Algorithm for generating irreducible site-occupancy configurations

Ji-Chun Lian,[1] Hong-Yu Wu,[1] Wei-Qing Huang,[1,*] Wangyu Hu,[2] and Gui-Fang Huang[1,†]

[1]*Department of Applied Physics, School of Physics and Electronics, Hunan University, Changsha 410082, China*

[2]*School of Materials Science and Engineering, Hunan University, Changsha 410082, China*

(Received 18 May 2020; accepted 29 September 2020; published 22 October 2020)

Generating irreducible site-occupancy configurations by taking advantage of crystal symmetry is a ubiquitous method for the acceleration of disordered structure prediction, which plays an important role in condensed matter physics and material science. Here, we present an algorithm for generating irreducible site-occupancy configurations, which works for arbitrary parent cells with any supercell expansion matrix, and for any number of atom types with arbitrary stoichiometry. The algorithm identifies the symmetrically equivalent configurations by searching the space group operations of the underlying lattice and building the equivalent atomic matrix based on it. Importantly, an integer representation of configurations can greatly accelerate the speed of elimination of duplicate configurations, resulting in a linear scale of run time with the number of irreducible configurations that are finally found. Moreover, based on our algorithm, we write the corresponding code named *disorder* in FORTRAN programming language, and the performance test results show that the time efficiency of our *disorder* code is superior to that of other related codes (*supercell*, *enumlib*, and SOD).

DOI: 10.1103/PhysRevB.102.134209

I. INTRODUCTION

Searching unknown structures is of paramount potential impact in condensed matter physics and materials science [1–4]. The high efficiency and accuracy of first-principles calculations in describing the structural thermodynamic stability [5–7] and other related properties [8–10] has greatly promoted the process of structure prediction; extensive structural ground state properties calculations become reality. However, the evaluating of structural relative stabilities based on first-principles calculations becomes challenging for the issue of site-occupancy disorder [11–15]; a subset of the structure prediction problem, whose structures are derived from a specific underlying lattice. The reason is that the underlying lattice is often a large supercell, which not only increases the computational load of first-principles, but also the number of structures derived from it dramatically increases due to combinatorial explosion. Nevertheless, the efforts have not been stopped and will never be stopped, as many profound physical and material phenomena are observed in disordered structures, and bring about important applications in fields such as metallic alloys [16,17], nonstoichiometric materials [18,19] and high-temperature superconductors [20–23]. Aiming to overcome this obstacle, several methods have been developed to reduce the generated atomic configurations [24–27]. In this paper, we focus on one of those methods, i.e., searching the irreducible configurations in a complete list of atomic configurations by taking advantage of crystal symmetry.

Many combinatorially distinct configurations are geometrically identical; they are related by the symmetry operations of the underlying lattice. The main idea of searching irreducible configurations is to generate an exhaustive list of combinatorially distinct configurations, then eliminating the duplicates, i.e., the symmetrically equivalent configurations. Such an idea is adopted by many existing algorithms [24,28–30], but the implementations are diverse from each other. In the following, several representative algorithms for generating irreducible site-occupancy configurations will be briefly introduced.

The algorithm implemented in SOD software (released in 2007) is one of the excellent representatives [24]. The SOD algorithm identifies the equivalent configurations utilizing space group operations of the underlying lattice, which are reading from a database of space group operations. The superiority of the SOD algorithm lies in its simplicity in concept and programming. However, it only works for the binary site-occupancy systems, and the nondiagonal supercell expansion matrix is forbidden in SOD software, although these shortcomings are caused by programming limitations, not by the approach itself. In addition, its run time will grow explosively with the increase of the number of configurations, which is mainly caused by an algorithmic problem (evitable), and partly resulted from a combinatorial problem (ineluctable).

Compared with the SOD algorithm, extraordinary progress is made by the algorithm implemented in *enumlib* software (released in 2008) [28]. The *enumlib* algorithm applies to any parent cell, arbitrary supercell expansion matrix, and multinary (binary, ternary, quaternary etc.) site-occupancy systems. For the first version, the *enumlib* algorithm applies only to the Bravais lattices, and it does not work at a certain stoichiometry. These problems have been solved in the latter two extensions [31,32]. The key concept of the *enumlib* algorithm is to use the quotient group associated with the underlying lattice and an integer representation of the configurations to determine all unique structures. Profiting by this concept,

*Corresponding author: wqhuang@hnu.edu.cn
†Corresponding author: gfhuang@hnu.edu.cn

the *enumlib* algorithm is orders of magnitude faster than the SOD algorithm and realized a linear scale of run time with the number of irreducible configurations, which is the best possible scaling for this type of problem.

Although the run time of the *enumlib* algorithm scales linearly with the number of irreducible configurations, the calculation time is far beyond a reasonable limit when the number of configurations is huge (say millions). Therefore, the algorithm performance still has the possibility to be further improved, which has become a reality in *supercell* software (released in 2016) [30]. The algorithm implemented in *supercell* is similar in essence to the *enumlib* algorithm, but its time efficiency is about ten times that of *enumlib* algorithm. However, the *supercell* algorithm cannot support nondiagonal supercell expansion matrix, which limits its exploration for different supercell shapes.

In this paper, we present an implementation for the generation algorithm of irreducible site-occupancy configurations. In our algorithm, we enumerate the combination configurations to represent the combinatorially distinct atomic configurations, and extend it to the multinary site-occupancy systems. Our algorithm identifies the symmetrically equivalent configurations by an equivalent atomic matrix, while the space group operations used to build it are searched from the structural information (only the lattice parameters and atomic positions are needed) of arbitrary supercell straightforwardly. As a result, the algorithm is applicable for arbitrary parent cells with any supercell expansion matrix including the nondiagonal one, which is forbidden in SOD and *supercell* softwares. In addition, a linear scale of run time with the number of irreducible configurations is achieved also, benefitted by the concept of using a series of consecutive integers to represent the configurations. Most of all, the speed of eliminating duplicate configurations is greatly accelerated, due to the efficient conversion algorithm of configurations to integers, and hence faster than that of the *supercell* algorithm.

II. METHODOLOGY

The algorithm for generating irreducible site-occupancy configurations can be summarized into four critical tasks: (1) searching space group operations to build the equivalent atoms matrix, (2) enumerating all atomic configurations based on combinatorics, (3) eliminating the duplicate configurations by using symmetries (the most time-consuming procedure), (4) converting atomic configurations into consecutive integers to accelerate the third task. In the following, a detailed description of the algorithm implementation is presented.

A. Searching space group operations

Searching space group operations is a key point for identifying equivalent atomic configurations. The algorithm for searching space group operations is based on *Spglib*: a software library for crystal symmetry search [33]. Although the original algorithm implemented in *Spglib* is used for the primitive cell, it can be applied to an arbitrary cell with appropriate modifications. The algorithm of searching space group operations used here is a simplified and modified version of *Spglib*.

The space group operation is defined by a 3×3 integer matrix $\boldsymbol{W}$ (rotation part) and a 3×1 decimal column vector $\boldsymbol{\omega}$ (translation part). An arbitrary atomic point $\boldsymbol{x}$ on the fractional coordinates must be sent to another atomic point (or itself) $\boldsymbol{x}'$ with the same atomic type by one of the valid space group operations $(\boldsymbol{W}, \boldsymbol{\omega})$ by $\boldsymbol{x}' = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{\omega}$.

In the following, a brief outline of the algorithm for searching all valid space group operations is presented (the elaborated description is given in Ref. [33]).

1. *Searching pure translation operations* **[See Ref. [33], Step (a)]**

The pure translation operations are expressed as $(\boldsymbol{I}, \boldsymbol{\omega})$, where $\boldsymbol{I}$ is the identity matrix. Generally, the input cell (i.e., the underlying lattice) is a nonprimitive cell, so that there are multiple pure translation operations, $(\boldsymbol{I}, \boldsymbol{\omega})$ will be found, and the number (N_t) of pure translation operations is the volume of the input cell divided by the volume of its primitive cell.

2. *Searching lattice point group operations* **[See Ref. [33], Step (f)]**

In this step, all the matrix elements of possible $\boldsymbol{W}$ are selected from $\{-1, 0, 1\}$ to satisfy $|\det(\boldsymbol{W})| = 1$. Afterwards, the possible $\boldsymbol{W}$ are further screened by the conditions for metric tensor $\boldsymbol{G}$, which is defined by $\boldsymbol{G} = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})^{\mathrm{T}}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$, where $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ is the lattice basis vectors. In the *spglib*, the basis vectors used to define metric tensor $\boldsymbol{G}$ is that of the primitive cell. Here, we replace it with that of the input cell to obtain the space group operations of arbitrary cell.

3. *Searching space group operations* **[See Ref. [33], Step (g)]**

For a nonprimitive cell, similar to pure translation operations, one $\boldsymbol{W}$ will correspond to multiple $\boldsymbol{\omega}$. Thus, the total number (N_s) of space group operations is the number (N_r) of its rotation part multiply by the number (N_{t}) of its translation part, i.e., $N_s = N_r \times N_t$. In reality, for one $\boldsymbol{W}$, only one (any one) corresponding $\boldsymbol{\omega}$ needs to be searched out, while the full space group operations can be obtained by combining pure translation operations.

4. *Building equivalent atomic matrix*

Building an equivalent atomic matrix will advantageous to identify the equivalent atomic configurations conveniently and quickly. A set of atomic points labeled with consecutive integers are transformed into another set of atomic points by a space group operation. The labels of the transformed atomic points constitute one row elements of the equivalent atomic matrix, while the complete equivalent atomic matrix can be obtained by traversing all space group operations.

B. Enumerating all atomic configurations

The enumeration of all possible atomic configurations is based on combinatorics. We first consider an underlying lattice with K atomic positions that can be occupied by N different types of atoms (the vacancies are deemed as a special type of atoms), and the atomic number of each type is $k_i (i = 1 \dots N)$, satisfying $K = \sum_{i=1}^{N} k_i$. The total number (N_{c}) of atomic configurations can be obtained by the multinomial

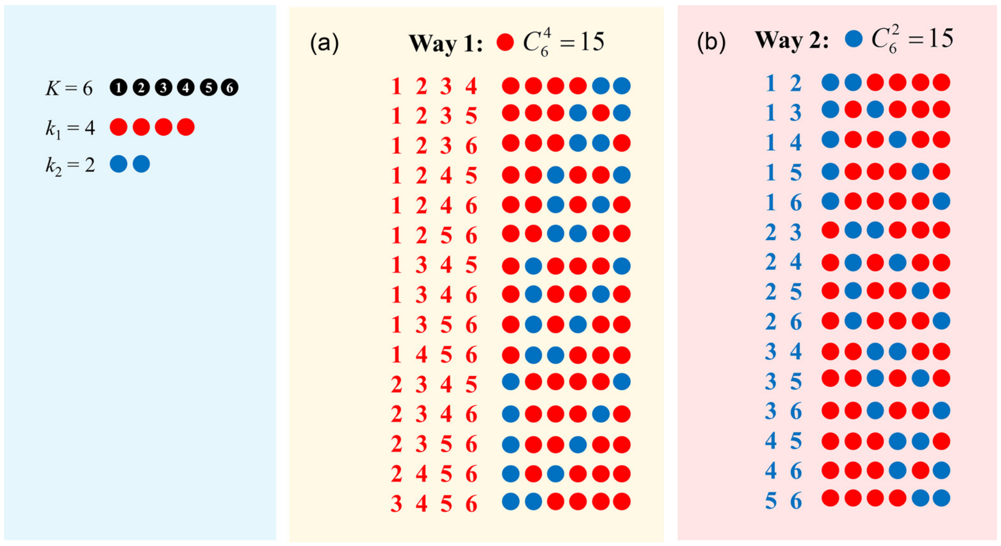

FIG. 1. The two enumeration ways of binary site-occupancy configurations for $K = 6, k_1 = 4$ (red), $k_2 = 2$ (blue). (a) The first enumeration way, i.e., enumerating the configurations for the red color, while the blue color is occupying the remaining atomic positions. (b) The second enumeration way, i.e., enumerating the configurations for the blue color, while the red color is occupying the remaining atomic positions.

coefficient:

$$N_c(k_1, k_2, \dots, k_N) = \frac{(k_1 + k_2 + \dots + k_N)!}{k_1!k_2!\cdots k_N!} = \frac{\left(\sum_{i=1}^N k_i\right)!}{\prod_{i=1}^N k_i!}. \tag{1}$$

For clarity, we will introduce our algorithm in two steps. Specifically, we start with a special case, i.e., the binary ($N = 2$) site-occupancy case, and then, extend it to the general case, i.e., the multinary ($N \geqslant 3$) site-occupancy case.

1. *Binary site-occupancy*

The binary site-occupancy is two different types of atoms occupying K atomic positions. In such a special case, according to Eq. (1), the total number of atomic configurations is expressed as a binomial coefficient:

$$N_c(k_1, k_2) = \frac{(k_1 + k_2)!}{k_1!k_2!} = \frac{K!}{k_1!k_2!} = C_K^{k_1} = C_K^{k_2}, \tag{2}$$

where $C_n^m = \frac{n!}{m!(n-m)!}$.

As mentioned above, our task is enumerating all possible atomic configurations. The algorithm enumerates all binary combination configurations to represent the atomic configurations. The binary combination configurations, corresponding to binomial coefficient C_n^m, are the possible ways to choose a subset of size m elements, disregarding their order, in an integer set from 1 to n with n elements. Hereafter, for convenience, we use different colors to represent different types of atoms, and a list **A** with $C_K^{k_1}$ rows and k_1 columns is adopted to store the atomic configurations for binary site-occupancy. Obviously, we only need to determine the atomic configurations of one color, while the others occupy the remaining atomic positions. Therefore, there are two ways to enumerate atomic configurations **A**, as shown in Fig. 1, where an example is for $K = 6$, $k_1 = 4$ (red), $k_2 = 2$ (blue). In practice, undoubtedly, we can enumerate the lesser one of the two colors to reduce the computational load.

2. *Multinary site-occupancy*

The multinary site-occupancy is N ($N \geqslant 3$) different types of atoms occupying K atomic positions. The total number of possible atomic configurations for the multinary site-occupancy can be obtained from Eq. (1). Here we rewrite it as

$$\begin{aligned} N_c(k_1, k_2, \dots, k_N) &= \frac{K!}{k_1!k_2!\cdots k_N!} \\ &= C_K^{k_1} C_{K-k_1}^{k_2} C_{K-k_1-k_2}^{k_3} \cdots C_{K-k_1-k_2-\dots-k_{N-2}}^{k_{N-1}}. \end{aligned} \tag{3}$$

Its physical meaning is obvious: first let k_1 atoms occupying K atomic positions, then let k_2 atoms occupying the remaining $K - k_1$ atomic positions, and so on. Of course, the order of $k_1, k_2, \dots, k_N$ is irrelevant. This means that we can decompose the multinary site-occupancy into several binary site-occupancies.

We represent these binary combination configurations, corresponding to binomial coefficients in Eq. (3), as lists $\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \dots, \mathbf{B}^{(N-1)}$, with $C_K^{k_1}, C_{K-k_1}^{k_2}, \dots, C_{K-k_1-k_2-\dots-k_{N-2}-2}^{k_{N-1}}$ rows and $k_1, k_2, \dots, k_{N-1}$ columns. Then, the multinary combination configurations **B** can be defined by simply uniting these binary combination configurations, as follows:

$$\begin{aligned} &B_n = B_i^{(1)} \cup B_j^{(2)} \cup \dots \cup B_k^{(N-1)}, \\ &n \in [1, N_c(k_1, k_2, \dots, k_N)], i \in \left[1, C_K^{k_1}\right], j \in \left[1, C_{K-k_1}^{k_2}\right], \\ &k \in \left[1, C_{K-k_1-k_2-\dots-k_{N-2}}^{k_{N-1}}\right]. \end{aligned}$$

However, unlike the binary site-occupancy, the combination configurations defined above are not the atomic configurations. Therefore, we need to convert the combination configurations to the atomic configurations.

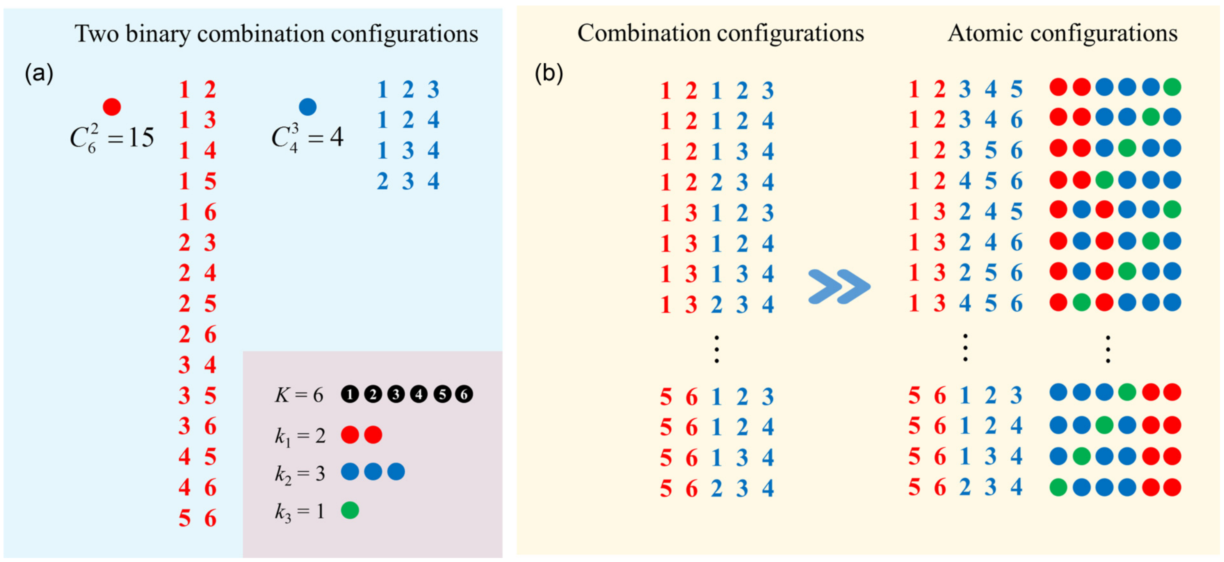

FIG. 2. The enumeration of ternary site-occupancy configurations for $K = 6$, $k_1 = 2$ (red), $k_2 = 3$ (blue), $k_3 = 1$ (green). (a) The binary combination configurations for red and blue colors. (b) The conversion of combination configurations to atomic configurations.

Given configuration **a**, and its corresponding combination configuration **b** can be disassembled into

$$a = a^{(1)} \cup a^{(2)} \cup \cdots \cup a^{(N-1)},$$
$$b = b^{(1)} \cup b^{(2)} \cup \cdots \cup b^{(N-1)}.$$

Moreover, a series of arrays $(L^{(1)}, L^{(2)}, \ldots, L^{(N-1)})$ with $(K, K - k_1, \ldots, K - k_1 - \cdots - k_{N-2})$ columns are used to label the atomic positions, where $L^{(1)}$ is the original atomic label, $L^{(2)}$ is the atomic label after removing $a^{(1)}$, and the like, $L^{(N-1)}$ is the atomic label after removing $a^{(1)} \cup a^{(2)} \cup \cdots \cup a^{(N-2)}$. Then, the transformation relationship between **a** and **b** can be written as

$$a_i^{(n)} = L_{b_i^{(n)}}^{(n)}, \tag{4}$$

where, $n \in [1, N - 1]$, $i \in [1, k_n]$, and i is the column index of $a^{(n)}$ or $b^{(n)}$. Here, we take the ternary site-occupancy as an example, and specify $K = 6$, $k_1 = 2$ (red), $k_2 = 3$ (blue), $k_3 = 1$ (green). For the ternary site-occupancy, we can decompose it into two binary site-occupancy, the combination configurations of them and the resulting atomic configurations are shown in Fig. 2.

C. Converting configurations to integers

As stated previously, the third task is eliminating the duplicate configurations, which is the most time-consuming procedure. The process of eliminating duplicate configurations can be greatly accelerated by the directly determination of configuration indexes, which is expressed as a set of consecutive integers. In the following, we will show how it is implemented and explain why it is so efficient.

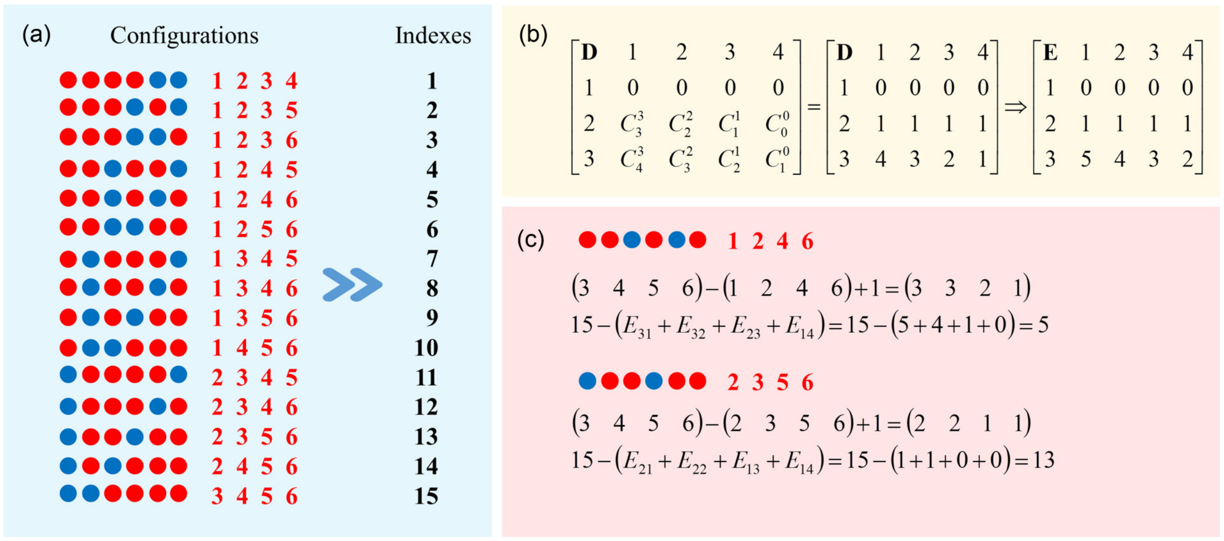

FIG. 3. The conversion of binary site-occupancy configurations to consecutive integers (indexes) for $K = 6$, $k_1 = 4$ (red), $k_2 = 2$ (blue). (a) The atomic configurations and its corresponding indexes. (b) The constructing flow of matrix **E**. (c) Two specific examples of atomic configurations to integers.

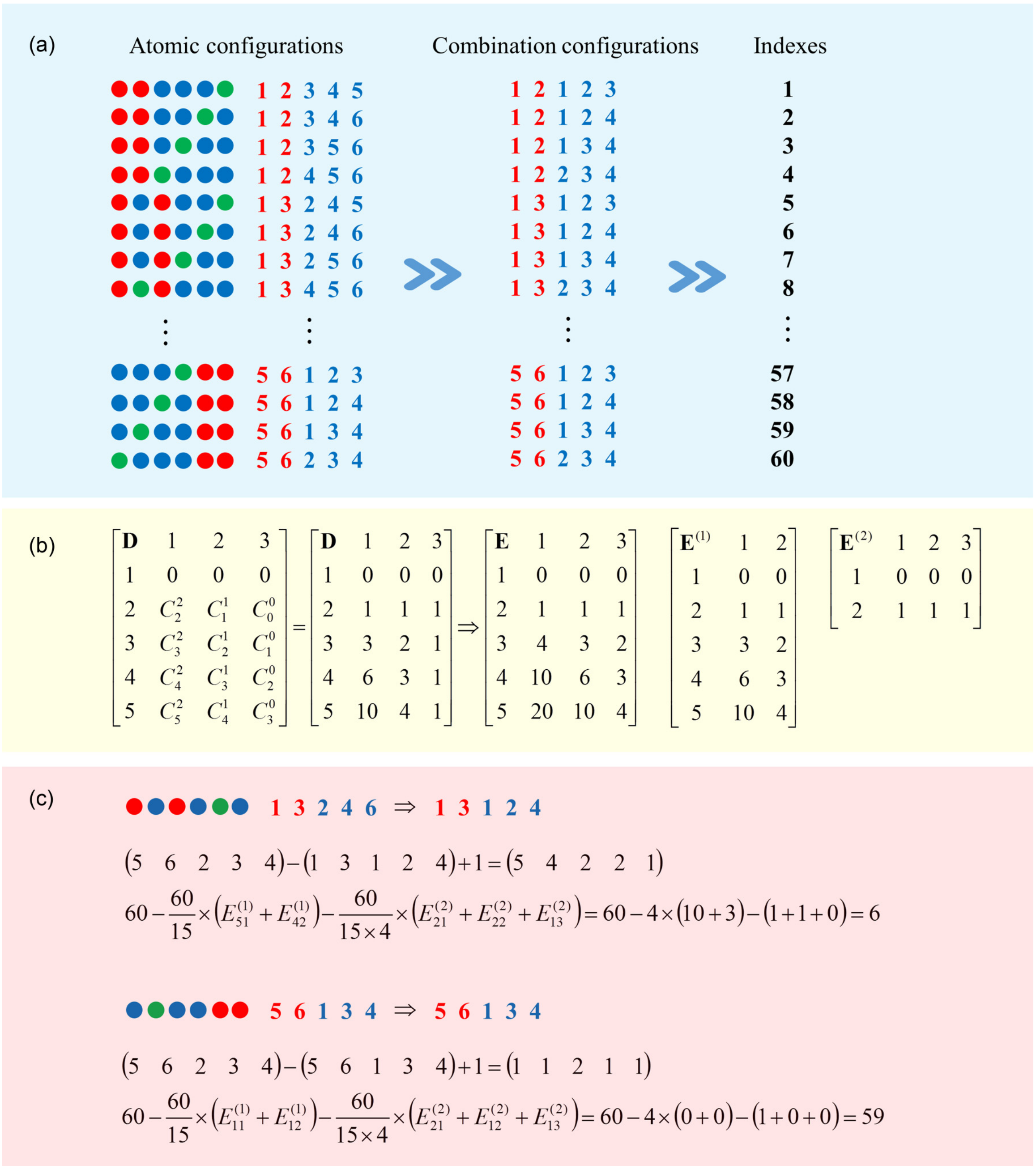

FIG. 4. The conversion of ternary site-occupancy configurations to consecutive integers (indexes) for $K = 6$, $k_1 = 2$ (red), $k_2 = 3$ (blue), $k_3 = 1$ (green). (a) The atomic configurations and its corresponding combination configurations and indexes. (b) The constructing flow of matrix $\mathbf{E}$ and its submatrices $\mathbf{E}^{(1)}$, $\mathbf{E}^{(2)}$. (c) Two specific examples of atomic configurations to integers.

1. Binary site-occupancy

For the binary site-occupancy, the configuration index of arbitrarily atomic configuration can be calculated by the following procedure. First, we build a matrix $\mathbf{D}$ as follows:

$$D_{ij} = \begin{cases} 0, i = 1 \\ C_{k_1-j+i-2}^{k_1-j}, i \neq 1 \end{cases},$$

where, the number of rows is $K - k_1 + 1$, the number of columns is k_1, i (j) is the row (column) index, and we define $C_n^0 = 1$. Based on matrix $\mathbf{D}$, another matrix $\mathbf{E}$ with the same size can be obtained, as shown below:

$$E_{ij} = \sum_{k=1}^{i} D_{kj}. \tag{5}$$

Then, given a configuration $\mathbf{a}$, its configuration index m can be expressed as

$$m = C_K^{k_1} - \sum_{j=1}^{k_1} E_{ij}, \tag{6}$$

where $i = A_{-1j} - a_j + 1$, A_{-1} is the last configuration in list **A**. Again, we specify $K = 6$, $k_1 = 4$ (red), $k_2 = 2$ (blue) as an example, and the calculation flow is shown in Fig. 3.

2. *Multinary site-occupancy*

For the multinary site-occupancy, similar to matrix **E** [see Eq. (5)] of the binary site-occupancy, a set of matrices $\mathbf{E}^{(n)}$ corresponding to the nth combination configurations can be built using the same method. Actually, however, these $N-1$ matrices are submatrices of a larger matrix, meaning that we only need to construct one matrix **E** whose number of rows and columns are the maximum of that in matrices $\mathbf{E}^{(n)}$. Afterwards, given configuration **a**, its configuration index m can be expressed as

$$m = N_c - \frac{N_c}{C_K^{k_1}}\sum_{j_1=1}^{k_1} E^{(1)}_{i_1 j_1} - \frac{N_c}{C_K^{k_1}C_{K-k_1}^{k_2}}\sum_{j_2=1}^{k_2} E^{(2)}_{i_2 j_2} - \cdots - \sum_{j_{N-1}=1}^{k_{N-1}} E^{(N-1)}_{i_{N-1} j_{N-1}},$$

where, $N_c = \frac{K!}{k_1!k_2!\cdots k_N!}$, $i_n = B^{(n)}_{-1j_n} - b^{(n)}_{j_n} + 1, n \in [1, N-1]$. Therefore, we need to convert the atomic configuration **a** into the corresponding combination configuration **b**, and it is not difficult to realize by the inverse process of Eq. (4). In addition, one can notice that when $N = 2$, the above formula will degenerate into the binary site-occupancy case [see Eq. (6)]. Finally, we specify $K = 6$, $k_1 = 2$ (red), $k_2 = 3$ (blue), $k_3 = 1$ (green) as an example, and the calculation flow is shown in Fig. 4.

D. Eliminating duplicate configurations

Our algorithm used to eliminate duplicate configurations is illustrated by a flowchart as shown in Fig. 5. The preprocessing algorithm (the red frame) involves the obtaining of space group operations, equivalent atomic matrix, all binary combination configurations, and matrix **E**. Afterwards, a parent-loop (the blue frame) traversing all atomic configurations, and a subloop (the green frame) traversing all space group operations are performed. In the parent-loop, if the atomic configuration has been eliminated, then skip it directly, otherwise add it to the list **I** (represents the irreducible configurations), and enters the subloop. In the subloop, the space group operation, including rotation, translation, and permutation, maps a configuration **a** to another configuration (or itself) **a′**. If the configuration **a′** has been eliminated, then skip it again, otherwise eliminate it. Obviously, the key point of the above process is how to eliminate the atomic configuration, and how to know a configuration has been eliminated. This is achieved by a logical list **U** corresponding to the atomic configurations one-by-one. At the beginning, all elements in **U** are setting to "**TRUE**", which means that all configurations have not been eliminated. If a configuration needs to be eliminated, set the corresponding element in **U** to "**FALSE**". All in all, the fast conversion of configurations to integers plays a pivotal role for the improvement of algorithm performance.

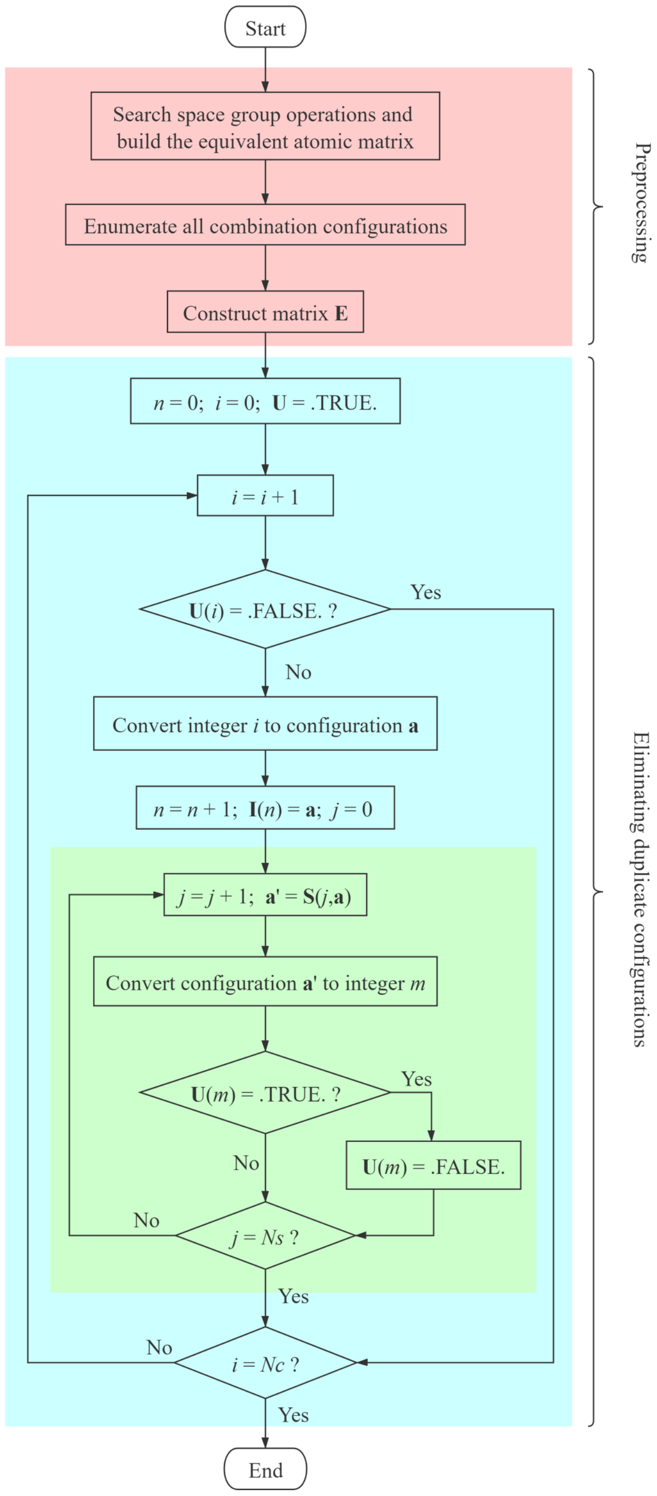
Start
Search space group operations and build the equivalent atomic matrix
Enumerate all combination configurations
Construct matrix E
Preprocessing
n = 0; i = 0; U = .TRUE.
i = i + 1
U(i) = .FALSE. ?
Yes
No
Convert integer i to configuration a
n = n + 1; I(n) = a; j = 0
j = j + 1; a' = S(j,a)
Convert configuration a' to integer m
U(m) = .TRUE. ?
Yes
U(m) = .FALSE.
No
No
j = Ns ?
Yes
No
i = Nc ?
Yes
End
Eliminating duplicate configurations

FIG. 5. The algorithm flowchart of eliminating duplicate configurations.

III. ALGORITHM PERFORMANCE AND SUPERIORITY

Based on aforementioned algorithm, we write the corresponding code named as *disorder* in FORTRAN programming language [34]. After that, an example is given to test the algorithm performance, and compared with those algorithms implemented in *supercell*, *enumlib,* and SOD softwares.

TABLE I. The calculation results of *disorder* (this work), *supercell*, *emumlib*, and SOD codes for all combinatorially distinct stoichiometries in a $2 \times 2 \times 2$ fcc lattice. The reported run time of the four codes is a real time on Intel® Xeon® E5-2678 processor, and only one thread is used during the test.

			Run time (sec)			
Stoichiometries	Total configurations	Irreducible configurations	*disorder*	*supercell*	*enumlib*	SOD
1:31	32	1	0.0218	13.354	0.0745	0.0681
2:30	496	5	0.0207	13.542	0.1517	0.0899
3:29	4960	14	0.0226	13.746	0.3102	0.5776
4:28	35960	71	0.0385	13.565	1.0856	21.173
5:27	201376	223	0.0812	13.721	3.2949	451.29
6:26	906192	874	0.2459	14.304	12.447	10248
7:25	3365856	2706	0.7622	16.553	39.866	> 2 days
8:24	10518300	8043	2.2275	23.266	119.01	/
9:23	28048800	20123	6.9478	36.764	300.72	/
10:22	64512240	45497	18.398	65.990	701.43	/
11:21	129024480	88716	41.692	116.29	1377.3	/
12:20	225792840	154379	81.531	191.36	2333.6	/
13:19	347373600	234803	137.11	286.38	3559.0	Crashed
14:18	471435600	318348	209.04	379.59	4857.5	Crashed
15:17	565722720	379926	283.44	454.09	5836.6	Crashed
16:16	601080390	404582	306.24	473.10	6123.4	Crashed
Cumulative	2248023842	1658311	1087.7	2125.7	25264	/

The example is based on a face-centered cubic (fcc) parent lattice, as many structures of intermetallic compounds are derived from it. The point group of the fcc lattice is $\boldsymbol{O_h}$, with 48 point group operations (rotation parts of space group operations), which is the highest symmetry. Moreover, the volume of the fcc unit cell (4 atoms) is 4 times of its primitive cell (1 atoms), which means that the fcc unit cell possesses 4 pure translation operations. For an fcc unit cell, therefore, the total number of space group operations is $48 \times 4 = 192$ (48 rotations and 4 translations). In our example, a $2 \times 2 \times 2$ supercell (32 atoms) is adopted to test the algorithm performance. This is a challenging test, because its $48 \times 4 \times 8 = 1536$ space group operations (48 rotations and 4×8 translations) is really high.

We enumerate all combinatorially distinct stoichiometries for binary site-occupancy and present the calculation results (containing total configurations, irreducible configurations and run time) in Table I. One can see that all four codes, including *disorder* (this work), *supercell*, *enumlib*, and SOD, give the same number of total and irreducible configurations, which proves the correctness of our algorithm. However, the run time of the four codes is different: the run time of SOD code grows sharply with the increase of the number of irreducible configurations, but that of other three codes grow slowly.

For the convenience of comparing the run time of *disorder*, *supercell*, and *enumlib* codes, we plot the run time as a function of the number of irreducible configurations, as shown in Fig. 6. One can notice that a linear relationship between run time and the number of irreducible configurations appeared when the number of irreducible configurations is greater than a critical value, for all the three codes. Such a critical value depends on the ratio of preprocessing time to total time: a linear scale emerged when the ratio is small enough, i.e., the preprocessing time is negligible compared with the total time. The large critical value of *supercell* is owing to its long preprocessing time primarily, while the small critical value of *enumlib* is mainly due to its long total time. For our *disorder* code, both the total time and preprocessing time are relatively small, so that its critical value is between that of *enumlib* and *supercell*. Moreover, we can see that our *disorder* code has the lowest run time throughout the stoichiometric range (see Fig. 6), and the cumulative run time (see last row of Table I) of *disorder* is 18 minutes, about half that of *supercell*

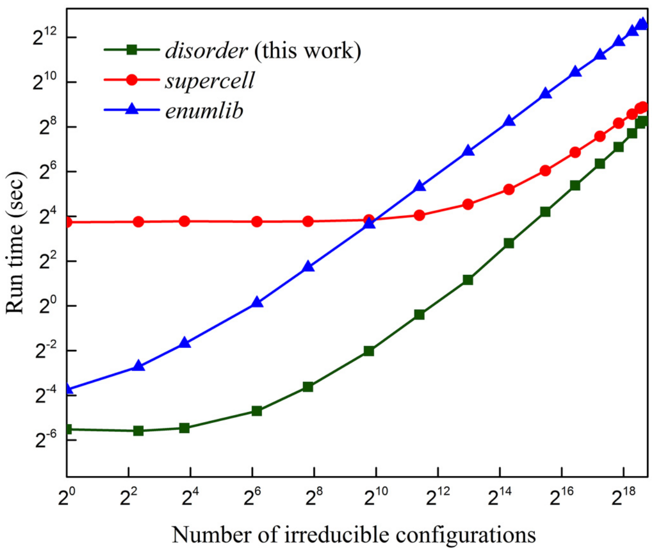

FIG. 6. The run time of *disorder*, *supercell*, and *enumlib* codes as a function of the number of irreducible configurations in a $2 \times 2 \times 2$ fcc lattice. One can see that the *disorder* code (this work) has the lowest run time in the whole range of the number of irreducible configurations.

TABLE II. The comparison of *disorder* (this work), *supercell*, *emumlib* and SOD codes. The performance is the cumulative time of all combinatorially distinct stoichiometries in a $2 \times 2 \times 2$ fcc lattice.

	disorder	*supercell*	*enumlib*	SOD
Public release	2020	2016	2008	2007
Version	/	v1.2	v2.0.4	v0.47
Released date	/	23-05-2019	23-09-2019	01-02-2019
Programming language	FORTRAN	C++	FORTRAN	FORTRAN
Nondiagonal supercell expansion matrix	Yes	No	Yes	No
Arbitrary number of atom types (i.e., the multinary systems)	Yes	Yes	Yes	No
Performance	18 min	35 min	7 h	/

(35 minutes) and less than one twenty-third that of *enumlib* (7 h), which indicates the high efficiency of our algorithm. As a summary, a brief comparison of *disorder, supercell*, *enumlib* and SOD codes are presented in Table II.

IV. SUMMARY

We have developed an algorithm for generating irreducible site-occupancy configurations. The algorithm processes the multinary site-occupancy problem by decomposing it into several binary site-occupancies. Afterwards, based on combinatorics, the algorithm enumerates all binary combination configurations, and converts them to atomic configurations. In the procedure of eliminating duplicate configurations, the algorithm identifies the duplicate configurations by an equivalent atomic matrix, which is built from the space group operations of the underlying lattice. In our algorithm, the space group operations are searched from the structural information of the underlying lattice directly, and works for an arbitrary parent cell with any supercell expansion matrix. Most importantly, an efficient conversion of configurations to integers is adopted to accelerate the process of eliminating duplicate configurations, and finally realized a linear scale of run time with the number of irreducible configurations, which is proved by the performance testing in a $2 \times 2 \times 2$ fcc lattice. Moreover, the results also indicate that the time efficiency of the algorithm is greatly superior to other algorithms with the same or different time complexity.

ACKNOWLEDGMENTS

We acknowledge financial support from National Natural Science Foundation of China (Grants No. 51772085 and No. U1830138)

[1] R. Caputo and A. Tekin, Ab initio crystal structure prediction. A case study: $NaBH_4$, J. Solid State Chem. **184**, 1622 (2011).

[2] R. Podeszwa, B. M. Rice, and K. Szalewicz, Predicting structure of molecular crystals from first-principles, Phys. Rev. Lett. **101**, 115503 (2008).

[3] Y. Wang, J. Lv, L. Zhu, and Y. Ma, Crystal structure prediction via particle-swarm optimization, Phys. Rev. B **82**, 094116 (2010).

[4] S. M. Woodley and R. Catlow, Crystal structure prediction from first-principles, Nat. Mater. **7**, 937 (2008).

[5] K. Hu, Q.-J. Chen, and S.-Y. Xie, Pressure induced superconductive tenfold coordinated TaS: A first-principles study, J. Phys.: Condens. Matter **32**, 085402 (2020).

[6] K. Hu, J. Lian, L. Zhu, Q. Chen, and S.-Y. Xie, Prediction of Fe_2P-type $TiTe_2$ under pressure, Phys. Rev. B **101**, 134109 (2020).

[7] A. J. Misquitta, G. W. A. Welch, A. J. Stone, and S. L. Price, A first-principles prediction of the crystal structure of C6Br2ClFH2, Chem. Phys. Lett. **456**, 105 (2008).

[8] P. A. Korzhavyi, I. A. Abrikosov, B. Johansson, A. V. Ruban, and H. L. Skriver, First-principles calculations of the vacancy formation energy in transition and noble metals, Phys. Rev. B **59**, 11693 (1999).

[9] J.-C. Lian, W.-Q. Huang, W. Hu, and G.-F. Huang, Electrostatic potential anomaly in 2D Janus transition metal dichalcogenides, Ann. Phys. (Berlin) **531**, 1900369 (2019).

[10] X. Y. Zhao and D. Vanderbilt, First-principles study of structural, vibrational, and lattice dielectric properties of hafnium oxide, Phys. Rev. B **65**, 233106 (2002).

[11] P. Soven, Contribution to the theory of disordered alloys, Phys. Rev. **178**, 1136 (1969).

[12] J. Biasco, J. Garcia, J. M. de Teresa, M. R. Ibarra, J. Perez, P. A. Algarabel, C. Marquina, and C. Ritter, Structural, magnetic, and transport properties of the giant magnetoresistive perovskites $La_{2/3}Ca_{1/3}Mn_{1-x}Al_xO_{3-\delta}$, Phys. Rev. B **55**, 8905 (1997).

[13] A. Landa, P. Soderlind, A. Ruban, L. Vitos, and L. Pourovskii, First-principles phase diagram of the Ce-Th system, Phys. Rev. B **70**, 224210 (2004).

[14] R. Grau-Crespo, N. H. de Leeuw, and C. R. A. Catlow, Cation distribution and magnetic ordering in $FeSbO_4$, J. Mater. Chem. **13**, 2848 (2003).

[15] V. Shuvaeva, Y. Azuma, K. Yagi, H. Terauchi, R. Vedrinski, V. Komarov, and H. Kasatani, Ti off-center displacements in $Ba_{1-x}Sr_xTiO_3$ studied by EXAFS, Phys. Rev. B **62**, 2969 (2000).

[16] J. Guevara, V. Vildosola, J. Milano, and A. M. Llois, Half-metallic character and electronic properties of inverse magnetoresistant $Fe_{1-x}Co_xSi$ alloys, Phys. Rev. B **69**, 184422 (2004).

[17] K. Ozdogan, E. Sasioglu, B. Aktas, and I. Galanakis, Doping and disorder in the Co_2MnAl and Co_2MnGa half-metallic Heusler alloys, Phys. Rev. B **80**, 029901 (2009).

[18] E. Koga, H. Moriwake, K.-I. Kakimoto, and H. Ohsato, Raman spectroscopic evaluation and microwave dielectric property of order/disorder and stoichiometric/nonstoichiometric $Ba(Zn_{1/3}Ta_{2/3})O_3$, Ferroelectrics **356**, 146 (2007).

[19] A. A. Rempel, A. I. Gusev, and M. Y. Belyaev, ^{93}Nb NMR study of an ordered and a disordered nonstoichiometric niobium carbide, J. Phys. C **20**, 5655 (1987).

[20] Chudnovsky, Hexatic vortex glass in disordered superconductors, Phys. Rev. B **40**, 11355 (1989).

[21] S. A. Davydov, B. N. Goshchitskii, A. E. Karkin, A. V. Mirmelshtein, V. I. Voronin, M. V. Sadovskii, V. L. Kozhevnikov, S. V. Verkhovskii, and S. M. Cheshnitskii, Effects of localization in atomic-disordered high-T_c superconductors, Int. J. Mod. Phys. B **03**, 87 (1989).

[22] I. Grosu, Veres, and Crisan, Decrease in critical temperature due to disorder and magnetic correlations in two-dimensional superconductors, Phys. Rev. B **50**, 9404 (1994).

[23] A. Williams, G. H. Kwei, R. B. Von Dreele, I. D. Raistrick, and D. L. Bish, Joint x-ray and neutron refinement of the structure of superconducting $YBa_2Cu_3O_{7-x}$, Phys. Rev. B **37**, 7960 (1988).

[24] R. Grau-Crespo, S. Hamad, C. R. A. Catlow, and N. H. de Leeuw, Symmetry-adapted configurational modelling of fractional site occupancy in solids, J. Phys.: Condens. Matter **19**, 256201 (2007).

[25] C. E. Mohn and W. Kob, A genetic algorithm for the atomistic design and global optimisation of substitutionally disordered materials, Comp. Mater. Sci. **45**, 111 (2009).

[26] J. A. Purton, M. Y. Lavrentiev, and N. L. Allan, Monte Carlo simulation of GaN/InN mixtures, J. Mater. Chem. **15**, 785 (2005).

[27] I. T. Todorov, N. L. Allan, M. Y. Lavrentiev, C. L. Freeman, C. E. Mohn, and J. A. Purton, Simulation of mineral solid solutions at zero and high pressure using lattice statics, lattice dynamics and Monte Carlo methods, J. Phys.: Condens. Matter **16**, S2751 (2004).

[28] G. L. W. Hart and R. W. Forcade, Algorithm for generating derivative structures, Phys. Rev. B **77**, 224115 (2008).

[29] S. Mustapha, P. D'Arco, M. De La Pierre, Y. Noel, M. Ferrabone, and R. Dovesi, On the use of symmetry in configurational analysis for the simulation of disordered solids, J. Phys.: Condens. Matter **25**, 105401 (2013).

[30] K. Okhotnikov, T. Charpentier, and S. Cadars, Supercell program: a combinatorial structure-generation approach for the local-level modeling of atomic substitutions and partial occupancies in crystals, J. Cheminformatics **8**, 17 (2016).

[31] G. L. W. Hart and R. W. Forcade, Generating derivative structures from multilattices: Algorithm and application to hcp alloys, Phys. Rev. B **80**, 014120 (2009).

[32] G. L. W. Hart, L. J. Nelson, and R. W. Forcade, Generating derivative structures at a fixed concentration, Comput. Mater. Sci. **59**, 101 (2012).

[33] A. Togo and I. Tanaka, *Spglib*: A software library for crystal symmetry search (unpublished).

[34] The latest version of *disorder* code is available on GitHub (https://github.com/jichunlian/disorder).