Taylor & Francis
Taylor & Francis Group

Science and Technology of Advanced Materials: Methods

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

CrySPY: a crystal structure prediction tool accelerated by machine learning

Tomoki Yamashita, Shinichi Kanehira, Nobuya Sato, Hiori Kino, Kei Terayama, Hikaru Sawahata, Takumi Sato, Futoshi Utsuno, Koji Tsuda, Takashi Miyake & Tamio Oguchi

To cite this article: Tomoki Yamashita, Shinichi Kanehira, Nobuya Sato, Hiori Kino, Kei Terayama, Hikaru Sawahata, Takumi Sato, Futoshi Utsuno, Koji Tsuda, Takashi Miyake & Tamio Oguchi (2021) CrySPY: a crystal structure prediction tool accelerated by machine learning, Science and Technology of Advanced Materials: Methods, 1:1, 87-97, DOI: 10.1080/27660400.2021.1943171

To link to this article: https://doi.org/10.1080/27660400.2021.1943171

Published online: 13 Jul 2021.

SCIENCE AND TECHNOLOGY OF ADVANCED MATERIALS: METHODS
2021, VOL. 1, NO. 1, 87–97
https://doi.org/10.1080/27660400.2021.1943171

Taylor & Francis
Taylor & Francis Group

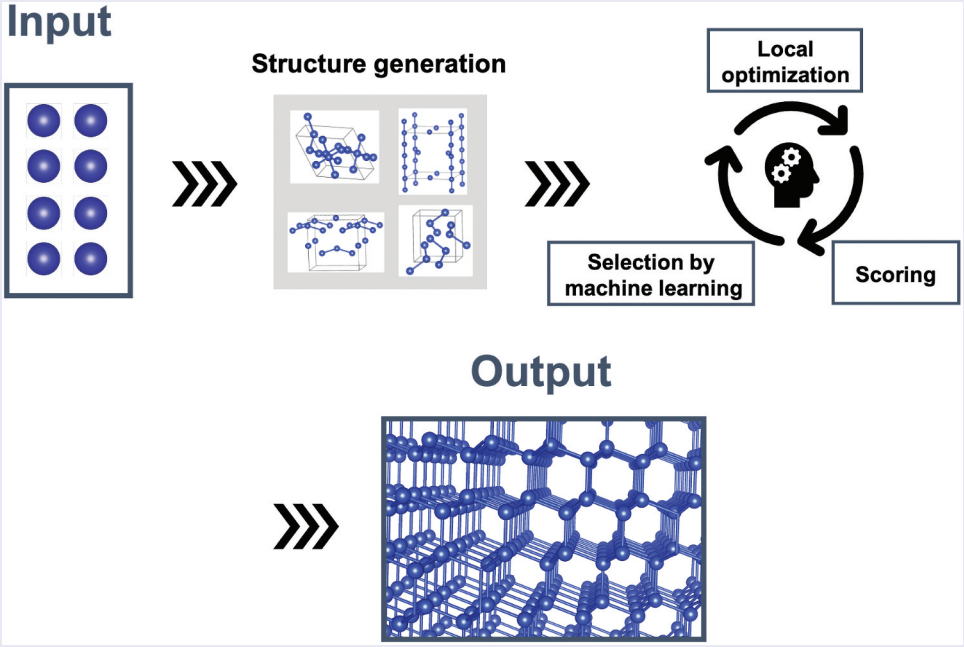
CrySPY: a crystal structure prediction tool accelerated by machine learning

Tomoki Yamashita[a,b], Shinichi Kanehira[b], Nobuya Sato[c], Hiori Kino[d,e], Kei Terayama[f], Hikaru Sawahata[g], Takumi Sato[a], Futoshi Utsuno[h], Koji Tsuda[d,i,j], Takashi Miyake[e] and Tamio Oguchi[b]

[a]Top Runner Incubation Center for Academia-Industry Fusion, Nagaoka University of Technology, Nagaoka, Japan; [b]Institute of Scientific and Industrial Research, Osaka University, Ibaraki, Japan; [c]Research Center for Computational Design of Advanced Functional Materials, National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan; [d]Research and Services Division of Materials Data and Integrated System, National Institute for Materials Science, Tsukuba, Japan; [e]Elements Strategy Initiative Center for Magnetic Materials, National Institute for Materials Science, Tsukuba, Japan; [f]Graduate School of Medical Life Science, Yokohama City University, Yokohama, Japan; [g]Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa, Japan; [h]Advanced Technology Research Laboratories, Idemitsu Kosan Co., Ltd., Sodegaura, Japan; [i]Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Japan; [j]Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan

ABSTRACT

We have developed an open-source software called CrySPY, which is a crystal structure prediction tool written in Python 3, and runs on Unix/Linux platforms. CrySPY enables anyone to easily perform crystal structure prediction simulations for materials discovery and design, and automates structure generation, structure optimization, energy evaluation, and efficiently selecting candidates using machine learning. Several searching algorithms are available such as random search, evolutionary algorithm, Bayesian optimization, and Look Ahead based on Quadratic Approximation. Machine learning is employed to efficiently select candidates for priority optimization. CrySPY does not require complex machine learning techniques for users. In the latest version of CrySPY, both atomic and molecular random structures can be generated. CrySPY supports VASP, QUANTUM ESPRESSO, OpenMX, soiap, and LAMMPS for local structure optimization and energy evaluation. CrySPY is distributed under the MIT license at https://github.com/Tomoki-YAMASHITA/CrySPY. Documentation of CrySPY is also available at https://Tomoki-YAMASHITA.github.io/CrySPY_doc.



KEYWORDS

Crystal structure prediction; Bayesian optimization; LAQA; evolutionary algorithm; first-principles calculations; machine learning; materials informatics

CLASSIFICATION

Materials data analysis

1. Introduction

Research on data-driven materials development based on computer simulations has been actively conducted for the last decades. With the development of computational capabilities, it has become possible to obtain a large amount of reliable data from first-principles calculations at high speed. However, first-principles calculations cannot be directly applied to the design of new materials for new compositions and unknown structures, because such

CONTACT Tomoki Yamashita yamashita06@vos.nagaokaut.ac.jp Top Runner Incubation Center for Academia-Industry Fusion, Nagaoka University of Technology, Nagaoka, Japan

quantum-mechanical approaches require their crystal structures as input. In recent years, crystal structure prediction (CSP) methods have made it possible to discover new materials in conjunction with first-principles calculations. The history of new material discovery by CSP is well summarized in the review paper by the pioneers in this field [1]. So far, a great deal of effort has been devoted to developing the searching algorithms, such as random search (RS) [2–5], simulated annealing [6,7], minima hopping [8,9], evolutionary algorithm (EA) [10–13], and particle swarm optimization (PSO) [14,15]. In particular, EA and PSO are widely used and efficient algorithms as implemented in USPEX [11–13] and CALYPSO [14,15], respectively.

Although significant progress in searching methodologies has been made over the years, CSP is quite a time-consuming task. In CSP, first-principles methods are mostly used for the structure relaxation and energy evaluation to find the global energy minimum. However, first-principles calculations are quite heavy for large systems. The development of more efficient searching algorithms is highly desired to reduce computational cost. In our previous study, we proposed two selection-type algorithms, Bayesian optimization (BO) [16,17] and Look Ahead based on Quadratic Approximation (LAQA) [18]. In the conventional methods, all generated structures are locally optimized in order, while the selection-type algorithms employ machine learning to efficiently select candidates for priority optimization. These machine learning approaches make it possible to find the most stable structure among a large number of structures with a small number of trials. Besides, it is worth mentioning that researches on the acceleration of local structure optimization have also been conducted using machine learning potential [19–23].

We have developed an open-source software called CrySPY, which enables anyone to easily perform CSP simulations for materials discovery and design. CrySPY automates structure generation, structure optimization and energy evaluation by external programs, and efficiently selecting candidates using machine learning. While some codes for CSP such as AIRSS [2–5], USPEX [11–13], CALYPSO [14,15], GASP [24], and XtalOpt [25] are released, CrySPY is unique in its feature of selecting candidates using machine learning. In Section 2, we provide an overview of the software. In Section 3, we elaborate on each searching algorithm. Some features of CrySPY are introduced in Section 4. Finally, we summarize our software in Section 5.

2. Overview

2.1. System requirement

CrySPY is written in the Python 3 programming language, and runs on Unix/Linux platforms including macOS. CrySPY requires the following external Python libraries:

- COMBO
- pymatgen
- PyXtal

COMBO [26] is an efficient Bayesian optimization library, pymatgen [27] has many features in materials analysis, and PyXtal [28] is useful for crystal structure generation and symmetry analysis. First-principles or classical interatomic potential codes are also needed for energy evaluation and local structure optimization. CrySPY is currently interfaced with the following program packages for such purposes:

- VASP
- QUANTUM ESPRESSO
- OpenMX
- soiap
- LAMMPS

VASP [29], QUANTUM ESPRESSO [30], and OpenMX [31] are first-principles program packages, and soiap [32] and LAMMPS [33] are classical molecular dynamics packages.

2.2. Simple execution

CrySPY is simple to run. CrySPY itself is available at https://github.com/Tomoki-YAMASHITA/CrySPY. Users only need to provide the input file of CrySPY, `cryspy.in`, and input files required for the structure optimizer. An example of `cryspy.in` is shown below:

```
[basic]
algo = RS
calc_code = VASP
tot_struc = 10
nstage = 2
njob = 5
jobcmd = qsub
jobfile = job_cryspy

[structure]
natot = 8
atype = Si
nat = 8

[VASP]
kppvol = 80 100
```

If VASP code is used, for example, `INCAR` and `POTCAR` files are needed as input files. Structure data and *k*-point input files such as `POSCAR` and `KPOINTS` are automatically generated by `CrySPY` with pymatgen. A job script to submit structure optimization jobs according to the system is also needed. Once the input files are prepared, the CSP simulation can be easily started by executing the following command:

```
$ python3 /path/to/cryspy.py
```

Table 1. Example of the result by CrySPY.

	Select	Spg_num	Spg_sym	Spg_num_opt	Spg_sym_opt	E_eV_atom	Magmom	Opt
251	1	107	I4mm	0	None	NaN	NaN	skip
14	1	175	P6/m	191	P6/mmm	−6.10057	27.5053	done
273	1	44	Imm2	44	Imm2	−6.54106	26.9209	not_yet
107	1	195	P23	200	Pm-3	−6.49809	26.6771	done
190	1	38	Amm2	38	Amm2	−6.68704	25.6406	done
111	2	71	Immm	71	Immm	−6.50534	26.2337	done
59	2	16	P222	16	P222	−6.47951	26.1596	done
288	2	12	C2/m	12	C2/m	−6.79212	25.1641	done
133	2	71	Immm	71	Immm	−6.56943	25.576	done
262	2	156	P3m1	0	None	NaN	NaN	skip
93	3	150	P321	0	None	NaN	NaN	skip
294	3	155	R32	155	R32	−6.28959	26.4513	not_yet
172	3	177	P622	191	P6/mmm	−5.96608	28.2994	done
145	3	177	P622	177	P622	−6.60823	24.9556	done
176	3	115	P-4 m2	115	P-4 m2	−6.56049	26.7792	done

Here, replace `/path/to/cryspy.py` with appropriate path of the main script, `cryspy.py`, for each user's system. When `cryspy.py` is executed, structure generation or job management is automatically performed, and the program will pause. They do not need to keep the program running.

2.3. Job automation

Structure optimization jobs are managed automatically by CrySPY. The number of the jobs to be submitted at a time is controlled by the input variable, `njob`, in `cryspy.in`. We adopt a stage-based system for structure optimization calculations. As an example of this, users can configure the following settings. In the first stage, only the ionic positions are relaxed, fixing the cell shape. Next, the ionic positions and cell shape are fully relaxed in the second stage. When a job is finished, the optimized structure data and energy can be automatically collected and a new job can be submitted.

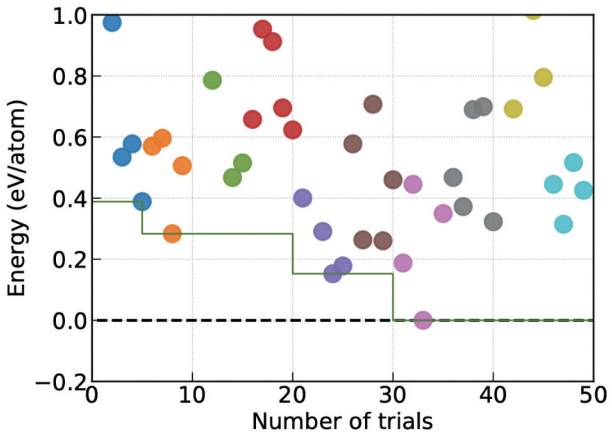


Figure 1. Example of CSP result by BO. Energies of optimized Y_2Co_{17} structures are shown. Each selection by BO is color-coded.

2.4. Analysis and visualization

CSP simulation data are serialized and saved using the pickle module of Python. The data can be easily analyzed using Python codes, and jupyter notebook templates for this purpose are available at https://github.com/Tomoki-YAMASHITA/CrySPY_utility. Figure 1 is an example of CSP results drawn using the template. The energies of optimized structures of Y_2Co_{17} obtained by CSP with BO are shown here. Each selection (five structure candidates) by machine learning is visualized in different colors. The result summary is also output as a text file, including the structure indices, the selection number, space group of the initial and optimized structures, energies, and so on, for quick checking as listed in Table 1.

In addition, initial and optimized structure data are written as text files in POSCAR and CIF format, so that users can see the structures using a crystal structure visualization tool such as VESTA [34].

3. Searching algorithms

The following searching algorithms are implemented in CrySPY:

- RS
- EA
- BO
- LAQA

RS relies on random structure generation and local structure optimization. Random structure generation is fundamental and used in other searching algorithms. EA puts emphasis on effective structure generation modeled after the theory of evolution, while BO and LAQA are the selection-type algorithms that can efficiently select potential candidates from a large number of structures by machine learning, reducing the number of searching trials. The key point here is that the selection-type algorithm can be used in combination with the structure generation methods by RS and EA.

3.1. Random search

A specified number of structures are generated randomly at the beginning, and local structure optimization is performed one by one. CrySPY has two modes of random structure generation: atomic crystal structure generation and molecular crystal structure generation. The latest version of CrySPY employs PyXtal library [28] as default to generate such random structures.

3.1.1. Atomic crystal structure generation

CrySPY can generate crystal structures with specific symmetry and chemical compositions using PyXtal. Optionally, it is also possible to generate the crystal structures using find_wy program [35] that we have been using. As default, a space group is randomly selected and the lattice parameters are automatically set to appropriate values. The volume of the unit cell can be specified within an arbitrary range in CrySPY.

Non-symmetric random structures can also be generated using a built-in function of CrySPY for low-symmetry CSP. In this case, a crystal system (not a space group) and lattice parameters are randomly selected within a physically appropriate range, and a specified number of atoms are arranged in the cell at random.

3.1.2. Molecular crystal structure generation

Interfacing with PyXtal enables symmetric molecular crystal generation. When the molecular geometry is given, random crystal structures with molecules occupying both general and special Wyckoff positions according to a specified space group can be generated. The following is an example of the structure section of the input file, `cryspy.in`, for two formula units of Li_3PS_4:

```
[structure]
struc_mode = mol
natot = 16
atype = Li P S
nat = 6 2 8
mol_file = ./Li.xyz ./PS4.xyz
nmol = 6 2
```

Here, Li and PS_4 units are defined in `./Li.xyz` and `./PS4.xyz` files, respectively. A single atom is treated as a molecule in the molecular crystal structure generation mode. In this example, a random molecular crystal structure is composed of six Li molecules (atoms) and two PS_4 molecules as specified in the input variable, `nmol`.

We generated 50 random molecular crystal structures using the above settings to validate whether a stable structures can be obtained. Total energy calculations and local structure optimizations were carried out using the density functional theory (DFT) with the projector-augmented wave (PAW) method [36] with the VASP code [29]. The generalized gradient approximation (GGA) by Perdew, Burke, and Ernzerhof [37] was employed for exchange-correlation functional. A cutoff energy of 625 eV for the plane-wave expansion of the wave function and k-point mesh density of 80 $Å^{-3}$ were used. The atomic coordinates and cell parameters were fully optimized until forces acting on every atom became at least smaller than 0.01 eV/Å. Figure 2 shows the most and second most stable structures predicted by our CSP simulation. Space group of the most and second most stable structures are $P\bar{4}3m$ and $Pmn2_1$, respectively. The total energy difference between them is only 10 meV/atom. Unfortunately, the former ($P\bar{4}3m$) was not observed in the experiment, while the latter ($Pmn2_1$) was obtained and is known as a stable structure, γ-Li_3PS_4 [38]. This can happen because the calculated energy difference is much smaller than the chemical accuracy (1 kcal/mol = 43 meV/atom). We might evaluate the total energy of the former structure a little too low with the DFT calculation. The predicted and experimental crystal parameters of γ-Li_3PS_4 were listed in Table 2. It is an almost perfect match except for the tiny amount of differences in lattice constants and atomic positions. These results demonstrate that the symmetric molecular structure generation method is quite useful for stable structure predictions of molecular crystals.

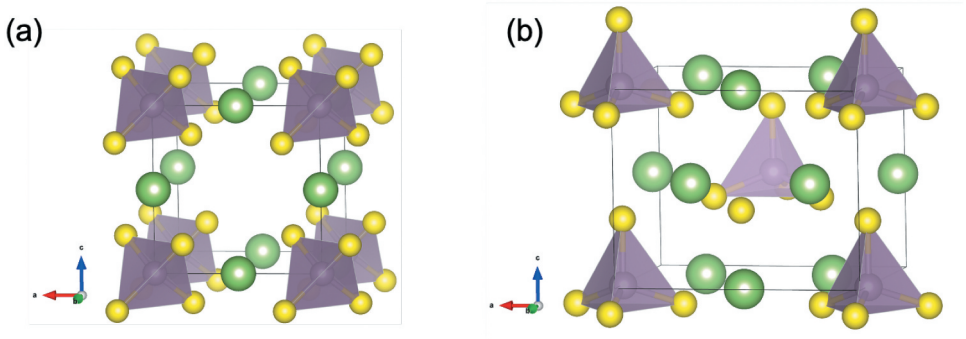


Figure 2. Two stable structures of Li_3PS_4 predicted by a CSP simulation drawn by VESTA [34]. (a) Space group $P\bar{4}3m$ and (b) space group $Pmn2_1$. Green, purple, and yellow spheres represent Li, P, and S atoms, respectively. The structure of $Pmn2_1$ corresponds to γ-Li_3PS_4 that is known as the most stable structure [38].

Table 2. Crystal parameters of predicted structure, and experimental ones [38] for γ-Li_3PS_4.

Predicted				
Space group $Pmn2_1$ (31) a = 7.78045; b = 6.61350; c = 6.21194				
Atom	Site	x	y	z
Li	4*b*	0.2432	0.3151	0.9970
Li	2*a*	0	0.1450	0.4853
P	2*a*	0	0.8173	0.9956
S	4*b*	0.2178	0.6722	0.8867
S	2*a*	0	0.1108	0.8849
S	2*a*	0	0.8090	0.3276
Experiment [38]				
Space group $Pmn2_1$ (31) a = 7.70829; b = 6.53521; c = 6.1365				
Atom	Site	x	y	z
Li	4*b*	0.2502	0.3309	0.0179
Li	2*a*	0	0.145	0.486
P	2*a*	0	0.8218	0.9942
S	4*b*	0.2185	0.6717	0.8857
S	2*a*	0	0.1083	0.888
S	2*a*	0	0.8049	0.3230

Even in the molecular crystal structure generation mode, CrySPY can specify the volume of the unit cell within an arbitrary range, keeping the molecular geometry. The feature to generate non-symmetric molecular crystal structures is also implemented in CrySPY for low-symmetry CSP. After randomly selecting a space group, the molecule is placed at a Wyckoff position and rotated randomly without considering the space group, leading to breaking the symmetry.

3.1.3. Constraint of minimum interatomic distance

In generating random structures, CrySPY can restrict the interatomic distance so that it is not too close. Here is an example of the input file to limit minimum interatomic distance for a Y-Co system:

```
[structure]
mindist_1 = 2.0 1.8
mindist_2 = 1.8 1.5
```

This means that minimum interatomic distances of Y-Y, Y-Co, and Co-Co are limited to 2.0, 1.8, and 1.5 Å, respectively. Structures with interatomic distances shorter than these values are automatically eliminated.

3.1.4. Constraint of space group

Users can limit space group. As an example, generated structures are limited to cubic systems with the following setting:

```
[structure]
spgnum = 195-230
```

3.2. Evolutionary algorithm

EA can be more effective than RS in generating structure. We implemented EA in CrySPY, referring to USPEX [11–13] and XtalOpt [25]. EA is based on that a population of structures is evolved, driven by survival of the fittest. Some lower-energy structures can become parents of a new generation of structures. CrySPY employs tournament selection and roulette selection for selecting parents from a population of surviving parents. Evolutionary operations for producing offspring from parents are of key importance. Crossover, permutation, and strain operations are available in CrySPY. The new generation includes not only structures produced by these evolutionary operations but also random structures and structures taken over by elite selection. In the elite selection, a part of the surviving parents are passed on to the next generation without mutation to keep the current best structures. An example of the EA section of the input file for a binary system is following:

```
[EA]
n_pop = 20
n_crsov = 10
n_perm = 3
n_strain = 5
n_rand = 2
n_elite = 2
n_fittest = 10
slct_func = TNM
t_size = 2
maxgen_ea = 5
mindist_ea_1 = 2.0 1.8
mindist_ea_2 = 1.8 1.5
```

Minimum interatomic distances can be also limited in producing offspring.

3.2.1. Parents selection

The number of surviving structures is specified in the input variable, `n_fittest`. The specified number of lower-energy structures excluding duplicates can be candidates to produce offspring. CrySPY has two methods to select the candidates. One is the tournament selection. A specified number of candidates (`t_size`) are randomly selected, and the candidate with the lowest energy among them is chosen as the parent. The other is the roulette selection. In this method, the probability of being selected as a parent is determined as:

$$p_i = \frac{f_i'}{\sum_k f_i'}, \tag{1}$$

where p_i and f_i' are the probability and linear-scaled fitness of ith candidate, respectively. The linear scaling of fitness is done by:

$$f_i' = \frac{a-b}{f_{\max}-f_{\min}} f_i + \frac{b f_{\max} - a f_{\min}}{f_{\max}-f_{\min}}, \tag{2}$$

where fitness f_i represents sign-reversed energy of ith candidate, a and b are parameters that define the range

of f_i', $f_{\max}$ and $f_{\min}$ are maximum and minimum values of fitness, respectively. For both methods, the probability of being selected as a parent is higher for candidates with lower energy.

3.2.2. Crossover

Figure 3 depicts a schematic crossover operation. Selected two parents are sliced near the center in a lattice vector after randomly translating their origin. Two offspring are produced by swapping the sliced parents, each carrying some genetic information. The number of atoms in the offspring is not necessarily the same as that of the parents at this time. Only a structure with a larger number of atoms is adopted. The number of atoms in the adopted structure is adjusted by removing or adding atoms near the border. Finally, the interatomic distances are checked.

3.2.3. Mutation

In permutation operation, two atoms of different elements are swapped in a selected parent. The number of times for permutation can be specified in the input file. Strain operation is also used as mutation. Lattice vectors a of a selected parent are transformed to a' by applying a strain matrix:

$$a' = \begin{pmatrix} 1+\eta_1 & \frac{1}{2}\eta_6 & \frac{1}{2}\eta_5 \\ \frac{1}{2}\eta_6 & 1+\eta_2 & \frac{1}{2}\eta_4 \\ \frac{1}{2}\eta_5 & \frac{1}{2}\eta_4 & 1+\eta_3 \end{pmatrix} a, \qquad (3)$$

where η_i are given by normal distribution with a mean of zero and a standard deviation.

3.3. Bayesian optimization

Our BO algorithm has been implemented in CrySPY. CSP using BO has been successfully applied to the known systems such as NaCl and Y_2Co_{17} [16]. The results have demonstrated that BO can significantly reduce the number of searching trials required to find the global minimum structure by 30–40% in comparison with pure RS, leading to much less computational cost. Although the candidate structures were created by random structure generation in the above study, it is possible to select from candidate structures generated by other methods such as EA.

BO is a well-established technique for black-box optimizations [39–41]. Figure 4 shows a CSP simulation procedure by BO, which is automated in CrySPY. First, a specified number of initial structures are randomly generated. We have to calculate the structure descriptors of the initial structures to learn the data and predict promising candidates. In the first selection, a specified number of structures are randomly selected and locally optimized to prepare the first training data. The descriptors for the optimized structures are updated. Then, the next candidates are selected by BO, and this sequence of the procedure is repeated. The below is an example of the BO section of the input file, `cryspy.in`:

```
[BO]
nselect_bo = 5
dscrpt = FP
score = TS
fp_rmin = 0.5
fp_rmax = 5.0
fp_npoints = 20
fp_sigma = 1.0
max_select_bo = 20
```

Here, one can specify the number of structures to be selected at a time, acquisition function, and parameters for calculating the structure descriptors.

3.3.1. Structure descriptor

To utilize machine learning, we have to convert crystal structures into vector data, called structure descriptor. Structure descriptors should measure the similarity of crystal structures and correlation between energy and structure. Although various structure descriptors have been developed recently [19,42–47], CrySPY currently supports only the F-fingerprint of Oganov and Valle [48] expressed as

$$F_{AB}(R) = \sum_{A_i,\text{cell}} \sum_{B_j} \frac{\delta(R - R_{ij})}{4\pi R_{ij}^2 \frac{N_A N_B}{V} \Delta} - 1. \qquad (4)$$

$F_{AB}(R)$ is a function of distance R, where A and B are chemical elements, R_{ij} is the interatomic distance between atoms i and j, N_A and N_B are the number of A and B atoms, and V is the unit cell volume. $\delta(R - R_{ij})$ is a Gaussian-smeared delta function, and the double sum runs over all atoms i within the unit cell and all j within the distance between minimum and maximum cutoff distance. The F-fingerprint is discretized over bins of width Δ to be treated as a vector.

The parameters for structure descriptors would affect the CSP performance. A common problem in BO is the need to predetermine the parameters. We

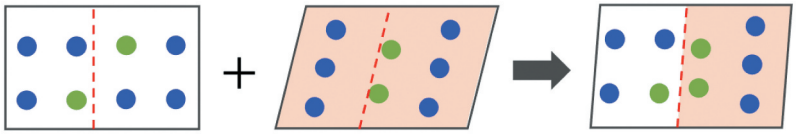


Figure 3. Crossover operation. As the offspring has one extra atom, an atom near the border (red broken line) is removed to adjust the number of atoms.

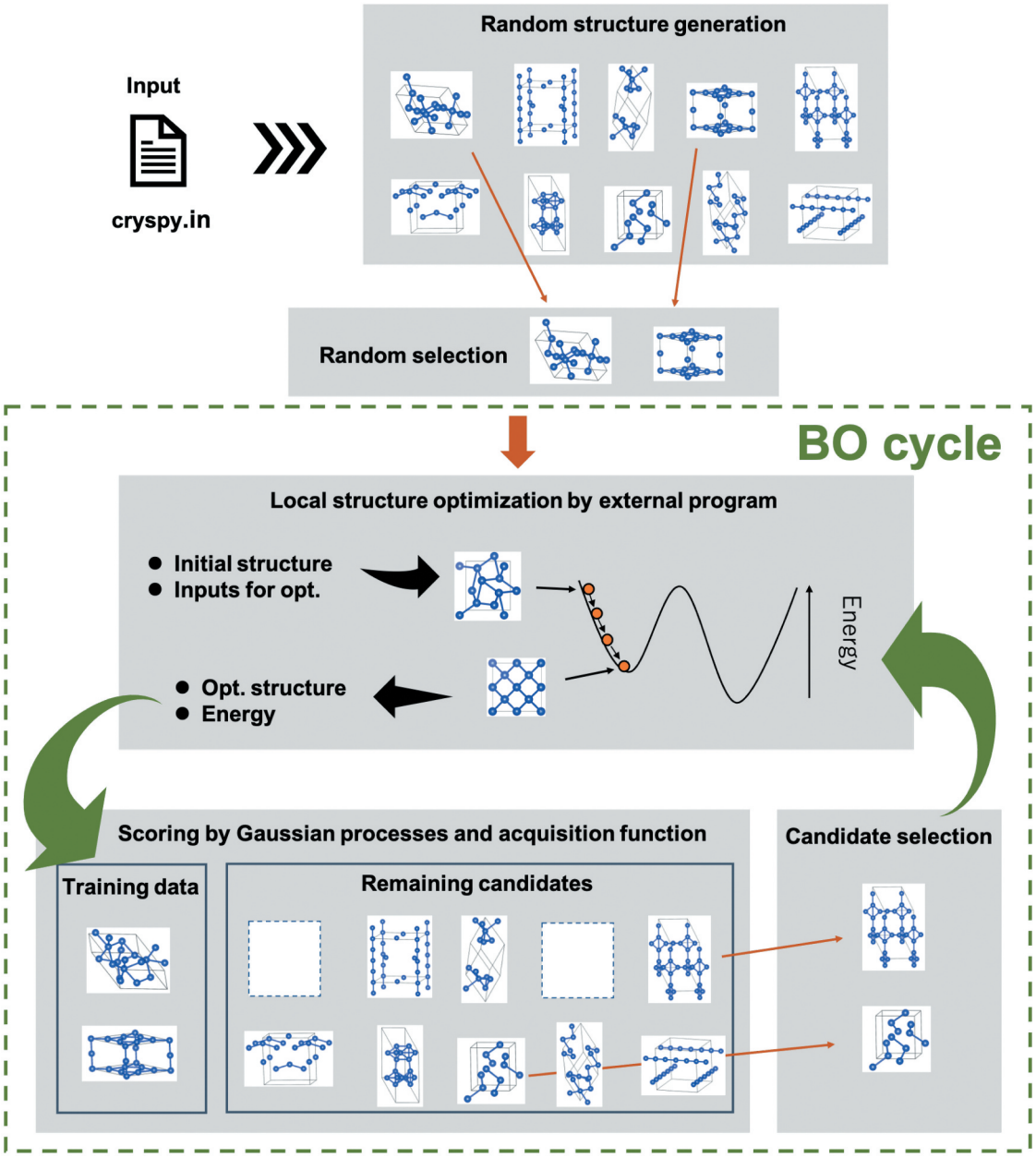


Figure 4. Simulation procedure in BO, which is automated in CrySPY.

also proposed an information measure of the distribution to estimate an appropriate parameter value, which can be predetermined [17]. CrySPY will be further developed to include other structure descriptors such as atom-centered symmetry functions [19,42,43], Smooth Overlap of Atomic Positions (SOAP) [44], orbital-field matrix (OFM) [45], and crystal graph [46]. Although various descriptors have been developed, selecting promising candidates in multi-specie systems becomes more difficult as the dimensions of the descriptors increase and become more complex. In general, BO has a problem with searching in high-dimensional space [49,50]. We cannot tell at this time whether BO can work efficiently in ternary or quaternary systems. To tackle this problem, investing the searching efficiency in multi-specie systems with the same atomic configuration but different number of atomic species such as diamond, zinc blende, and chalcopyrite structures will be a key task for us. Moreover, we will investigate the efficiency of BO when dimensionality reduction such as principal component analysis is applied to complicated descriptors for future work.

3.3.2. COMBO in CrySPY

CrySPY is interfaced with COMBO [26], which is a Python library designed for BO that employs Thompson sampling (TS) [41], random feature maps [51], and automatic hyperparameter tuning. CrySPY does not require complex machine learning techniques for users. A Gaussian process [52] is commonly used in BO to predict a true function. The uncertainty of the predicted function is also evaluated as predictive variance. The next candidates are selected based on the

acquisition function computed from the predictive value and variance. The following acquisition functions are available in CrySPY: probability of improvement (PI) [53], expected improvement (EI) [39], and TS. BO enables us to balance the tradeoff between exploration and exploitation of the search space. The candidates with lower predicted values and large variance are preferentially selected. CrySPY has an additional feature to output the predictive value and variance as a text file during CSP simulations.

A random feature map can be used instead of a Gaussian process to speed up the computation. It allows us to approximate a Gaussian process with a Bayesian linear model. The number of features in the random feature map, which determines the accuracy and computational complexity, is specified in the input file as:

```
[BO]
num_rand_basis = 2000
```

3.4. LAQA

As mentioned above, BO requires a structure descriptor, and the searching efficiency strongly depends on the structure descriptor. For non-experts, there are difficulties such as how to predetermine the parameters for the structure descriptor. Besides, there are many cases where the selected candidate with a high-energy structure does not provide important information for learning, and the local structure optimization of that candidate takes inefficient time. To simplify and improve the efficiency, another selection-type algorithm, LAQA, has been developed [18]. LAQA allows us to find the most stable structure with a minimum number of total local optimization steps. In our previous study, the computational cost can be significantly reduced by 50–90% compared to RS [18]. In the case of LAQA as well as BO, the candidate structures can be generated by any methods such as EA.

Figure 5 illustrates a CSP simulation procedure by LAQA, which is automated in CrySPY. Unlike conventional CSP methods, in LAQA, local structure optimization of the selected candidates is not performed all at once. We proceed only a few optimization steps for the selected candidates, and pause the simulation. LAQA can roughly estimate the final energy during local optimization, and the total computational cost can be reduced by controlling the local optimization step based on that estimated energy. A rough estimate of the final energy is used as the score L, and predicted using a quadratic approximation from the current energy and force acting each atom as:

$$L = -E + c\frac{F^2}{2\Delta F}, \tag{5}$$

where E is the energy per atom, F is the averaged norm of the atomic force, ΔF is the force difference from the previous step, and c is the bias parameter. The next candidates are selected based on the score L and preferentially optimized, thereby enabling an efficient search for the global minimum without unnecessary optimization calculations. The below is an example of the LAQA section of the input file, `cryspy.in`:

```
[LAQA]
nselect_laqa = 5
```

The input for LAQA is quite simple. One needs to specify the number of candidates to be selected at a time. Optionally, the bias parameter c can be specified. It is important to note that LAQA does not require any structure descriptors. One do not have to worry about the descriptors getting complicated in large and multi-specie systems.

4. Other features

4.1. Loading initial structures

User-defined initial structures can be used instead of random structures in CrySPY. The initial structure data should be put in `./data/pkl_data/init_struc_data.pkl` (relative path from the input file) before starting the CSP simulation. The loading flag also should be turned on in the input file as:

```
[option]
load_struc_flag = True
```

4.2. Appending structures

Initial structures can be appended at any time during the CSP simulation, except when using EA. By simply increasing the value of the input variable, `tot_struc`, in the basic section, CrySPY can append random structures to the initial structure data. Furthermore, CrySPY has a feature to generate and append structures using EA from already optimized structures and their energy data by turning on the flag as:

```
[option]
append_struc_ea = True
```

The input variables for EA as listed in Section 3.2 are also needed. This allows us to use the hybrid algorithms of EA and BO, or EA and LAQA, which could be more efficient. We will investigate searching efficiency of the hybrid algorithms in several materials for future work.

4.3. Skip and recalculation

Sometimes first-principles calculations do not work and give strange energies with wrong electronic structures. In such a case, CrySPY can skip the optimization and energy evaluation by manually editing the file, `./work/000016/stat_job` (relative path from the input file as an example of structure ID 16). If

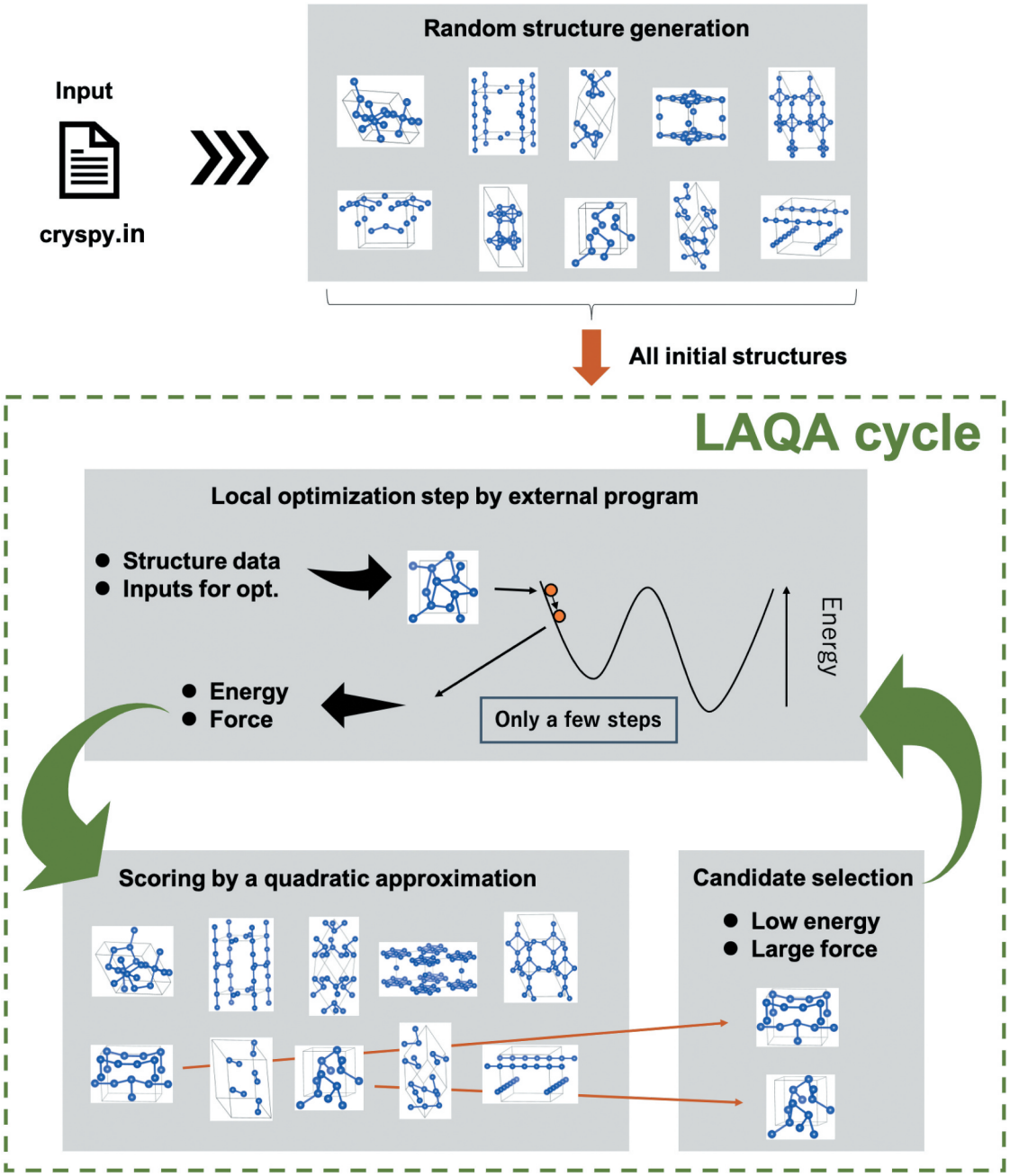


Figure 5. Simulation procedure in LAQA, which is automated in CrySPY.

necessary, CrySPY can recalculate the structure by specifying the structure ID in the input file as:

```
[option]
recalc = 16
```

4.4. Step data

Optionally, CrySPY can save the data in the process of local structure optimization for the structure, energy, force, and stress, by turning on the corresponding flag in the input file as:

```
[option]
energy_step_flag = True
struc_step_flag = True
force_step_flag = True
stress_step_flag = True
```

These can be used to generate a database for various applications such as machine learning potential generation.

5. Conclusion

We introduced a CSP tool CrySPY written in Python 3, which runs on Unix/Linux platforms. This open-source code allows us to automate structure generation, structure optimization, energy evaluation, and efficiently selecting candidates using machine learning. The searching algorithms available in CrySPY are highlighted. In particular, the

selection-type algorithms can efficiently select candidates for priority optimization, reducing the computational cost significantly. Non-experts can conduct CSP simulations using CrySPY without the need for special skills. CrySPY is interfaced with several structure optimizers such as VASP, QUANTUM ESPRESSO, OpenMX, soiap, and LAMMPS. The current version of the software is 0.9.1 at the time of writing. CrySPY will be further developed to include more structure descriptors and new features. The open-source code is developed in github. The documentation and utility of CrySPY are also available. We hope our open-source software will help to standardize research with CSP for materials discovery and design.

Acknowledgements

We are grateful to T. Ishikawa for useful discussion on EA. We thank Z. Hou for providing the code to calculate structure descriptors. Part of the computation in this work was conducted using the facilities of the Supercomputer Center, the Institute for Solid State Physics, and the Fujitsu PRIMERGY CX400M1/CX2550M5 (Oakbridge-CX) in the Information Technology Center, The University of Tokyo. It was also partly carried out by Research Institute for Information Technology, Kyushu University.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by Materials research by Information Integration Initiative (MI^2I) project of the Support Program for Starting Up Innovation Hub, and Core Research for Evolutional Science and Technology (CREST) [Grant number JPMJCR1502] from Japan Science and Technology Agency (JST). It was also supported by Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan as a social and scientific priority issue (Creation of new functional devices and high-performance materials to support next-generation industries; CDMSI) and (Building Innovative Drug Discovery Infrastructure Through Functional Control of Biomolecular Systems) to be tackled by using a post-K computer. It was supported by JSPS KAKENHI Grant Number JP18K13474 and JP20J13011. This work was also supported by MEXT Leading Initiative for Excellent Young Researchers.

References

[1] Oganov AR, Pickard CJ, Zhu Q, et al Structure prediction drives materials discovery. Nat Rev Mater. 2019;4:331–348.
[2] Pickard CJ, Needs RJ. High-pressure phases of silane. Phys Rev Lett. 2006;97:045504.
[3] Pickard CJ, Needs RJ. Structure of phase iii of solid hydrogen. Nat Phys. 2007;3(7):473–476.
[4] Pickard CJ, Needs RJ. Ab initio random structure searching. J Phys: Condens Matter. 2011;23(5):053201.
[5] Needs RJ, Pickard CJ. Perspective: role of structure prediction in materials discovery and design. APL Mater. 2016;4(5):053210.
[6] Kirkpatrick S, Gelatt JCD, Vecchi MP. Optimization by simulated annealing. Science. 1983;220(4598):671–680.
[7] Pannetier J, Bassas-Alsina J, Rodriguez-Carvajal J, et al Prediction of crystal structures from crystal chemistry rules by simulated annealing. Nature. 1990;346(6282):343–345.
[8] Goedecker S. Minima hopping: an efficient search method for the global minimum of the potential energy surface of complex molecular systems. J Chem Phys. 2004;120(21):9911–9917.
[9] Amsler M, Goedecker S. Crystal structure prediction using the minima hopping method. J Chem Phys. 2010;133(22):224104.
[10] Bush TS, Catlow CR, Battle PD. Evolutionary programming techniques for predicting inorganic crystal structures. J Mater Chem. 1995;5(8):1269–1272.
[11] Oganov AR, Glass CW. Crystal structure prediction using *ab initio* evolutionary techniques: principles and applications. J Chem Phys. 2006;124(24):244704.
[12] Oganov AR, Lyakhov AO, Valle M. How evolutionary crystal structure prediction works—d why. Acc Chem Res. 2011;44(3):227–237.
[13] Lyakhov AO, Oganov AR, Stokes HT, et al New developments in evolutionary structure prediction algorithm USPEX. Comput Phys Commun. 2013;184(4):1172–1182.
[14] Wang Y, Lv J, Zhu L, et al Crystal structure prediction via particle-swarm optimization. Phys Rev B. 2010;82:094116.
[15] Zhang Y, Wang H, Wang Y, et al Computer-assisted inverse design of inorganic electrides. Phys Rev X. 2017;7:011017.
[16] Yamashita T, Sato N, Kino H, et al Crystal structure prediction accelerated by Bayesian optimization. Phys Rev Mater. 2018;2(1):013803.
[17] Sato N, Yamashita T, Oguchi T, et al Adjusting the descriptor for a crystal structure search using Bayesian optimization. Phys Rev Mater. 2020;4(3):033801.
[18] Terayama K, Yamashita T, Oguchi T, et al Fine-grained optimization method for crystal structure prediction. NPJ Comput Mater. 2018;4(1):32.
[19] Behler J, Parrinello M. Generalized neural-network representation of high-dimensional potential-energy surfaces. Phys Rev Lett. 2007;98:146401.
[20] Bartók AP, Payne MC, Kondor R, et al Gaussian approximation potentials: the accuracy of quantum mechanics, without the electrons. Phys Rev Lett. 2010;104:136403.
[21] Deringer VL, Csányi G, Proserpio DM. Extracting crystal chemistry from amorphous carbon structures. ChemPhysChem. 2017;18(8):873–877.
[22] Deringer VL, Proserpio DM, Csányi G, et al Data-driven learning and prediction of inorganic crystal structures. Faraday Discuss. 2018;211:45–59.
[23] Podryabinkin EV, Tikhonov EV, Shapeev AV, et al Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning. Phys Rev B. 2019;99(6):064114.
[24] Tipton WW, Hennig RG. A grand canonical genetic algorithm for the prediction of multi-component phase diagrams and testing of empirical potentials. J Phys: Condens Matter. 2013;25(49):495401.

[25] Lonie DC, Zurek E. Xtalopt: an open-source evolutionary algorithm for crystal structure prediction. Comput Phys Commun. 2011;182(2):372–387.
[26] Ueno T, Rhone TD, Hou Z, et al Combo: an efficient bayesian optimization library for materials science. Mater Discovery. 2016;4:18–21.
[27] 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.
[28] Fredericksa S, Sayrea D, Zhua Q. PyXtal: a Python library for crystal structure generation and symmetry analysis. Comput Phys Commun. 2021;261:107 810.
[29] Kresse G, Furthmüller J. Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set. Phys Rev B. 1996;54:11169–11186.
[30] Giannozzi P, Baroni S, Bonini N, et al Quantum espresso: a modular and open-source software project for quantum simulations of materials. J Phys: Condens Matter. 2009;21(39):395502.
[31] OpenMX, open source package for material explorer. Available from: http://www.openmx-square.org .
[32] soiap is a classical molecular dynamics code. Available from: https://github.com/nbsato/soiap .
[33] Plimpton S. Fast parallel algorithms for short-range molecular dynamics. J Comput Phys. 1995;117(1):1–19.
[34] Momma K, Izumi F. *VESTA3* for three-dimensional visualization of crystal, volumetric and morphology data. J Appl Crystallogr. 2011;44(6):1272–1276.
[35] find_wy can find a suitable combination of Wyckoff positions. Available from: https://github.com/nim-hrkn/find_wy .
[36] Blöchl PE. Projector augmented-wave method. Phys Rev B. 1994;50(24):17953–17979.
[37] Perdew JP, Burke K, Ernzerhof M. Generalized gradient approximation made simple. Phys Rev Lett. 1996;77(18):3865–3868.
[38] Homma K, Yonemura M, Kobayashi T, et al Crystal structure and phase transitions of the lithium ionic conductor Li_3PS_4. Solid State Ion. 2011;182(1):53–58.
[39] Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive black-box functions. J Global Optim. 1998;13(4):455–492.
[40] Snoek J, Larochelle H, Adams R. Practical bayesian optimization of machine learning algorithms. Adv Neural Inf Process Syst. 2012;4:2951–2959.
[41] Chapelle O, Li L. An empirical evaluation of thompson sampling. Adv Neural Inf Process Syst. 2011;24:2249–2257.
[42] Behler J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. J Chem Phys. 2011;134(7):074106.
[43] Gastegger M, Schwiedrzik L, Bittermann M, et al WACSF - Weighted atom-centered symmetry functions as descriptors in machine learning potentials. J Chem Phys. 2018;148(24):241709.
[44] Bartók AP, Kondor R, Csányi G. On representing chemical environments. Phys Rev B. 2013;87 (18):184115.
[45] Lam Pham T, Kino H, Terakura K, et al Machine learning reveals orbital interaction in materials. Sci Technol Adv Mater. 2017;18(1):756–765.
[46] Xie T, Grossman JC. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. Phys Rev Lett. 2018;120(14):145301.
[47] Zimmermann NER, Jain A. Local structure order parameters and site fingerprints for quantification of coordination environment and crystal structure similarity. RSC Adv. 2020;10(10):6063–6081.
[48] Oganov AR, Valle M. How to quantify energy landscapes of solids. J Chem Phys. 2009;130(10):104504.
[49] Moriconi R, Deisenroth MP, Sesh Kumar KS. High-dimensional bayesian optimization using low-dimensional feature spaces. Mach Learn. 2020;109(9): 1925–1943.
[50] Shahriari B, Swersky K, Wang Z, et al Taking the human out of the loop: a review of bayesian optimization. Proc IEEE. 2016;104(1):148–175.
[51] Rahimi A, Recht B. Random features for large-scale kernel machines. Adv Neural Inf Process Syst. 2007;4:1177–1184.
[52] Rasmussen CE, Williams CKI. Gaussian processes for machine learning. Cambridge: The MIT Press; 2006.
[53] Kushner HJ, New A. Method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. J Basic Eng. 1964;86(1):97–106.