by Bayesian optimization. ACS Central Sci. 2019;5(2):319–326. doi: 10.1021/acscentsci.8b00802
[11] Sun S, Tiihonen A, Oviedo F, et al. A data fusion approach to optimize compositional stability of halide perovskites. Matter. 2021;4(4):1305–1322. doi: 10.1016/j.matt.2021.01.008
[12] Tamura R, Osada T, Minagawa K, et al. Machine learning-driven optimization in powder manufacturing of Ni-Co based superalloy. Mater Design. 2021;198:109290. doi: 10.1016/j.matdes.2020.109290
[13] Matsuda S, Lambard G, Sodeyama K. Data-driven automated robotic experiments accelerate discovery of multi-component electrolyte for rechargeable Li-O_2 batteries. Cell Rep Physical Sci. 2022;3(4):100832. doi: 10.1016/j.xcrp.2022.100832
[14] Ozaki Y, Suzuki Y, Hawai T, et al. Automated crystal structure analysis based on blackbox optimisation.

npj Comput Mater. 2020;6(1):75. doi: 10.1038/s41524-020-0330-9
[15] Tamura R, Sumita M, Terayama K, et al. Automatic Rietveld refinement by robotic process automation with RIETAN-FP. Sci Technol Adv Mater Methods. 2022;2(1):435–444. doi: 10.1080/27660400.2022.2146470
[16] Chakraborti N. Genetic algorithms in materials design and processing. Int Mater Rev. 2004;49(3–4):246–260. doi: 10.1179/095066004225021909
[17] Patra TK, Meenakshisundaram V, Hung JH, et al. Neural-network-biased genetic algorithms for materials design: evolutionary algorithms that learn. ACS Comb Sci. 2017;19(2):96–107. doi: 10.1021/acscombsci.6b00136
[18] Dieb TM, Ju S, Yoshizoe K, et al. MDTS: automatic complex materials design using Monte Carlo tree search. Sci Technol Adv Mater. 2017;18(1):498–503. doi: 10.1080/14686996.2017.1344083
[19] Li J, Zhang J, Tamura R, et al. Self-learning entropic population annealing for interpretable materials design. Digital Discov. 2022;1:295–302. doi: 10.1039/D1DD00043H
[20] Kitai K, Guo J, Ju S, et al. Designing metamaterials with quantum annealing and factorization machines. Phys Rev Res. 2020;2(1):013319. doi: 10.1103/PhysRevResearch.2.013319
[21] Koshikawa AS, Ohzeki M, Kadowaki T, et al. Benchmark test of black-box optimization using D-Wave quantum annealer. J Phys Soc Jpn. 2021;90(6):064001. doi: 10.7566/JPSJ.90.064001
[22] Izawa S, Kitai K, Tanaka S, et al. Continuous black-box optimization with an Ising machine and random subspace coding. Phys Rev Res. 2022;4(2):023062. doi: 10.1103/PhysRevResearch.4.023062
[23] King RD, Rowland J, Oliver SG, et al. The automation of science. Science. 2009;324(5923):85–89. doi: 10.1126/science.1165620
[24] Boyd J. Robotic laboratory automation. Science. 2002;295(5554):517–518. doi: 10.1126/science.295.5554.517
[25] Olsen K. The first 110 years of laboratory automation: technologies, applications, and the creative scientist. SLAS Technol. 2012;17(6):469–480. doi: 10.1177/2211068212455631
[26] Macarron R, Banks MN, Bojanic D, et al. Impact of high-throughput screening in biomedical research. Nat Rev Drug Discov. 2011;10(3):188–195. doi: 10.1038/nrd3368
[27] MacLeod BP, Parlane FGL, Morrissey TD, et al. Self-driving laboratory for accelerated discovery of thin-film materials. Sci Adv. 2020;6(20):eaaz8867. doi: 10.1126/sciadv.aaz8867
[28] Shimizu R, Kobayashi S, Watanabe Y, et al. Autonomous materials synthesis by machine learning and robotics. APL Mater. 2020;8(11):111110. doi: 10.1063/5.0020370
[29] Dave A, Mitchell J, Kandasamy K, et al. Autonomous discovery of battery electrolytes with robotic experimentation and machine learning. Cell Rep Physical Sci. 2020;1(12):100264. doi: 10.1016/j.xcrp.2020.100264
[30] Burger B, Maffettone PM, Gusev VV, et al. A mobile robotic chemist. Nature. 2020;583(7815):237–241. doi: 10.1038/s41586-020-2442-2
[31] Roch LM, Häse F, Kreisbeck C, et al. ChemOS: an orchestration software to democratize autonomous discovery. PLoS One. 2020;15(4):1–18. doi: 10.1371/journal.pone.0229862
[32] Motoyama Y, Tamura R, Yoshimi K, et al. Bayesian optimization package: PHYSBO. Comput Phys Commun. 2022;278:108405. doi: 10.1016/j.cpc.2022.108405
[33] Terayama K, Sumita M, Tamura R, et al. Pushing property limits in materials discovery via boundless objective-free exploration. Chem Sci. 2020;11:5959–5968. doi: 10.1039/D0SC00982B
[34] Terayama K, Tamura R, Nose Y, et al. Efficient construction method for phase diagrams using uncertainty sampling. Phys Rev Mater. 2019;3:033802. doi: 10.1103/PhysRevMaterials.3.033802
[35] Matsuda S, Nishioka K, Nakanishi S. High-throughput combinatorial screening of multi-component electrolyte additives to improve the performance of Li metal secondary batteries. Sci Rep. 2019;9(1):6211. doi: 10.1038/s41598-019-42766-x
[36] Ong SP, Richards WD, Jain A, et al. Python materials genomics (pymatgen): a robust, open-source python library for materials analysis. Comput Mater Sci. 2013;68:314–319. doi: 10.1016/j.commatsci.2012.10.028
[37] Ward L, Agrawal A, Choudhary A, et al. A general-purpose machine learning framework for predicting properties of inorganic materials. npj Comput Mater. 2016;2(1):1–7. doi: 10.1038/npjcompumats.2016.28
[38] RDKit. https://www.rdkit.org/
[39] Ojih J, Al-Fahdi M, Rodriguez AD, et al. Efficiently searching extreme mechanical properties via boundless objective-free exploration and minimal first-principles calculations. npj Comput Mater. 2022;8(1):143. doi: 10.1038/s41524-022-00836-1
[40] Tamura R, Deffrennes G, Han K, et al. Machine-learning-based phase diagram construction for high-throughput batch experiments. Sci Technol Adv Mater. 2022;2(1):153–161. doi: 10.1080/27660400.2022.2076548
[41] Katsube R, Terayama K, Tamura R, et al. Experimental establishment of phase diagrams guided by uncertainty sampling: an application to the deposition of Zn–Sn–P films by molecular beam epitaxy. ACS Mater Lett. 2020;2(6):571–575. doi: 10.1021/acsmaterialslett.0c00104
[42] Hu WH, Chen TT, Tamura R, et al. Topological alternation from structurally adaptable to mechanically stable crosslinked polymer. Sci Technol Adv Mater. 2022;23(1):66–75. doi: 10.1080/14686996.2021.2025426
[43] Ota H, Shima K, Ue M, et al. Effect of vinylene carbonate as additive to electrolyte for lithium metal anode. Electrochimica Acta. 2004;49(4):565–572. doi: 10.1016/j.electacta.2003.09.010
[44] Zhang XQ, Cheng XB, Chen X, et al. Fluoroethylene carbonate additives to render uniform Li deposits in lithium metal batteries. Adv Funct Mater. 2017;27(10):1605989. doi: 10.1002/adfm.201605989
[45] Ginsbourger D, Janusevskis J, Le Riche R. Dealing with asynchronicity in parallel Gaussian process based global optimization. Mines Saint-Etienne. 2011;hal00507632.
[46] Janusevskis J, Le Riche R, Ginsbourger D. Expected improvements for the asynchronous parallel global optimization of expensive functions: potentials and challenges. In: Hamadi Y Schoenauer M, et al., editors. Learning and intelligent optimization. Berlin, Heidelberg: Springer; 2012. p. 413–418. doi: 10.1007/978-3-642-34413-8_37