JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim

Application Note

SHRY: Application of Canonical Augmentation to the Atomic Substitution Problem

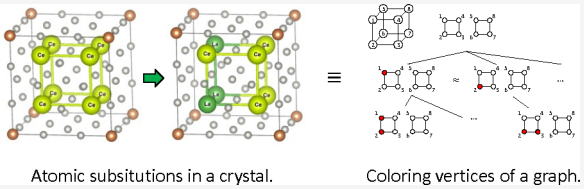
Genki Imam Prayogo,* Andrea Tirelli, Keishu Utimula, Kenta Hongo, Ryo Maezono, and Kousuke Nakano*

Cite This: *J. Chem. Inf. Model.* 2022, 62, 2909−2915

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: A common approach for studying a solid solution or disordered system within a periodic *ab initio* framework is to create a supercell in which certain amounts of target elements are substituted with other elements. The key to generating supercells is determining how to eliminate symmetry-equivalent structures from many substitution patterns. Although the total number of substitutions is on the order of trillions, only symmetry-inequivalent atomic substitution patterns need to be identified, and their number is far smaller than the total. Our developed Python software package, which is called SHRY (Suite for High-throughput generation of models with atomic substitutions implemented by Python), allows the selection of only symmetry-inequivalent structures from the vast number of candidates based on the canonical augmentation algorithm. SHRY is implemented in Python 3 and uses the CIF format as the standard for both reading and writing the reference and generated sets of substituted structures. SHRY can be integrated into another Python program as a module or can be used as a stand-alone program. The implementation was verified through a comparison with other codes with the same functionality, based on the total numbers of symmetry-inequivalent structures, and also on the equivalencies of the output structures themselves. The provided crystal structure data used for the verification are expected to be useful for benchmarking other codes and also developing new algorithms in the future.



INTRODUCTION

Atomic substitution in solids is the most common strategy in materials science to tune material properties for applications such as alloying and cation/anion/vacancy doping, which often demonstrate unprecedented functionalities superior to non-substituted materials.[1] Within the scientific and technological communities, there is a considerable demand for the ability to predict the extent to which substitutions affect material properties using *ab initio* simulations, as well as the ability to evaluate whether substitutions are useful for achieving the desired properties. For instance, if we consider vacancy-type defects as a form of substitution (by a vacancy), this demand covers another industrially important problem: understanding how defects affect material properties.[2,3] This problem arises during the evaluation of sample qualities, damages, and degradation. *Ab initio* calculations based on density functional theory (DFT) are the most promising framework for such an assessment; they utilize the microscopic structure models of substitutions.

There are many strategies for studying disordered crystals or solid solutions within the DFT framework. If one considers disorder effects at the DFT level with the preservation of periodicity of a target crystal, virtual crystal approximation (VCA)[4] or coherent potential approximation (CPA)[5] combined with the Korringa−Kohn−Rostoker (KKR) method[6,7] are good choices. The former simulates disordered crystals or solid solutions by mixing (all-electron or pseudo) potentials of the constituent elements based on the composition, whereas the latter utilizes an effective potential constructed from the Green's function of the effective medium. However, because both methods require special implementations, not all *ab initio* codes can perform them. The so-called *supercell method* is a considerably simpler yet very powerful strategy. In this method, substitutions are performed within the supercell of the original (i.e., unsubstituted) unit cell. While the supercell approach is simple and applicable for any *ab initio* framework, an intrinsic problem is that the approach introduces an *artificial* periodic boundary condition that does not exist in real materials. A direct approach to alleviate this artificial periodicity is to increase the size of the supercell. Although a larger supercell inevitably increases computational cost, recent advances in high-performance computing allow us to handle it even at the phonon analysis level.[8,9] Further, a larger supercell affords a finer resolution of substitution concentration compared to an unsubstituted unit cell because

Received: April 4, 2022
Published: June 9, 2022

https://doi.org/10.1021/acs.jcim.2c00389
J. Chem. Inf. Model. 2022, 62, 2909−2915

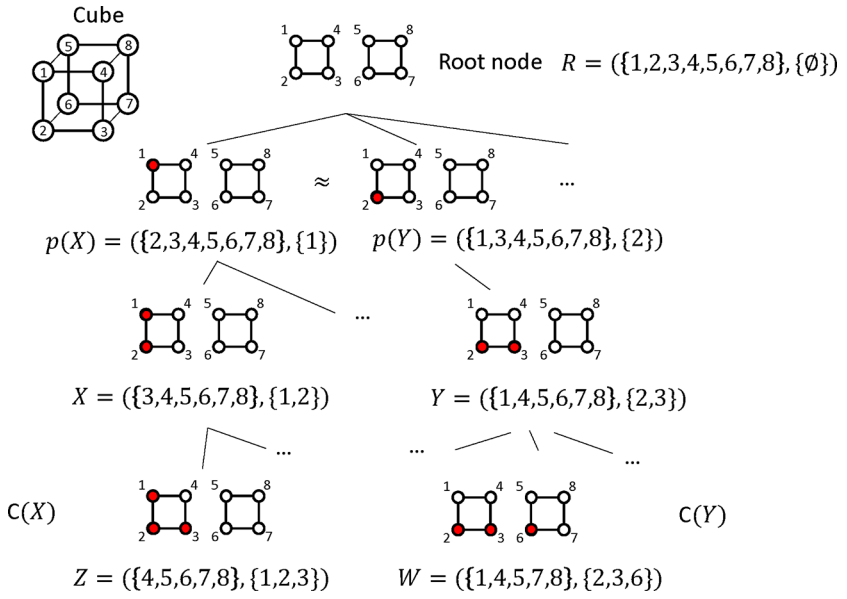


Figure 1. Example of the search tree, where three vertices out of eight are colored in red in a cube. $p(X)$ denotes the parent node of X, whereas $C(X)$ denotes the set of children of X.

only discrete substitutions are allowed in the supercell approach.[10] In addition, more sophisticated methods to alleviate the artificial periodicity have been devised. One such successful method is the method of quasi-random structures (SQS),[11,12] which finds the closest periodic supercell to a genuine disordered state based on correlation functions. If the thermodynamical properties are in focus, the cluster expansion method with Monte Carlo sampling[13] is another very effective technique for studying disordered crystals.[14] Several schemes have been developed[15−18] to correct spurious interactions between supercell images for studying charged defects in crystals more quantitatively. These methods ensure that the supercell approach is effective and reliable. In this paper, we focus on the supercell approach.

Within the supercell approach, a solid solution or disordered system is modeled with a supercell in which certain amounts of target elements are substituted with other elements. The number of possible atomic substitution patterns grows *combinatorically* with the substitution concentration and size of the supercell; this readily causes technical issues in the *ab initio* framework. A simple but powerful solution is to randomly select several configurations from the vast number of the possible combinations.[19,20] With an efficient implementation, this random sampling method is not limited by the number of configurations.[20] Further, the method is easy to implement. In this paper, we focus on an exhaustive approach of exploring all the symmetry-inequivalent structures. Indeed, many configurations in the redundant structure set are related to crystallographic symmetry operations; however, *ab initio* simulations require only symmetry-inequivalent structures. This implies that these *redundant* configurations containing many symmetry-equivalent configurations can be further categorized into those composed of *only symmetry-inequivalent* configurations. If the number of symmetry-inequivalent structures is too large, one can also select several feasible configurations after enumerating the symmetry-inequivalent structures according to a criterion such as the Ewald energy.

Several software packages have been developed to address such vast combinations of atomic substitution in a solid; examples include Supercell,[21] Materials Studio (i.e., Disorder tool),[22] enumlib[23] (also combined with Pymatgen (Python Materials Genomics)[24]), Crystal,[25,26] and Sod.[27] Recently, we developed a Python package called Shry (a Suite for High-throughput generation of models with atomic substitutions implemented in Python). Shry allows one to select only symmetry-inequivalent structures from a set of redundant structures based on the canonical augmentation algorithm.[28] Shry is implemented in Python 3 and uses the CIF format as the standard for both reading and writing the reference and generated sets of substituted structures. Shry can be integrated into another Python program as a module or can be used as a stand-alone program. In the following, we describe the package specification, underlying method, implementation, and several benchmark and validation tests.

SUMMARY OF TECHNICAL DETAILS

Shry is implemented in Python 3 and uses the CIF format as the standard for both reading and writing the reference and generated sets of substituted structures. Structure abstractions, manipulations, and symmetry analysis rely on Pymatgen[24] and spglib.[29] The key point for the implementation is identifying and avoiding, among all possible permutations, multiple instances of symmetrically identical structures. The implementation of a variant of the canonical augmentation[30] algorithm is unique to Shry, which allows symmetrically unique structures corresponding to the specified substitute concentration to be recursively generated from the unique structures. As a command line tool, a CIF format file containing site occupation information can be used as an input structure. Shry writes all the identified symmetry-inequivalent structures as separate CIF files. A user can also specify the maximum number of outputted CIF files if the total number of structures is too large to write all of them. The total number of symmetry-inequivalent structures can be estimated *a priori* by Polya's enumeration theorem,[31,32] which provides the expected number of irreducible structures. To enable a user to include Shry as a module in their own Python scripts, we provide

https://doi.org/10.1021/acs.jcim.2c00389
J. Chem. Inf. Model. 2022, 62, 2909−2915

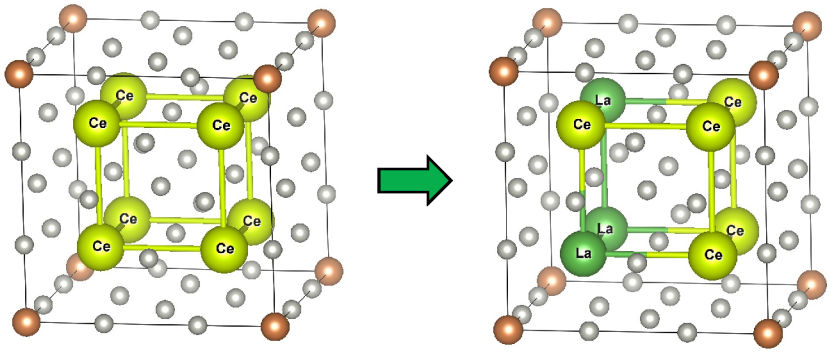


Figure 2. Atomic substitutions corresponding to Figure 1. $Ce_8Pd_{24}Sb \rightarrow (Ce_5,La_3)Pd_{24}Sb$, where Ce and La atoms are on the 8*g* Wyckoff site. The crystal structure[34] was obtained from the ICSD database[35,36] (CollCode: 83378). The space group is 221-$Pm\bar{3}m$. The crystal structures are depicted using VESTA.[37]

several examples of Python scripts in the distribution, which also include interfaces for SHRY with PYMATGEN.

Since our formulation described in the Supporting Information is quite formal (but entirely self-contained), we here present several hints on the properties of canonical augmentation. The objective of SHRY is to eliminate symmetry-equivalent structures from the possible atomic substitution patterns. A straightforward solution, classifying symmetry-inequivalent structures after determining all possible configurations, is redundant and practically unfeasible because the number of total combinations increases exponentially (i.e., combinatorial explosion). The key observation to alleviate the combinatorial explosion problem is that the atomic substitution problem can be mapped into a vertex coloring problem (Figures 1 and 2). Here, we consider a very simple case: the problem of coloring the eight vertices of a cube with two colors, as shown in Figure 1. We color white vertices with red in a stepwise manner until three of them become red. This is essentially equivalent to the substitution in which three out of eight atoms on a Wyckoff site are substituted with another element (e.g., see Figure 2; $Ce_8Pd_{24}Sb \rightarrow (Ce_5,La_3)Pd_{24}Sb$, where Ce and La atoms are on the 8*g* Wyckoff position and the space group is 221-$Pm\bar{3}m$). Then, the question of how many symmetry-inequivalent structures exist in all the possible substitution patterns essentially becomes equivalent to the question of how many symmetrically unique coloring patterns exist in a given graph.

A search tree to find the unique coloring patterns can be constructed as shown in Figure 1, where the search tree is truncated at the depth corresponding to the number of colored vertices. The tree is traversed by a proper algorithm such as the depth-first traversal procedure. The mapping itself does not solve the combinational explosion problem, because one visits all the nodes in the worst-case scenario. If one can truncate large parts of the search tree at the upper level without visiting lower nodes, the computational cost can be drastically decreased. For instance, in Figure 1, since $p(X)$ and $p(Y)$ are symmetrically equivalent, all the children of $p(Y)$ such as Y and W can be disregarded without visiting these nodes, if all the children associated with $p(X)$ such as X and Z have been already visited. This so-called *isomorphic rejection*[28] procedure is one of the key points in the present study. Such truncations should be performed using only *local information*. Indeed, if one needs to store all the information on the visited nodes for the truncation (i.e., *nonlocal information* is needed), the combinatorial explosion problem still remains. The orderly generation and canonical augmentation exploit the so-called *canonicity* for a node representation to truncate a search tree using only *local information* (see the Supporting Information for the mathematical definition). They are very powerful algorithms for the classification of combinatorial structures, the main idea of which is to construct only nonequivalent objects (namely, symmetry-inequivalent structures in the atomic substitution problem) and thereby classify such objects. The orderly generation and canonical augmentation were first introduced by Read[33] and McKay,[30] respectively.[28] They are similar algorithms in the sense that both exploit the canonicity. However, they also show some significant differences. The orderly generation requires that all nodes should themselves be canonical.[28] In contrast, the canonical augmentation does not require the canonicity of nodes themselves; rather, it requires a search tree to be generated *in a canonical manner*.[28] This feature of the canonical augmentation allows us to occasionally avoid the computation of time-consuming canonicity tests in the tree search. In the Supporting Information, we outline the mathematical details of the algorithms implemented in SHRY.

COMPARISON WITH EXISTING SOFTWARE PACKAGES

Since the atomic substitution problem is very general and important, several software packages have already been developed. To our knowledge, the site-occupancy disorder (SOD) package is the pioneering work in this field.[27] The software package has been applied for many disordered systems since 2007. Another open-source package with similar functionalities is the ENUMLIB package.[23] The code enables us to generate derivative superstructures of a parent lattice in a considerably efficient manner, and it has been integrated into PYMATGEN.[24] SUPERCELL[21] is the most recent and most sophisticated software package in the field. The package implements considerably efficient algorithms for the atomic substitution problem. The atomic substitution problem can be solved using several commercial software packages. The DISORDER TOOL implemented in MATERIALS STUDIO[22] can generate solid solutions with a very user-friendly graphical user interface (GUI). Similar functionalities are implemented in a well-known DFT code, CRYSTAL program,[26,38] to treat disorder systems within the periodic *ab initio* framework. Within the existing solutions, Okhotonikov et al.[21] reported that only the SUPERCELL package could handle cases with a large number of permutations because the other programs crash on such computationally demanding test cases. Therefore, for the

https://doi.org/10.1021/acs.jcim.2c00389
J. Chem. Inf. Model. 2022, 62, 2909−2915

Table 1. Total Number of Possible Atom Combinations for Different Supercell Sizes of the $A_{0.5}Pb_{0.5}Te$ System[a]

Total number of atoms in simulation cell	8	16	24	32	64
Supercell size $a \times b \times c$	1 × 1 × 1	1 × 1 × 2	1 × 1 × 3	1 × 2 × 2	2 × 2 × 2
Symmetry operations	192	128	192	256	1536
Total combinations (symmetry-inequivalent structures)	6(1)	70(8)	924 (34)	12,870 (153)	601,080,390 (404,582)
Performance of SHRY (s)	2.85(1.21)	1.70(1.17)	1.78(1.27)	1.85(1.39)	161.91(160.99)
Performance of SUPERCELL (s)	0.03(0.01)	0.03(0.02)	0.04(0.02)	0.06(0.04)	8.40(200.72)

[a]The computational times of SHRY and SUPERCELL were measured on an Intel Xeon G-6242 (2.80 GHz, 16 cores, 32 threads) processor using the TIME command, where the time-consuming I/O operations were disabled. Both real and user (in parentheses) times are shown in the SHRY and SUPERCELL rows. The distributed binary SUPERCELL program was used for the test.

benchmark and validation tests, the relevant comparison is mostly between SHRY ver. 1.1.0 and SUPERCELL ver. 2.0.2.

BENCHMARK TEST

$A_xPb_{1-x}Te$ is a typical benchmark system used for testing the performance of an atomic substitution program. Table 1 summarizes the results for this benchmark test. Since SHRY is implemented in Python (i.e., an interpreted language), it is intrinsically slower than the other tools implemented in compiled programming languages such as C++. Note that the actual computational time of SUPERCELL is much smaller than its CPU time because SUPERCELL is multithreaded very well. In contrast, SHRY is not multithreaded at present.

We benchmarked SHRY and SUPERCELL for a variety of systems. For the benchmark data set, we randomly chose several CIF files from the Crystallography Open Database (COD)[39] for each space group, as listed in Table S2 (Supporting Information), and we generated CIF files containing up to three atomic substitution sites. The supercell size and atomic substitution sites were randomly and repeatedly selected for each compound until the total number of symmetry-inequivalent structures becomes less than 10^9. The number of compounds used for the benchmark test is 500. Table S2 lists the compounds, unique IDs in the database, and other relevant information used for the benchmark test. Further, we report the sizes of supercells, total numbers of substituted structures, and numbers of symmetry-inequivalent structures. The computational times required to find all symmetry-inequivalent structures are summarized in the rightmost columns of Table S2. We confirmed that both the number of substitution patterns including symmetry-equivalent structures and the number of symmetry-inequivalent structures are consistent between SHRY and SUPERCELL for all compounds listed in Table S2 with the given substitution patterns. We show the computational times of SHRY on the benchmark set in Figure 3. The computational time is almost constant up to $N \sim 10^4$ in the small N region, where N refers to the number of symmetrically distinct configurations. This is because preconditioning steps, such as finding the symmetry operations of an unsubstituted structure, are the rate-determining steps of the algorithms (e.g., the actual computational times are less than 3 s for $N \leq 10^4$). Instead, the computational time in the large N region increases as N increases; however, it scales *linearly* up to $N \sim 10^9$. Since SHRY is purely implemented in Python and not yet parallelized, the actual computational time is longer than that of other packages realizing the same functionality, such as SUPERCELL shown in Figure S3. There are several possible strategies for speeding up the computation in SHRY, such as parallelizing the code or rewriting the core parts with Cython or C++, which would be promising future works.

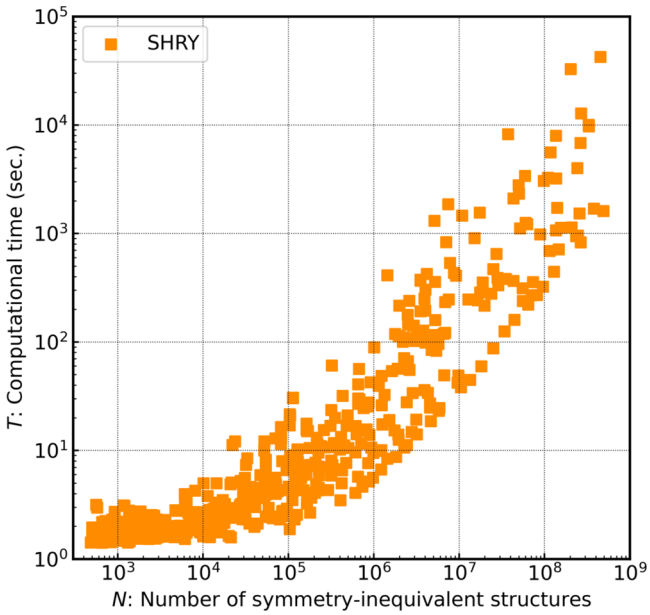


Figure 3. Computational times of SHRY on a benchmark data set. On the horizontal axis, we place the number of symmetrically distinct structures, whereas the vertical axis indicates the absolute computational time.

VALIDATION TEST

We compared not only computational times and the numbers of structures but also the redundancies and equivalencies of the generated structures among the software packages. We performed both (a) intra-software and (b) inter-software tests. (a) The *intra-software* code test checked whether each program generates zero redundant structures. Thus, for each program, we investigated whether each generated structure is symmetry inequivalent to all the others (Figure 4(a)). *Brute-force* comparisons were performed for this test (i.e., all the possible combinations were investigated). (b) The *inter-software* code test checked whether the symmetry-inequivalent structure sets generated by SHRY and SUPERCELL (or DISORDER TOOL) are identical. Thus, we built pairs of SHRY and SUPERCELL (or DISORDER TOOL) structures and confirmed that there is no excess or deficiency (Figure 4(b)).

For the validation test, we used a total of 600 compounds, which were taken from COD.[39] Table S3 lists the compounds, unique IDs in the database, and other relevant information used for the validation test. No discrepancy was found among the three codes (SHRY, SUPERCELL, and DISORDER TOOL). Both the intra-software and inter-software tests were successful for all the 600 compounds with SHRY and SUPERCELL (DISORDER TOOL was tested only for some cases; see the Supporting

https://doi.org/10.1021/acs.jcim.2c00389
J. Chem. Inf. Model. 2022, 62, 2909−2915

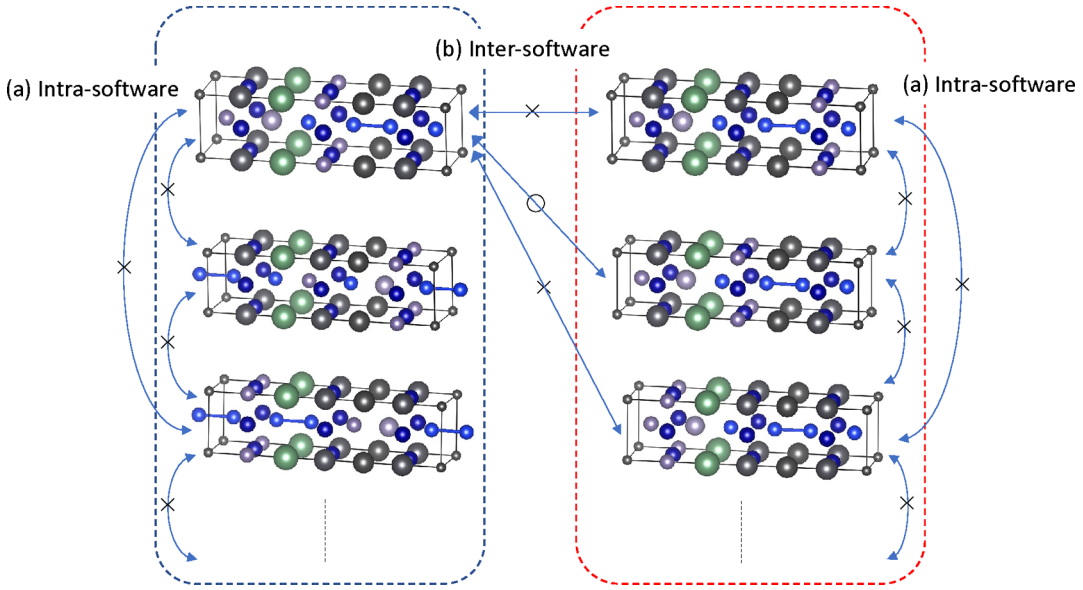


Figure 4. Schematics of (a) intra-software and (b) inter-software tests. The intra-software test checked whether each program provides zero redundant structures, where *brute-force* comparisons were performed (i.e., all the possible combinations were considered). The inter-software test checked whether the symmetry-inequivalent structure sets generated by SHRY and SUPERCELL (DISORDER TOOL) are identical; i.e., there is no excess or deficiency among them. The crystal structures are depicted using VESTA.[37]

Information). All the used CIF files are uploaded in our git repository.

CONCLUSION

We developed a Python software package, SHRY, that enables the selection of only symmetry-inequivalent structures from a vast number of candidates. This is very useful when employing the supercell approach to study a solid solution or a disordered system in an *ab initio* calculation. Although we showed several examples in this paper where some elements are substituted by other elements, the application of SHRY is not limited to them. For instance, SHRY can also be applied for studying vacancies and charge disproportionation in crystals within the supercell approach as far as symmetry-inequivalent patterns are concerned. Notice that the magnetic structures (e.g., configurations of up and down spins) cannot be studied with SHRY at present because the current implementation does not support Shubnikov groups. There are several perspectives for the future development of SHRY. One major concern is the long absolute computational time. Since SHRY is purely implemented in Python and is not multithreaded at present, the absolute computational time is not compatible with other codes for large N. A possible solution is parallelizing the code, which should be very efficient because canonical augmentation can be performed independently. Another possibility is rewriting the core parts of SHRY with much a faster language, such as C++, like other scientific packages. From the algorithmic point of view, a great advantage of the canonical augmentation algorithm used in SHRY is the use of the subobject invariant (Supporting Information), which enables us to avoid computing the time-consuming canonicity test. The present implementation, i.e., the sum of the distance matrix, is sometimes not capable of distinguishing objects belonging to different orbits on a subobject. Finding a more distinguishable subject invariant is an interesting direction for future work that must be pursued to fully exploit the advantage of the canonical augmentation algorithm, realizing a more efficient exhaustive search.

DATA AND SOFTWARE AVAILABILITY

The Python software package, SHRY, is distributed on our GITHUB repository [https://github.com/giprayogo/SHRY] under the MIT license. The package is archived in the Zenodo repository [10.5281/zenodo.5652360]. All the CIF files used for the benchmark and validation tests are also available from our GITHUB repository [https://github.com/giprayogo/SHRY], which is also archived in the Zenodo repository [10.5281/zenodo.5652360], and those of unsubstituted structures are available from the Crystallography Open Database (COD)[39] [https://www.crystallography.net/cod/].

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.2c00389.

Mathematical details of the algorithms implemented in SHRY. Details of crystal structures (CIF files) used in the benchmark and validations tests. Tables S2 and S3: Unique IDs of compounds in database, compositions, and substituted Wyckoff position(s). Measured computational times plotted in Figure 3 summarized in rightmost columns of Table S2. (PDF)

AUTHOR INFORMATION

Corresponding Authors

Genki Imam Prayogo − *School of Materials Science, JAIST, Nomi, Ishikawa 923-1292, Japan;* orcid.org/0000-0001-8365-5325; Email: g.prayogo@icloud.com

Kousuke Nakano − *International School for Advanced Studies (SISSA), 34136 Trieste, Italy; School of Information Science, JAIST, Nomi, Ishikawa 923-1292, Japan;* orcid.org/0000-0001-7756-4355; Email: kousuke_1123@icloud.com

https://doi.org/10.1021/acs.jcim.2c00389
J. Chem. Inf. Model. 2022, 62, 2909−2915

Authors

Andrea Tirelli − *International School for Advanced Studies (SISSA), 34136 Trieste, Italy*

Keishu Utimula − *School of Materials Science, JAIST, Nomi, Ishikawa 923-1292, Japan;* orcid.org/0000-0002-5461-9760

Kenta Hongo − *Research Center for Advanced Computing Infrastructure, JAIST, Nomi, Ishikawa 923-1292, Japan;* orcid.org/0000-0002-2580-0907

Ryo Maezono − *School of Information Science, JAIST, Nomi, Ishikawa 923-1292, Japan;* orcid.org/0000-0002-5875-971X

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.2c00389

Author Contributions

R.M. and K.H. initiated the idea, and K.U. implemented the preliminary code of SHRY. G.I.P. and K.N. were involved in the implementation of the production code of SHRY. A.T. contributed to the theoretical foundations of the work. K.N. and A.T. prepared the initial draft of the manuscript, and then, all authors critically revised it, provided comments, and approved the final version.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The authors acknowledge the facilities of the Research Center for Advanced Computing Infrastructure at JAIST. G.I.P. gratefully acknowledges financial support from JST SPRING (Grant Number JPMJSP2102). A.T. acknowledges financial support from MIUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2017 (Grant Number 2017BZPKSZ). K.H. is grateful for financial support from the HPCI System Research Project (Project IDs: hp210019, hp210131, and jh210045), MEXT-KAKENHI (JP16H06439, JP17K17762, JP19K05029, JP19H05169, JP21K03400, JP21H01998, and JP22H02170), and the U.S. Air Force Office of Scientific Research (Award Number FA2386-20-1-4036). R.M. is grateful for financial support from MEXT-KAKENHI (JP19H04692 and JP21K03400), the U.S. Air Force Office of Scientific Research (AFOSR-AOARD/FA2386-17-1-4049; FA2386-19-1-4015), and JSPS Bilateral Joint Projects (JPJSBP120197714). K.N. acknowledges support from JSPS Overseas Research Fellowships, a Grant-in-Aid for Early-Career Scientists (Grant Number JP21K17752), and a Grant-in-Aid for Scientific Research (C) (Grant Number JP21K03400). The authors thank Dr. Kirill Okhotnikov for help with the benchmark test using SUPERCELL.

■ REFERENCES

(1) Kageyama, H.; Hayashi, K.; Maeda, K.; Attfield, J. P.; Hiroi, Z.; Rondinelli, J. M.; Poeppelmeier, K. R. Expanding frontiers in materials chemistry and physics with multiple anions. *Nat. Commun.* **2018**, *9*, 1−15.

(2) Ichibha, T.; Prayogo, G.; Hongo, K.; Maezono, R. A new ab initio modeling scheme for the ion self-diffusion coefficient applied to the ϵ-Cu_3Sn phase of the CuSn alloy. *Phys. Chem. Chem. Phys.* **2019**, *21*, 5158−5164.

(3) Ichibha, T.; Benali, A.; Hongo, K.; Maezono, R. Ti interstitial flows giving rutile TiO_2 reoxidation process enhancement in (001) surface. *Phys. Rev. Mater.* **2019**, *3*, 125801.

(4) Bellaiche, L.; Vanderbilt, D. Virtual crystal approximation revisited: Application to dielectric and piezoelectric properties of perovskites. *Phys. Rev. B* **2000**, *61*, 7877−7882.

(5) Soven, P. Coherent-Potential Model of Substitutional Disordered Alloys. *Phys. Rev.* **1967**, *156*, 809−813.

(6) Korringa, J. On the calculation of the energy of a Bloch wave in a metal. *Physica* **1947**, *13*, 392−400.

(7) Kohn, W.; Rostoker, N. Solution of the Schrödinger equation in periodic lattices with an application to metallic lithium. *Phys. Rev.* **1954**, *94*, 1111.

(8) Nakano, K.; Hongo, K.; Maezono, R. Phonon dispersions and Fermi surfaces nesting explaining the variety of charge ordering in titanium-oxypnictides superconductors. *Sci. Rep.* **2016**, *6*, 29661.

(9) Nakano, K.; Hongo, K.; Maezono, R. Investigation into Structural Phase Transitions in Layered Titanium-Oxypnictides by a Computational Phonon Analysis. *Inorg. Chem.* **2017**, *56*, 13732−13740.

(10) Utimula, K.; Hunkao, R.; Yano, M.; Kimoto, H.; Hongo, K.; Kawaguchi, S.; Suwanna, S.; Maezono, R. Machine-Learning Clustering Technique Applied to Powder X-Ray Diffraction Patterns to Distinguish Compositions of $ThMn_{12}$-Type Alloys. *Adv. Theory Simul.* **2020**, *3*, 2000039.

(11) Wei, S.-H.; Ferreira, L. G.; Bernard, J. E.; Zunger, A. Electronic properties of random alloys: Special quasirandom structures. *Phys. Rev. B* **1990**, *42*, 9622−9649.

(12) Zunger, A.; Wei, S.-H.; Ferreira, L. G.; Bernard, J. E. Special quasirandom structures. *Phys. Rev. Lett.* **1990**, *65*, 353−356.

(13) Seko, A.; Koyama, Y.; Tanaka, I. Cluster expansion method for multicomponent systems based on optimal selection of structures for density-functional theory calculations. *Phys. Rev. B* **2009**, *80*, 165122.

(14) Yoshio, S.; Hongo, K.; Nakano, K.; Maezono, R. High-Throughput Evaluation of Discharge Profiles of Nickel Substitution in $LiNiO_2$ by Ab Initio Calculations. *J. Phys. Chem. C* **2021**, *125*, 14517−14524.

(15) Freysoldt, C.; Neugebauer, J.; Van de Walle, C. G. Fully Ab Initio Finite-Size Corrections for Charged-Defect Supercell Calculations. *Phys. Rev. Lett.* **2009**, *102*, 016402.

(16) Rurali, R.; Cartoixà, X. Theory of defects in one-dimensional systems: application to Al-catalyzed Si nanowires. *Nano Lett.* **2009**, *9*, 975−979.

(17) Murphy, S. T.; Hine, N. D. M. Anisotropic charge screening and supercell size convergence of defect formation energies. *Phys. Rev. B* **2013**, *87*, 094111.

(18) Kumagai, Y.; Oba, F. Electrostatics-based finite-size corrections for first-principles point defect calculations. *Phys. Rev. B* **2014**, *89*, 195205.

(19) D'Arco, P.; Mustapha, S.; Ferrabone, M.; Noël, Y.; Pierre, M. D. L.; Dovesi, R. Symmetry and random sampling of symmetry independent configurations for the simulation of disordered solids. *J. Phys.: Condens. Matter* **2013**, *25*, 355401.

(20) Okhotnikov, K. Comment on "Symmetry and random sampling of symmetry independent configurations for the simulation of disordered solids". *arXiv Preprint*, arXiv:1606.08062, 2016.

(21) Okhotnikov, K.; Charpentier, T.; Cadars, S. Supercell program: a combinatorial structure-generation approach for the local-level modeling of atomic substitutions and partial occupancies in crystals. *J. Cheminformatics* **2016**, *8*, 1−15.

(22) *Materials Studio*, Version 18.1.0.2017; Dassault Systèmes BIOVIA, San Diego, 2018.

(23) Hart, G. L. W.; Forcade, R. W. Algorithm for generating derivative structures. *Phys. Rev. B* **2008**, *77*, 224115.

(24) Ong, S. P.; Richards, W. D.; Jain, A.; Hautier, G.; Kocher, M.; Cholia, S.; Gunter, D.; Chevrier, V. L.; Persson, K. A.; Ceder, G. Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Comput. Mater. Sci.* **2013**, *68*, 314−319.

(25) Mustapha, S.; D'Arco, P.; De La Pierre, M.; Noël, Y.; Ferrabone, M.; Dovesi, R. On the use of symmetry in configurational

analysis for the simulation of disordered solids. *J. Condens. Matter Phys.* **2013**, *25*, 105401.
(26) Dovesi, R.; Erba, A.; Orlando, R.; Zicovich-Wilson, C. M.; Civalleri, B.; Maschio, L.; Rerat, M.; Casassa, S.; Baima, J.; Salustro, S.; Kirtman, B. Quantum-mechanical condensed matter simulations with CRYSTAL. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **2018**, *8*, No. e1360.
(27) Grau-Crespo, R.; Hamad, S.; Catlow, C. R. A.; de Leeuw, N. Symmetry-adapted configurational modelling of fractional site occupancy in solids. *J. Phys.: Condens. Matter* **2007**, *19*, 256201.
(28) Kaski, P.; Östergård, P. R. J. *Classification Algorithms for Codes and Designs (Algorithms and Computation in Mathematics)*; Springer-Verlag: Berlin, Heidelberg, 2005.
(29) Togo, A.; Tanaka, I. Spglib: a software library for crystal symmetry search. *arXiv Preprint*; arXiv 1808.01590, 2018.
(30) McKay, B. D. Isomorph-Free Exhaustive Generation. *J. Algorithms* **1998**, *26*, 306−324.
(31) Pólya, G. Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen. *Acta Math* **1937**, *68*, 145−254.
(32) Pólya, G.; Read, R. C. H. *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds*; Springer-Verlag: Berlin, Heidelberg, 1987.
(33) Read, R. C. Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. *Ann. Discrete Math.* **1978**, *2*, 107−120.
(34) Gordon, R. A.; DiSalvo, F. J. Crystal structure and magnetic susceptibility of $Ce_8Pd_{24}Sb$. *Zeitschrift für Naturforschung B* **1996**, *51*, 52−56.
(35) Bergerhoff, G.; Hundt, R.; Sievers, R.; Brown, I. The inorganic crystal structure data base. *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 66−69.
(36) Hellenbrandt, M. The inorganic crystal structure database (ICSD)—present and future. *Crystallogr. Rev.* **2004**, *10*, 17−22.
(37) Momma, K.; Izumi, F. VESTA 3 for three-dimensional visualization of crystal, volumetric and morphology data. *J. Appl. Crystallogr.* **2011**, *44*, 1272−1276.
(38) Dovesi, R.; Pascale, F.; Civalleri, B.; Doll, K.; Harrison, N. M.; Bush, I.; D'Arco, P.; Noël, Y.; Rérat, M.; Carbonnière, P.; Causà, M.; Salustro, S.; Lacivita, V.; Kirtman, B.; Ferrari, A. M.; Gentile, F. S.; Baima, J.; Ferrero, M.; Demichelis, R.; De La Pierre, M. The CRYSTAL code, 1976−2020 and beyond, a long story. *J. Chem. Phys.* **2020**, *152*, 204111.
(39) Gražulis, S.; Chateigner, D.; Downs, R. T.; Yokochi, A. F. T.; Quirós, M.; Lutterotti, L.; Manakova, E.; Butkus, J.; Moeck, P.; Le Bail, A. Crystallography Open Database − an open-access collection of crystal structures. *J. Appl. Crystallogr.* **2009**, *42*, 726−729.

https://doi.org/10.1021/acs.jcim.2c00389
J. Chem. Inf. Model. 2022, 62, 2909−2915