RandSpg: an open-source program for generating atomistic crystal structures with specific spacegroups

Patrick Avery[a], Eva Zurek[a,*]

[a]*Department of Chemistry, State University of New York at Buffalo, Buffalo, New York, 14260-3000*

Abstract

A new algorithm, RandSpg, that can be used to generate trial crystal structures with specific space groups and compositions is described. The program has been designed for systems where the atoms are independent of one another and it is therefore primarily suited towards inorganic systems. The structures that are generated adhere to user defined constraints such as: the lattice shape and size, stoichiometry, set of space groups to be generated, and factors that influence the minimum interatomic separations. In addition, the user can optionally specify if the most general Wyckoff position is to be occupied or constrain select atoms to specific Wyckoff positions. Extensive testing indicates that the algorithm is efficient and reliable. The library is lightweight, portable, dependency-free and is published under a license recognized by the Open Source Initiative. A web interface for the algorithm is publicly accessible at `http://xtalopt.openmolecules.net/randSpg/randSpg.html`. RandSpg has also been interfaced with the XtalOpt evolutionary algorithm for crystal structure prediction, and it is illustrated that the use of symmetric lattices in the first generation of randomly created individuals decreases the number of structures that need to be optimized to find the global energy minimum.

Keywords: Crystal; Crystal Structure; Structure Prediction; Computational Crystallography; Symmetry; Space Group; Wyckoff Position; Wyckoff Site; Inorganic Materials

*Corresponding author.
E-mail address: ezurek@buffalo.edu (E. Zurek)

PROGRAM SUMMARY

Program Title: RandSpg
Licensing provisions: BSD 3-clause [1]
Programming language: C++
Nature of problem: Trial structure models are required for: (i) determining the crystal structure of a compound given its powder X-ray diffraction data using the direct space method, (ii) creating the first set of individuals using *a priori* crystal structure prediction. For both of these problems the initial guess can greatly influence the success rate of the algorithm used for structure determination, especially for crystals with large unit cells. The unit cells of over 99% of inorganic crystals possess some element of symmetry, and it may be possible to obtain partial information about their crystal structures experimentally. Therefore, an algorithm that is able to create trial structures with user defined constraints including the crystal's composition, space group and unit cell parameters is desired.
Solution method: The RandSpg algorithm is able to determine every possible combination of Wyckoff positions for a given space group and composition. The algorithm randomly picks one of these combinations and adds atoms to particular Wyckoff sites wherein the Cartesian coordinates are chosen randomly such that they satisfy user-defined minimum interatomic distance constraints. In addition, the program can optionally generate crystals where user-defined atoms are placed at specific Wyckoff sites.

References:

[1] `http://opensource.org/licenses/BSD-3-Clause`

1. Introduction

Unique symmetry distributions have been observed in different classes of extended systems. Over 80% of organic crystals possess the space groups $P2_1/c$ (36.6%), $P\bar{1}$ (16.92%), $P2_12_12_1$ (11.00%), $C2/c$ (6.95%), $P2_1$ (6.35%) and $Pbca$ (4.24%). The three most frequent space groups for inorganic systems are $Pnma$ (8.25%), $P2_1/c$ (8.15%) and $Fm\bar{3}m$ (4.42%) [1], and less than 1% of inorganic crystals belong to the $P1$ space group [2]. The fact that symmetry is prevalent in extended matter implies that imposing symmetry constraints may be able to help accelerate the determination of an unknown crystal structure.

In certain situations powder diffraction using X-rays or neutrons must be employed to solve a structure because single crystals are not available. These techniques are more difficult because the diffraction peaks overlap and it may not be possible to obtain structure factor amplitudes. One way to solve structures from powder diffraction is by using "direct space" or "model building" methods [3]. These approaches use global optimization techniques, such as simulated annealing or genetic/evolutionary algorithms (GAs/EAs), to generate model structures whose calculated diffraction patterns are compared to those obtained experimentally. In some cases the crystal energy, atomic coordination and other factors may also be employed to determine a structure's fitness (likelihood of acceptance or chance of procreation).

The first step in structure determination is to find those unit cells and space groups that could potentially reproduce the experimental diffraction data. Typically a list of possibilities, along with their ranking, are obtained. These candidate solutions are employed by the global optimization method that carries out the search. In the case of molecular crystals, bond distances, angles, dihedrals and connectivity constraints are employed to reduce the number of degrees of freedom (DoF), thereby simplifying the search. Building suitable trial structures for non-molecular compounds is more difficult, however the number of DoF can be reduced by using building blocks such as coordination polyhedra. Another way to accelerate the determination of the structure of atomistic systems is by using randomly generated unit cells that adhere to the experimentally determined symmetry constraints. Towards this end Deng and Dong wrote the program SMEPOC [4, 5]. SMEPOC determines all possible Wyckoff positions that yield a particular chemical composition; these are referred to as equivalent position combination (EPC) models. SMEPOC can be thought of as an automation and generalization of EPC techniques employed by pioneering crystallographers that solved crystal structures manually [6]. **First-principles-assisted structure solution (FPASS) is**

another approach that takes X-ray diffraction data and uses it for structure determination [7]. In addition to the diffraction data, FPASS uses a genetic algorithm along with density functional theory (DFT) energies and mined information on Wyckoff site occupancies from the Inorganic Crystal Structure Database to perform structure determination.

Another instance where the generation of random structures with symmetry constraints is useful is in *a priori* crystal structure prediction (CSP) [8]. CSP is also carried out via global optimization schemes, however the fitness of a given individual can be determined without recourse to experimental information. Instead, fitness is related to the structure's energy or enthalpy as computed with a program based on empirical potentials or first-principles methods such as DFT. Of course, if experimental data is available it can be used to simplify and accelerate the search **(as in FPASS)**, however it is *not required*.

Programs for CSP typically begin by generating a "chemically sensible" random set of individuals whose geometries are optimized to the nearest local minimum, and whose energies/enthalpies are employed to determine fitness. Unfortunately, the likelihood that purely random structures will be low in energy is inversely proportional to the number of atoms in the unit cell. It has been shown that systems with very low or very high energies tend to be symmetrical [9, 10]. This suggests that imposing symmetry constraints on otherwise randomly generated structures may facilitate the exploration of low-lying regions of the potential energy surface. Some of the global optimization schemes for CSP where generation of symmetric structures has proven to be useful are random searches [11–13], particle swarm optimization (PSO) [14] and EAs/GAs [15, 16].

For molecular compounds, random crystals with specific space groups are typically generated by taking the molecule or asymmetric unit and applying symmetry operations to it ("duplicating" it) until the correct space group is obtained. This method has proven to be successful in molecular CSP [11, 12]. If prior connectivity and relative placement of atoms are already known, this dramatically decreases the number of DoF, and thus the search may be much simpler. One potential downfall of this approach, however, is that it may yield structures with a large number of formula units in the primitive cell. Because of the lack of connectivity constraints, algorithms that can be used for non-molecular (often inorganic) systems typically use different approaches, as described below.

In the *ab initio* random structure searching (AIRSS) algorithm candidate individuals are either created randomly (using symmetry constraints) or via mutations of structures that were found to be particularly stable [13]. Symmetry constraints are also available in the CALYPSO (Crystal structure AnaLYsis by PSO) program

[14]. CALYPSO generates a crystal with a specific space group by finding a set of Wyckoff positions from that space group that yield the correct composition and then adding atoms to these sites. The first set of individuals is constructed using symmetry constraints, and future structures are made by PSO. Tests on a supercell of TiO_2 showed that the global minimum, rutile, could be found much more quickly by CALYPSO when symmetry constraints were employed [17].

A particularly popular set of methods in CSP are EAs/GAs, and many programs are available [15, 18–31]. The first generation of individuals in an EA search are randomly constructed, and child structures are made via mutations of a single parent or combinations of two parents. One of the most popular EA programs, USPEX, can impose symmetry constraints on the systems comprising the first generation in two different ways [15, 16]. The "cell-splitting" technique imposes translational symmetry by replicating subcells of randomly generated coordinates. Tests on a supercell of silica, $Si_{24}O_{48}$, showed that the average number of generations required to find the coesite structure decreased from 20.35 to 14.08 when cell-splitting (and an improved version of the heredity operator) were employed [15]. A few years later a symmetry operator that enabled the generation of all of the 230 space groups via adding atoms to specific Wyckoff sites was implemented within USPEX [16]. The way in which USPEX generates symmetric crystals is by adding atoms to the most general Wyckoff position of a space group that is compatible with the unit cell composition. Atoms may also be added to more specific Wyckoff positions to achieve the right stoichiometry. If the distance between two atoms related by symmetry is smaller than a tolerance set by the user, their coordinates are averaged. It was shown that USPEX could predict the structure of the garnet pyrope ($Mg_{24}Al_{16}Si_{24}O_{96}$) with 160 atoms in the cubic unit cell when the cell parameters were fixed, and various improvements to the code, including symmetric initialization, were employed [16].

Despite the fact that a number of programs are available that can generate crystals with particular space groups, only SMEPOC is able to determine every possible combination of Wyckoff positions for a given composition. SMEPOC, however, is not available under a license that is recognized by the Open Source Initiative (OSI) [32] (eg. the Gnu Public License, GPL, or Berkley Software Distribution, BSD). In fact, AIRSS, CALYPSO and USPEX are also not distributed under OSI-approved licenses. This means that the code used within these programs is not generally available and cannot be re-used and included in other programs. Therefore, we have written RandSpg, an open-source program that can find every possible combination of Wyckoff positions for a given composition. In addition to a number of standard options (eg. volume and interatomic distance

constraints), advanced options that are not available in other programs can be set. For example, the user can decide whether or not the most general Wyckoff position should be employed, and atoms of different types can be placed on user-specified Wyckoff positions. RandSpg is available as a standalone program, and it has also been incorporated into the open source evolutionary algorithm XtalOpt[18–20]. Herein, we describe the algorithm and test its success rate in generating structures with the right space group, and its impact on evolutionary algorithm performance.

2. Algorithm details

2.1. Finding all possible combinations of Wyckoff positions

The most general Wyckoff position has the highest multiplicity, and the atoms found on it do not lie on any symmetry elements. The remaining sites are the so-called special positions, and the atoms that occupy these sites reside on symmetry elements of the cell. A Wyckoff position is unique if all of its coordinates are fixed, that is, they do not contain variables. A unique position can only be used once, whereas one that is not unique can be used an infinite number of times. Consider for example space group 63; the coordinates for its Wyckoff positions are provided in Table 1. The most general Wyckoff position is 16*h*, and 4*a* is unique while 4*c* is not.

RandSpg finds all possible combinations of Wyckoff positions for a particular composition. It begins by determining all of the possibilities for a single atom type. A doubly recursive function is employed towards this end, and a flow chart illustrating the procedure is given in Figure 1. Basically, the function tries to use every Wyckoff position (in combination with the other sites) at least once, at least twice, etc. until it cannot use it anymore. A position is considered unusable if it has a higher multiplicity than the number of atoms left, or if it is unique and is already

Multiplicity	**Wyckoff Letter**	**Site Symmetry**	**Coordinates** (0,0,0) + (1/2,1/2,0)
16	h	1	(x,y,z) (-x,-y,z+1/2) (-x,y,-z+1/2) (x,-y,-z) (-x,-y,-z) (x,y,-z+1/2) (x,-y,z+1/2) (-x,y,z)
8	g	..m	(x,y,1/4) (-x,-y,3/4) (-x,y,1/4) (x,-y,3/4)
8	f	m..	(0,y,z) (0,-y,z+1/2) (0,y,-z+1/2) (0,-y,-z)
8	e	2..	(x,0,0) (-x,0,1/2) (-x,0,0) (x,0,1/2)
8	d	-1	(1/4,1/4,0) (3/4,3/4,1/2) (3/4,1/4,1/2) (1/4,3/4,0)
4	c	m2m	(0,y,1/4) (0,-y,3/4)
4	b	2/m..	(0,1/2,0) (0,1/2,1/2)
4	a	2/m..	(0,0,0) (0,0,1/2)

Table 1: The Wyckoff positions for space group 63 (*Cmcm*) [33]. The post-perovskite structure of $MgSiO_3$ in Fig. 2 crystallizes in this space group.

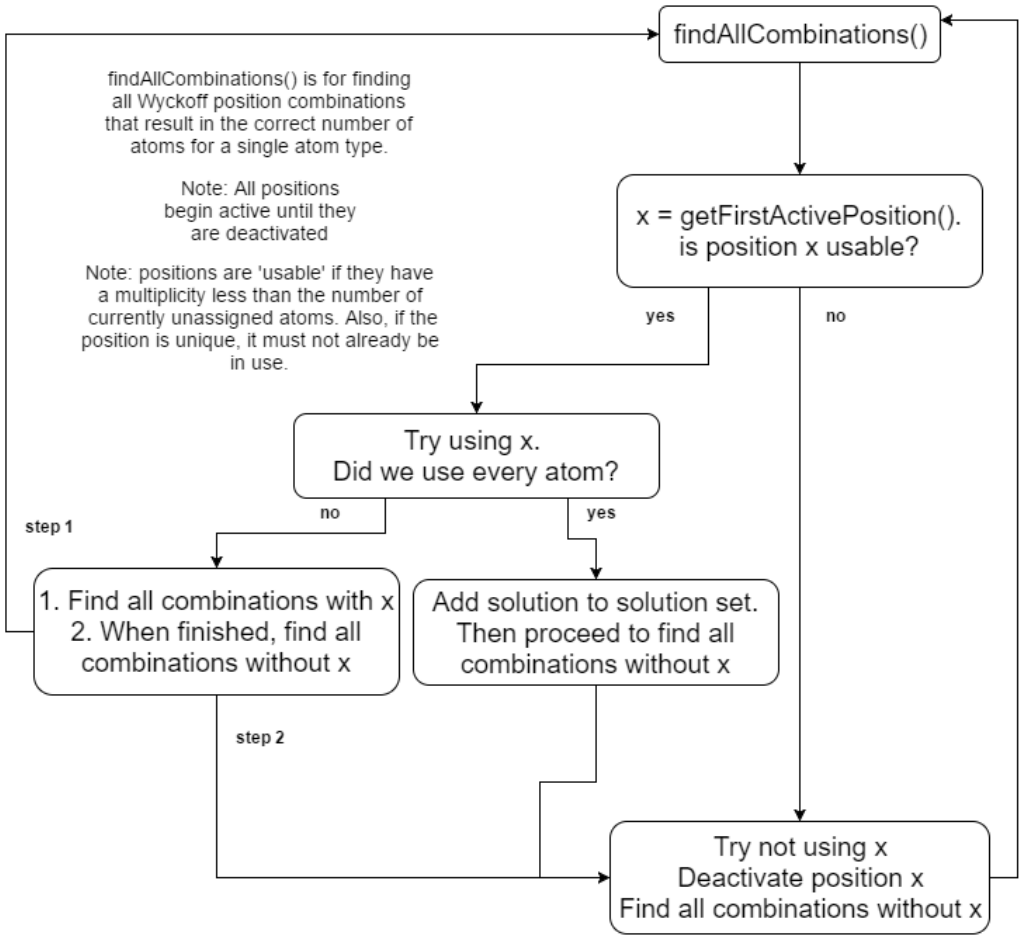

Figure 1: Flow chart for finding all possible combinations of Wyckoff positions for a single atom type.

in use. For example, in the mantle of the earth, $Mg_4Si_4O_{12}$ forms a *Cmcm* (space group 63) structure called post-perovskite, illustrated in Figure 2. From Table 1, it can be seen that all possible combinations of Wyckoff positions for O would be the following: $\{8g, 4c\}$, $\{8g, 4b\}$, $\{8g, 4a\}$, $\{8f, 4c\}$, $\{8f, 4b\}$, $\{8f, 4a\}$, $\{8e, 4c\}$, $\{8e, 4b\}$, $\{8e, 4a\}$, $\{8d, 4c\}$, $\{8d, 4b\}$, $\{8d, 4a\}$, $\{4c, 4c, 4c\}$, $\{4c, 4c, 4b\}$, $\{4c, 4c, 4a\}$, and $\{4c, 4b, 4a\}$. Note that because $4c$ is not unique, it may be re-used, but because $4a$ and $4b$ are unique, they may not be re-used.

After all combinations have been found for all of the individual atom types, a cross join is performed between the different types thereby yielding the possible

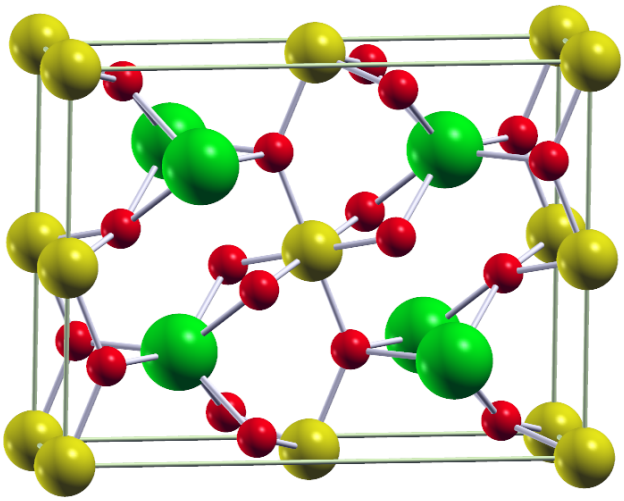

Figure 2: Structure of post-perovskite, a high-pressure phase of $MgSiO_3$. Mg atoms are green, Si atoms are yellow, and O atoms are red.

combinations of Wyckoff positions for the whole system (these are called "system possibilities" in the program). Since the individual possibilities for both Mg and Si are $\{4c\}$, $\{4b\}$, and $\{4a\}$, a cross join between their possibilities would yield: $\{Mg : 4c, Si : 4c\}$, $\{Mg : 4c, Si : 4b\}$, $\{Mg : 4c, Si : 4a\}$, $\{Mg : 4b, Si : 4c\}$, $\{Mg : 4a, Si : 4c\}$, $\{Mg : 4b, Si : 4a\}$, and $\{Mg : 4a, Si : 4b\}$. Note that because $4c$ is not unique, it may be occupied any number of times. However, since $4b$ and $4a$ are unique, they may be occupied by either Mg or Si, but not both. Thus, cross joins that result in a unique Wyckoff position being used more than once are discarded.

After the cross join is completed, all possible combinations have been found. If the user specifies Wyckoff position constraints (for example, Mg is forced to be in Wyckoff position $4b$), any combinations that do not satisfy these constraints are removed. In addition, if the user specifies that the most general Wyckoff position is to be occupied at least once, any combinations that do not employ the most general site are removed as well. For the real post-perovskite structure, the Wyckoff positions are as follows: $\{Mg : 4c, Si : 4a, O : 8f, O : 4c\}$.

For large systems, the number of possible combinations of Wyckoff positions can become exceptionally large. In order to remove this computational bottleneck from the program, sites that have the same multiplicity and uniqueness (they are both unique or not unique) are grouped together. For the combinations of Wyckoff

positions given above for O_{12} with a space group of 63, the list shortens to the following: $\{1(8g, 8f, 8e), 4c\}$, $\{1(8g, 8f, 8e), 1(4b, 4a)\}$, $\{8d, 4c\}$, $\{8d, 1(4b, 4a)\}$, $\{4c, 4c, 4c\}$, $\{4c, 4c, 1(4b, 4a)\}$, and $\{4c, 4b, 4a\}$. In this notation, $n(a, b, c...)$ means choose n of the Wyckoff positions given in parentheses. For example, $1(4b, 4a)$ means to choose either $4b$ or $4a$. For this example, the grouping reduced the number of combinations for O_{12} from 16 to 7. This method dramatically improves the speed of RandSpg – especially for large systems and space groups with numerous sites. After all of the possible Wyckoff positions are found, one set is randomly chosen and the variables in its coordinates are determined as described in Sec. 2.3.

2.2. Generating the Lattice Parameters

The lattice vectors and angles that are permissible depend upon the space group that is chosen. A simple example is that of a cubic lattice for which $a = b = c$ and $\alpha = \beta = \gamma = 90.0°$. When RandSpg generates a lattice, it chooses the values randomly within the constraints of the crystal type for the given space group and the minimum and maximum values for the lattice vectors and angles set in the input file by the user (see Sec. 3). Next, the lengths are scaled so that the volume of the crystal falls within the permissible minimum and maximum values. If this rescaling moves a lattice parameter outside of its allowed range, the crystal gets discarded and a new lattice is generated. Note that only hexagonal cells are used for trigonal systems within RandSpg. If the lattice parameters requested in the input file are not compatible with the requested space group, an error message is printed to the screen indicating that a crystal of the right shape cannot be generated with the current settings.

2.3. Adding the atoms to the lattice

Once all of the possible combinations of Wyckoff positions have been found, and the lattice has been generated, RandSpg chooses one of the combinations of Wyckoff sites (so-called "atom assignments") randomly. Values for the variables in the Wyckoff position coordinates are randomly generated, and atoms are added to the cell according to the symmetry at the given site. This procedure is repeated until all coordinates are found for all of the atoms required for the desired composition. After the addition of each atom, RandSpg checks to make sure that interatomic distance constraints were not violated. The minimum distance between two atoms is obtained by finding the sum of their radii (the default is the covalent radii in OpenBabel [34], but the user may override these defaults as described in Sec. 3). If an atom that is added to the cell is too close to the other atoms within the structure, it is removed and, if its Wyckoff site contains a variable, a new set

of random numbers is generated for it. RandSpg makes **500 × (the number of variables in the Wyckoff site)** attempts to find suitable coordinates for the atom. **Since this can be one, two or three, the maximum number of attempts will be 500, 1000, or 1500**. If it fails, a new lattice is generated, and a new set of atom assignments from the possible Wyckoff combinations is made. The number of attempts **(default of 100)** that the program makes to generate coordinates for a given space group can be set in the input file.

Adding atoms to the unit cell is certainly the bottleneck in the program, especially if the user specifies large radii and a small volume for the crystal. In addition, this procedure becomes more time consuming for larger unit cells. The reason for this is that it becomes more difficult to satisfy the minimum interatomic distance constraints. A larger number of different atom types also slows down the code: as shown in Sec. 4.2, generating acceptable coordinates for quaternary systems with up to 60 atoms in the unit cell required, on average, about 5 seconds, whereas a similar sized ternary system required, on average, 1 second.

3. Computational Details

3.1. Basic Usage

RandSpg is written in C++ and does not employ any external dependencies, allowing for ease of use and compilation. Compiling RandSpg requires CMake and a C++ compiler with C++0x capabilities. The library was built and tested using g++4.8.4 on Linux and MSVC 2013 on Windows. Instructions for compiling the program are provided in the README file.

A sample input file is illustrated in Fig. 3. This file is also provided in the root directory of the program package. The composition and space groups to be generated are the only input required for RandSpg to run, all other parameters will employ default values unless otherwise specified. For the program to generate chemically sensible structures it is important to choose reasonable values for the allowed minimum and maximum volumes. Moreover, an appropriate choice for the atomic radii and scaling factors, which are important for determining the minimum interatomic distances, must be provided. If these values are not chosen judiciously, atoms may overlap or be too far apart to form bonds. The minimum and maximum volumes are bounded by the range of lattice parameters. If the user specifies a range outside of these bounds an error message is printed. The default radii are the covalent radii employed in OpenBabel [34], and the default scaling factor is 1.0. In certain situations, such as conditions of extreme pressure, it may be desirable to set the scaling factor to a value less than unity. Providing the

```
# Anything to the right of a hash is a comment
# Composition is set by atomic symbols followed by number as such:
composition            = Mg4Al2

# The spacegroups to generate are set as follows (hyphens and commas work)
spacegroups            = 1-8, 10, 25, 28, 30-32

# lattice mins and maxes set constraints on the lattice to be generated.
# Distances are in Angstroms and angles are in degrees.
#                           a,    b,    c, alpha,  beta, gamma
latticeMins            =  3.0,  3.0,  3.0,  60.0,  60.0,  60.0
latticeMaxes           = 10.0, 10.0, 10.0, 120.0, 120.0, 120.0

# minVolume and maxVolume specify constraints on the volume in Angstroms
# If a crystal's volume is not within this range, it will be rescaled so that
# it is. If the minVolume tag is removed or specified to be -1, there will be no
# minVolume. The same goes for maxVolume.
minVolume              = 450
maxVolume              = 500

# numOfEachSpgToGenerate tells us how many crystals of each spg to generate
numOfEachSpgToGenerate = 3

# For advanced users: by default, the program will only generate a spacegroup
# for a crystal if it can use the most general Wyckoff position at least
# once. This is because the spacegroup is not guaranteed if the most
# general Wyckoff position is not used at least once. The user, however,
# may turn off that option here by setting it to false. If this is the case,
# more spacegroups may be generated for a particular composition, but there is
# a chance that it won't be the correct spacegroup.
#forceMostGeneralWyckPos = false

# For advanced users: this allows us to force an element to be assigned
# to a specific Wyckoff position. If you wish to force an element to be in
# the same Wyckoff position multiple times, just repeat the tag multiple times
# (i. e. add 'forceWyckPos Mg = a' on as many lines as you want to force the
# Wyckoff position).
#forceWyckPos Mg        = a

# We can set minimum radii (Angstroms) for individual atoms
#setRadius Ti           = 0.5

# or for all atoms. A min radius for an individual atom gets precedence over
# this one. If the default min radius of the atom is less than this input
# value, then the min radius of that atom is set to be this value.
# Units are in Angstroms.
setMinRadii            = 0.3

# This scaling factor will scale all radii that were not explicity set.
# The new radii are equal to radius * scalingFactor
# This is useful, for example, when changing the pressure of a crystal.
scalingFactor          = 0.5

# This sets the maximum number of attempts to generate any given spacegroup
maxAttempts            = 100

# This sets the output directory
outputDir              = spgGenOut

# Verbosity indicates how much output to generate in the log file
# 'n' is no output, 'r' is regular output, and 'v' is verbose output
verbosity              = r
```

Figure 3: Sample input file for RandSpg. This file is available in the program package.

ranges for the lattice parameters may be useful, especially if experimental data is available. The defaults are 3-10 Å for the lattice lengths and 60-120° for the angles. The user can specify how many structures should be generated for each space group; by default this value is one.

***** The duplicated section (mentioned by the reviewer) was removed from here. *****

By default, RandSpg always uses the most general Wyckoff position (the position with the highest multiplicity) at least once because an incorrect space group may be generated otherwise. The reason for this is that some of the more specific Wyckoff positions are identical for different space groups. The user may override this default by changing the tag "forceMostGeneralWyckPos" to false, but there is a chance that the structures that are generated will not have the desired space group. It is also possible to force certain elements to occupy user-defined Wyckoff positions.

If any problems are encountered during the program execution, an error message will be printed to stdout or stderr. The log file provides the values of the parameters that were employed, and states if they were the default settings. Output files are named as <composition>_<spg>-<index> and saved in a directory whose name may be specified by the user. **The output files are in the POSCAR format that is used by the Vienna *ab initio* Simulation Package, VASP [35]. If one wishes to use a different file format, there are many programs available that can be used to perform a file conversion including OpenBabel[36], Aconvasp[37], ASE[38], and Pymatgen[39].**

3.2. Integration into a Program

For implementing these functions into a C++ program, one simply needs to create a randSpgInput object and use it to call RandSpg::randSpgCrystal(). The function returns a crystal object **which contains all relevant crystal information and several relevant functions (including one to write to a POSCAR file)**. The options that can be employed in randSpgInput are described in the README file of RandSpg or in the header file include/randSpg.h. The README file also contains instructions on how to create a randSpgInput object and how to call RandSpg::randSpgCrystal(). Information for some of the useful functions in the Crystal class is provided in the README file as well.

A web interface to RandSpg is publicly available at
http://xtalopt.openmolecules.net/randSpg/randSpg.html.

3.3. Empirical Potentials

In Sec. 4.3 we report the results of tests that were carried out to determine the success rate of generating the global energy minimum of titanium dioxide completely randomly as compared to when RandSpg was employed. Further tests were used to gauge if the performance of the evolutionary algorithm XtalOpt would improve if symmetric structures created with RandSpg were used in the first random generation of individuals. A 16 formula unit supercell of TiO_2 was chosen for these tests, and the geometry optimizations were carried out using the GULP optimizer [40–42] and the empirical parameter set previously employed in Ref. [18]. This system was chosen as a test case based on the availability of the potential and because it has been used to benchmark the performance of XtalOpt and other programs for crystal structure prediction [16–18, 43–45]. The global minimum structure of TiO_2 is rutile with 6 atoms per unit cell and the computational protocol employed yielded a volume of 10.40 $Å^3$/atom, and minimum interatomic distances of 2.49, 3.52, and 1.93 Å for the O-O, Ti-Ti and Ti-O contacts.

4. Results

4.1. Accuracy

To test the reliability of the program we created ten crystal lattices for every space group ensuring that the most general Wyckoff position was employed. In order to guarantee that a crystal could be generated for the given crystal type the following compositions were used for triclinic, monoclinic, orthorhombic, tetragonal, trigonal, hexagonal and cubic lattices: Ti_2O_4, Ti_4O_8, $Ti_{16}O_{32}$, $Ti_{24}O_{36}$, $Ti_{24}O_{36}$ and $Ti_{96}O_{192}$, respectively. The input files for these tests may be found in the git repository [46] under samples/accuracyTests. Next, the space groups of the crystals were determined using the FINDSYM algorithm [47, 48] and a tolerance of 0.0001 Å. For the 2300 structures that were made the success rate was 100%. In addition, the same test was performed for cells generated wherein the requirement to use the most general Wyckoff position at least once was removed. Using a tolerance of 0.0001 Å, the success rate was 98.2%. That is, 2,258 lattices that were created with RandSpg were identified to have the correct space group via FINDSYM. These high success rates illustrate that RandSpg can reliably make random atomistic crystals with specific space groups. In general, the lower the multiplicity of a particular Wyckoff position, the greater the likelihood that it is common to a number of space groups. Thus, solely using more specific (lower multiplicity) Wyckoff positions increases the chances of generating structures with the wrong

space group. However, as the above tests illustrate, employing the most general Wyckoff position is sufficient to ensure that the correct space group is created.

4.2. Performance

Tests were performed using a single processor on a 1.7 GHz AMD Athlon(tm) X4 860K Quad Core Processor. The code was optimized using the Callgrind tool of Valgrind [49]. Tests revealed that adding atoms to the lattice was the main bottleneck of the program. In this step RandSpg randomly chooses the variables in the Wyckoff positions, checks to determine if the minimum interatomic distances constraints are satisfied, and chooses another set of variables if they are not. Because decreasing the atomic radii and increasing the volume increases the minimum interatomic distances that are allowed, these parameters affect the amount of time it takes to generate acceptable coordinates for a particular system and space group. However, care should be used when choosing these parameters because volumes that are too large and interatomic distances that are too small will yield structures that are chemically not sensible. Clearly, a greater amount of atoms or atom types that comprise the unit cell, and space groups that have a large number of Wyckoff positions increases the amount of time required to generate a lattice.

Figs. 4 and 5 illustrate the results of a few of the speed tests that were performed on structures with the $(TiO_2)_x$ stoichiometry. The scaling factor and minimum radius employed yielded atomic radii of 0.33 Å for oxygen and 0.8 Å for Ti. The range of allowed volumes was $\left(10 \pm \frac{10}{n}\right)$ Å^3/atom where n is the number of atoms. The inputs for these tests may be found in the program package under samples/speedTests. The results shown in Fig. 4 were obtained by generating ten structures for each space group and averaging the results, whereas each data point in Fig. 5 was obtained for an average of one hundred individuals. Note that instances where the algorithm failed to create a set of coordinates were not included in these averages because this usually happened when no combination of Wyckoff positions for the space group could satisfy the requested composition. When these failures occurred they were fast because RandSpg first determines if the stoichiometry/space group combination is possible prior to attempting to generate coordinates.

Fig. 4 provides the average time it took to generate a valid crystal with the $(TiO_2)_x$ stoichiometry vs. the total number of atoms in the cell, $3x$. Because some combinations of space group and composition were impossible (eg. because each Wyckoff position in space group 8, *Cm*, has an even multiplicity it is impossible for a structure with this symmetry to contain an odd number of atoms), and some

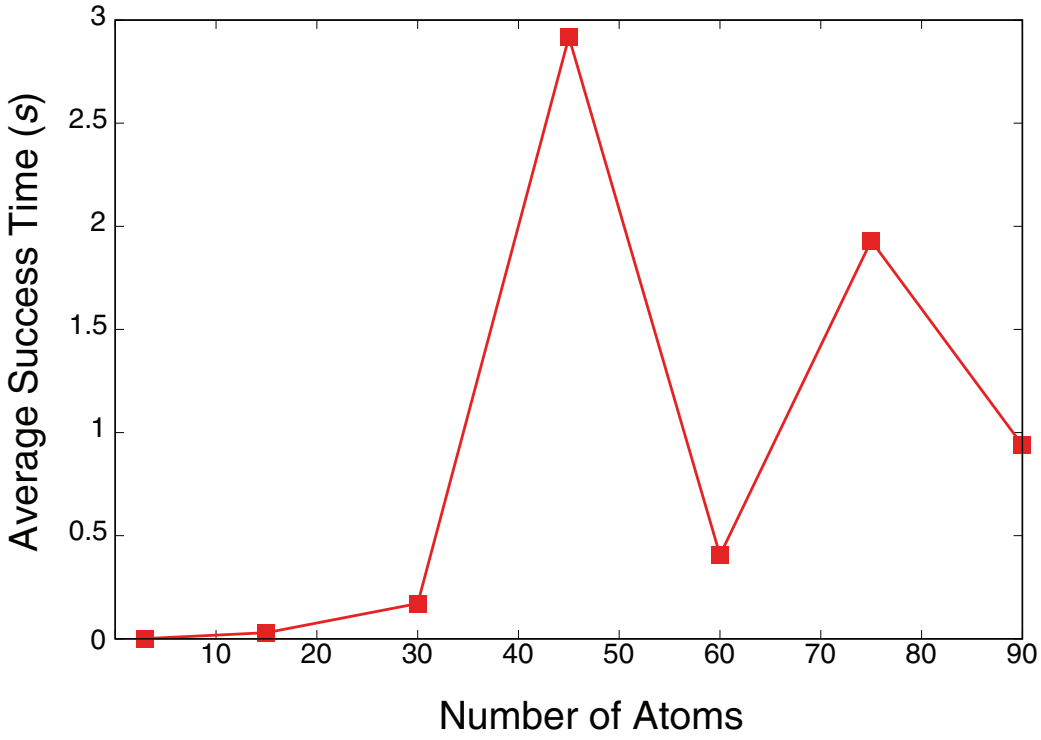

Figure 4: Average time required for generating a structure with RandSpg for the $(TiO_2)_x$ stoichiometry vs. the total number of atoms in the unit cell (3, 15, 30, 45, 60, 75, and 90). The data is provided by the points and the lines are a guide to the eye.

space groups require a longer time to generate acceptable coordinates, one cannot make a direct comparison between each set of runs. In general the time required to make a structure increases with increasing number of atoms in the unit cell. Note that for cells with a large odd number of atoms, such as 45 and 75, the time was longer than for systems that were similar in size but had an even number of atoms. The reason for this is that it is easiest to generate cells with fewer Wyckoff positions wherein each Wyckoff position has a small multiplicity. And, the Wyckoff positions in these types of "simple" space groups tend to have even-numbered multiplicities, so they can therefore not accommodate an odd number of atoms. For example, space group 8 has two Wyckoff positions with multiplicities of two and four. A few other "simple" space groups that have three or fewer Wyckoff positions with multiplicities of two and four are 4, 5, 7, and 9. This is a major factor that decreases the average success time of even-numbered cells (30, 60, and 90).

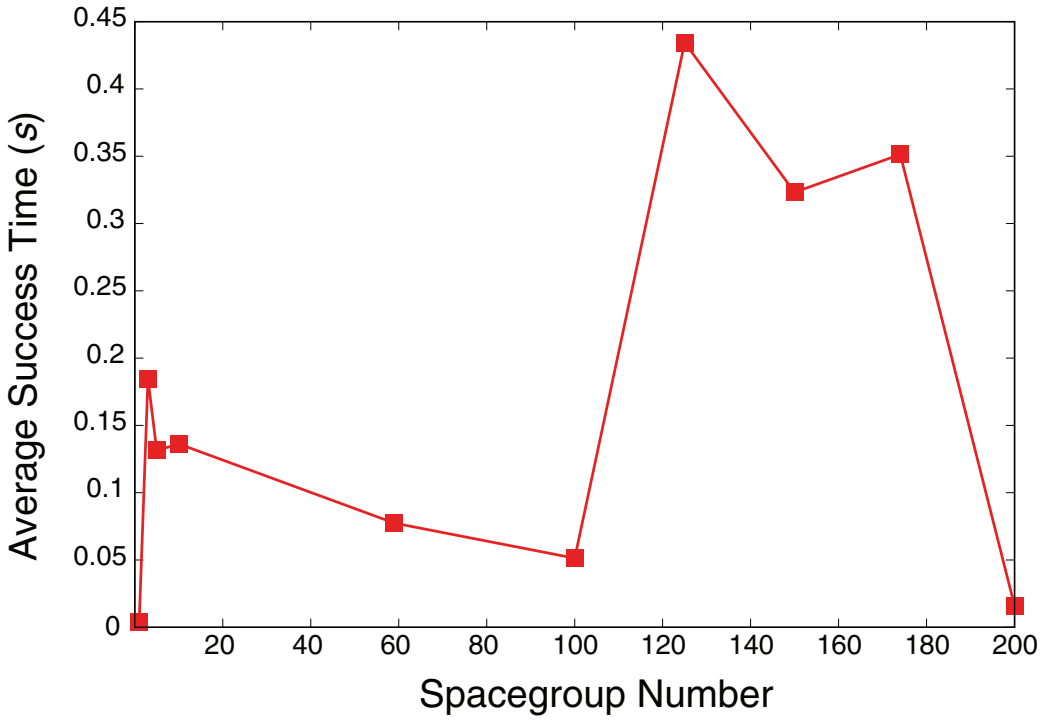

Figure 5: Average time required for generating a structure with RandSpg for the $Ti_{30}O_{60}$ stoichiometry vs. the space group number. The space groups tested were 1 ($P1$), 3 ($P2$), 5 ($C2$), 10 ($P2/m$), 59 ($Pmmn$), 100 ($P4bm$), 125 ($P4/nbm$), 150 ($P321$)), 174 ($P-6$), and 200 ($Pm-3$). The data is provided by the points and the lines are a guide to the eye.

In Fig. 5 the average success times are provided for particular space groups and the $(TiO_2)_{30}$ stoichiometry. It can be clearly seen that there is no consistent trend with the space group number. This is also evident in the fact that it took less than half a second to create structures for the space groups tested in Fig. 5, whereas the average over all possible space groups for the 90 atom system in Fig. 4 is closer to one second. Importantly, on average it took 3 seconds or less to create binary structures with up to 90 atoms in the unit cell. This suggests that the generation of symmetric structures in the first set of individuals in CSP will not be a bottleneck in the search.

Further tests were carried out to deduce how the number of atom types influences the time required to create an individual. All of the cells contained a total of 60 atoms and up to four different types. The stoichiometries employed were Ti_{60}, $Ti_{30}O_{30}$, $Ti_{20}Mg_{20}O_{20}$, and $Ti_{15}Mg_{15}Si_{15}O_{15}$. The volumes ranged from 790

to 810 Å^3, the scaling factor was 0.5, and the default radii were employed. Ten structures of every possible space group were generated, and the results were averaged. Fig. 6 illustrates that the average time required to make a structure was one second or less for the ternary, binary and single component systems. However, for $Ti_{15}Mg_{15}Si_{15}O_{15}$ the average time rose to five seconds suggesting that the algorithm may scale exponentially with the number of atom types comprising the cell. Thus, it might be prohibitively computationally expensive to construct extremely large unit cells with many different atom types using RandSpg in combination with CSP. However, in many situations this is unlikely to be necessary; for example most CSP searches employ on the order of 20-50 randomly generated structures for the first set of individuals. In addition, the structural optimizations will likely be the bottleneck in searches on large and complicated unit cells, even if empirical potentials are employed.

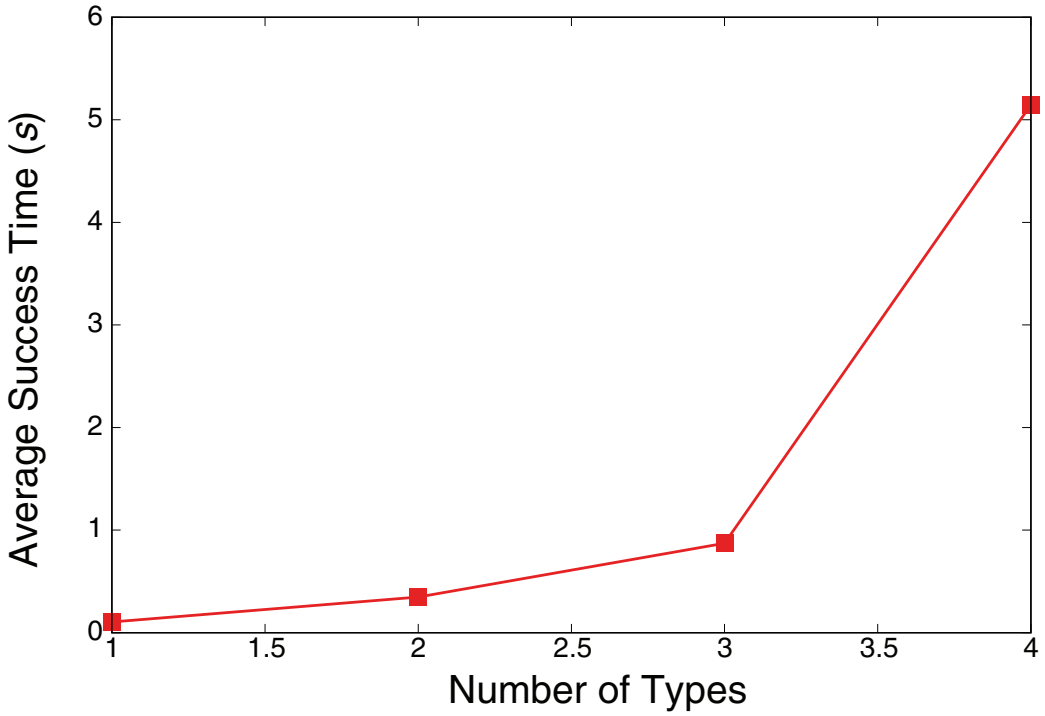

Figure 6: Average time required to generate structures with RandSpg for a monoatomic (Ti_{60}), binary ($Ti_{30}O_{30}$), ternary ($Ti_{20}Mg_{20}O_{20}$) and quaternary ($Ti_{15}Mg_{15}Si_{15}O_{15}$) system. The data is provided by the points and the lines are a guide to the eye.

4.3. RandSpg *and Crystal Structure Prediction*

Tests were carried out to determine if the use of randomly generated symmetric structures with RandSpg improved the efficiency of computational CSP. Towards this end 3250 crystalline lattices containing 16 TiO_2 units were made with and without RandSpg and optimized using GULP as discussed in Sec. 3.3. Fig. 7 illustrates the distribution of enthalpies obtained. In the set of structures that were made completely randomly only 2 individuals (0.062%) yielded the rutile supercell. In a previous study we found that 0.070% of the lattices obtained in such a manner corresponded to rutile [18], suggesting that our margin of error is less than 0.01%. When RandSpg as implemented within XtalOpt was employed and the space groups were chosen randomly, 58 of the lattices that were generated (1.78%) yielded the rutile structure. This is a significant improvement: for completely random structures over 1600 are required to have some assurance the rutile supercell is created, whereas this number drops to around 56 when RandSpg was employed.

The distribution of enthalpies for the structures that were created randomly corresponded roughly to a Gaussian centered at around -622 eV with a spread of -636.8 to -615.4 eV and a standard deviation of 5.3 eV (the enthalpy of the rutile supercell was calculated to be -636.8 eV). When RandSpg was employed the range of enthalpies was significantly larger, -636.8 to +58.0 eV (standard deviation of 98.3 eV), and even the data obtained for enthalpies lower than -615 eV (see Fig. 7(c)) deviated significantly from the Gaussian distribution because of the large number of low enthalpy structures that were generated. Symmetric structures tend to either be very stable or very unstable [9, 10], and this is the reason for the much larger range of enthalpies obtained with RandSpg. In fact, the average enthalpy of the purely random structures was -625.5 eV whereas for those made with RandSpg this number increased to -563.8 eV. This suggests that the first set of individuals should be larger when using symmetry constraints in CSP. **Alternatively, if generating a larger first set of individuals is not possible, the user could specify a larger number of structures with $P1$ symmetry.**

Investigation of the unstable structures showed that they had high symmetry coupled with numerous oxygen atoms close to each other. The potential employed uses a Coulombic interaction term with charges of -1.098e for oxygen and 2.196e for titanium. Such clusters of oxygen atoms are unfavorable due to electrostatic repulsion, but the only way they can move apart from each other while still keeping the high symmetry of the lattice is to increase the volume. Indeed, all of the optimized high energy structures had unreasonably large volumes. To overcome this problem, one could turn off

symmetry constraints in the external code used for geometry optimization, or move a few of the atoms a small distance to break the symmetry.

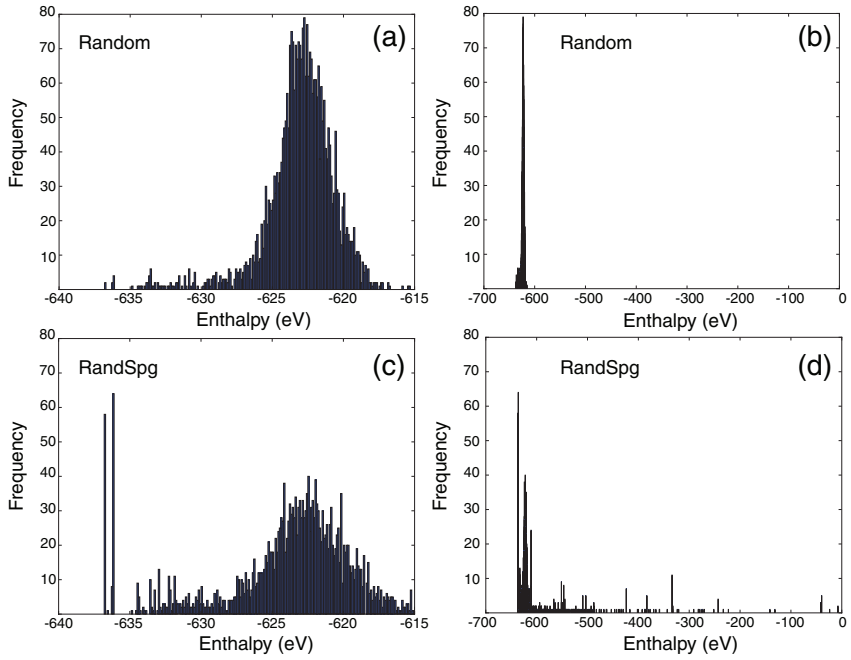

Figure 7: Distribution of the enthalpies of 3250 generated crystals with the stoichiometry $Ti_{16}O_{32}$. The lattices were generated (a,b) randomly using XtalOpt and (c,d) using RandSpg as implemented within XtalOpt and then optimized with GULP. The same data is plotted in (a/b) and (c/d) except a larger enthalpy range is employed to highlight that in (a/b) structures with enthalpies larger than -615 eV were not located, whereas (c/d) generated many systems with enthalpies less negative than -615 eV.

When the cell parameters and minimum interatomic distances between pairs of atoms were fixed to match those found within rutile, a success rate of 23.8% was obtained. This clearly demonstrates that using the correct structural parameters and appropriate settings for the interatomic distances can greatly speed up a structure search.

The CALYPSO program can also be used to create random structures with symmetry constraints. A similar test wherein 3250 structures were generated for a cell containing 16 TiO_2 units was performed [17]. The success rate for locating rutile was ~6.2%. **Unfortunately, since the set of constraints that CALYPSO used is not fully known, a direct comparison between RandSpg and CALYPSO is not possible. A discrepancy in the success rate between the**

two codes may, however, arise because RandSpg enumerates Wyckoff positions differently than CALYPSO since CALYPSO does not enumerate every possible combination.

which is substantially larger than what was found here. The higher success rate is likely due to two factors. First of all, within CALYPSO the user can specify minimum interatomic distances between every set of atoms, and the tests were carried out using distances that matched those found within rutile. The RandSpg tests carried out here, on the other hand, employed the default radii used by OpenBabel since these would be the values employed given no information about the system. Secondly, the way in which Wyckoff positions are selected by CALYPSO is different than within RandSpg, and this is also likely to affect the results.

Further tests were carried out to determine if using RandSpg to generate the first set of random individuals within an evolutionary structure search could increase the success rate and/or how quickly the global minimum structure is found. We employed the EA XtalOpt [18] version R9 [20] with the 16 formula unit supercell of rutile as the target structure. Previous tests using an earlier version of XtalOpt showed that for 100 searches, the target structure was always found when 280 individuals or less were made [45].

Each of the following values were obtained by averaging the results of 20 runs, and within each run the rutile structure was found when 350 or fewer lattices were optimized (only one run required 346 individuals to locate rutile, the other searches found it in 226 structures or less). For an initial population of 20, it took on average 107.7 optimized crystals using random generation and 72.3 optimized crystals using RandSpg to find rutile. For an initial population of 50, it took on average 141.0 optimized crystals using random generation and 63.2 optimized crystals using RandSpg to find rutile. Thus, it can be seen that RandSpg significantly improves the efficiency of the EA. Note that when the first set of individuals were generated purely randomly, increasing the initial population from 20 to 50 increased the average number of individuals that were required to find rutile by about 30 as well. This illustrates that increasing the number of randomly generated structures did not improve the efficiency of the algorithm, and the reason for this likely stems from the fact that the enthalpies of the generated structures assumed a Gaussian-like distribution as illustrated in Fig. 7(a). However, when RandSpg was employed the efficiency improved using a larger initial population. The reason for this is that RandSpg has the propensity to create many more low (and high) enthalpy structures as compared with pure random generation, as illustrated in Fig. 7(d). We therefore recommend that larger initial populations are

employed when RandSpg is used within an evolutionary structure search. **If this is not possible, we recommend that the user specifies a set number of $P1$ symmetry structures to be generated or disables symmetry in the external code used for geometric relaxation.**

5. Conclusion

The RandSpg program can determine all possible combinations of Wyckoff positions that are allowed for a particular space group and chemical composition. Tests show that RandSpg had a 100% success rate in generating coordinates for crystals with a particular space group when the most general Wyckoff position was employed at least once. If the requirement to use the most general site was removed, a 98.2% success rate was achieved. The crystals that are generated by RandSpg are subject to a number of user defined constraints. This includes minimum and maximum values for the lattice lengths and angles, and unit cell volume. In addition, the user can specify if the most general Wyckoff position is to be employed at least once and force atoms of different types to occupy specific Wyckoff positions. The minimum interatomic distances allowed between atoms can also be set.

RandSpg has been interfaced with the XtalOpt [18–20] evolutionary algorithm for *a priori* crystal structure prediction. Tests showed that when the initial generation of individuals was created using RandSpg, the evolutionary algorithm was able to more quickly locate the global minimum structure as compared to when symmetric constraints were not enforced. In the future RandSpg may also prove to be useful in structure determination from powder diffraction data via direct space methods.

RandSpg is written in C++ and it is published under the open source 3-clause BSD license [50]. The program can be run independently or within XtalOpt. In addition, a web interface is available on the internet at
http://xtalopt.openmolecules.net/randSpg/randSpg.html.

Acknowledgements

The authors thank Zack Falls for fruitful discussion regarding space groups and Wyckoff positions. We acknowledge the NSF (DMR-1005413) and the ONR (N000141612583) for financial support and the Center for Computational Research (CCR) at SUNY Buffalo for computational support.

References

[1] H. B. Werner and D. Kassner, "The perils of cc: Comparing the frequencies of falsely assigned space groups with their general population," *Acta Cryst. B*, vol. 48, pp. 356–369, 1992.

[2] V. S. Urusov and T. N. Nadezhina, "Frequency distribution and selection of space groups in inorganic crystal chemistry," *J. Struct. Chem.*, vol. 50, pp. S22–S37, 2009.

[3] R. Černý and V. Favre-Nicolin, "Direct space methods of structure determination from powder diffraction: principles, guidelines and perspectives," *Z. Kristallogr.*, vol. 222, pp. 105–113, 2007.

[4] X. D. Deng and C. Dong, "Smepoc - a computer program for the automatic generation of trial structural models for inorganic compounds with symmetry restriction," *J. Appl. Crystallogr.*, vol. 42, pp. 953–958, 2008.

[5] X. D. Deng and C. Dong, "Epcryst: a computer program for solving crystal structures from powder diffraction data," *J. Appl. Crystallogr.*, vol. 44, pp. 230–237, 2011.

[6] J. M. Reddy, A. R. Storm, and K. Knox, "The crystal structure of V2Ga5," *Z. Kristallogr.*, vol. 121, pp. 441–448, 1965.

[7] B. Meredig and C. Wolverton, "A hybrid computational-experimental approach for automated crystal structure solution," *Nat. Mater.*, vol. 12, pp. 123–127, 2012.

[8] E. Zurek, "Discovering new materials via **a priori** crystal structure prediction," in *Reviews in Computational Chemistry* (K. B. Lipkowitz, ed.), vol. 29, pp. 274–326, Hoboken, New Jersey: John Wiley & Sons, Inc., 2016.

[9] D. J. Wales, "Symmetry, near-symmetry and energetics," *Chem. Phys. Lett.*, vol. 285, pp. 330–336, 1998.

[10] D. J. Wales, "Erratum to 'symmetry, near-symmetry and energetics'," *Chem. Phys. Lett.*, vol. 294, p. 262, 1998.

[11] A. Asmadi, J. Kendrick, and F. J. J. Leusen, "Crystal structure prediction and isostructurality of three small organic halogen compounds," *Phys. Chem. Chem. Phys.*, vol. 12, pp. 8571–8579, 2010.

[12] A. Gavezzotti, "Polymorphism of 7-dimethylaminocydopenta[c]coumarin: Packing analysis and generation of trial crystal structures," *Acta Cryst. B*, vol. 52, pp. 201–208, 1996.

[13] C. J. Pickard and R. J. Needs, "Ab initio random structure searching," *J. Phys.: Condens. Matter*, vol. 23, pp. 053201 (1–23), 2011.

[14] Y. Wang, J. Lv, L. Zhu, and Y. Ma, "Crystal structure prediction via particle–swarm optimization," *Phys. Rev. B*, vol. 82, pp. 094116 (1–8), 2010.

[15] A. O. Lyakhov, A. R. Oganov, and M. Valle, "How to predict very large and complex crystal structures," *Comp. Phys. Comm.*, vol. 181, pp. 1623–1632, 2010.

[16] A. O. Lyakhov, A. R. Oganov, H. T. Stokes, and Q. Zhu, "New developments in evolutionary structure prediction algorithm uspex," *Comp. Phys. Comm.*, vol. 184, pp. 1172–1182, 2013.

[17] Y. Wang, M. Miao, J. Lv, L. Zhu, K. Yin, H. Liu, and Y. Ma, "An effective structure prediction method for layered materials based on 2d particle swarm optimization algorithm," *J. Chem. Phys.*, vol. 137, pp. 224108 (1–6), 2012.

[18] D. C. Lonie and E. Zurek, "Xtalopt: An open-source evolutionary algorithm for crystal structure prediction," *Comput. Phys. Commun.*, vol. 182, pp. 372–387, 2011.

[19] D. C. Lonie and E. Zurek, "New version announcement: Xtalopt version r7: An open-source evolutionary algorithm for crystal structure prediction," *Comput. Phys. Commun.*, vol. 182, pp. 2305–2306, 2011.

[20] Z. Falls, D. C. Lonie, P. Avery, A. Shamp, and E. Zurek, "Xtalopt version r9: An open-source evolutionary algorithm for crystal structure prediction," *Comp. Phys. Commun.*, vol. 199, pp. 178–179, 2016.

[21] C. W. Glass, A. R. Oganov, and N. Hansen, "Uspex—evolutionary crystal structure prediction," *Comp. Phys. Comm.*, vol. 175, pp. 713–720, 2006.

[22] A. R. Oganov, A. O. Lyakhov, and M. Valle, "How evolutionary crystal structure prediction works — and why," *Acc. Chem. Res.*, vol. 44, pp. 227–237, 2011.

[23] A. N. Kolmogorov, S. Shah, E. R. Margine, A. F. Bialon, T. Hammerschmidt, and R. Drautz, "New superconducting and semiconducting Fe-B compounds predicted with an ab initio evolutionary search," *Phys. Rev. Lett.*, vol. 105, pp. 217003 (1–4), 2010.

[24] S. Bahmann and J. Kortus, "Evo - evolutionary algorithm for crystal structore prediction," *Comp. Phys. Comm.*, vol. 184, pp. 1618–1625, 2013.

[25] W. W. Tipton, C. R. Bealing, K. Mathew, and R. Hennig, "Structures, phase stabilities, and electrical potentials of li-si battery anode materials," *Phys. Rev. B*, vol. 87, p. 184114, 2013.

[26] W. W. Tipton and R. Hennig, "A grand canonical genetic algorithm for the prediction of multi-component phase diagrams and testing of empirical potentials," *J. Phys.: Condens. Matter*, vol. 25, pp. 495401 (1–14), 2013.

[27] S. Q. Wu, M. Ji, C. Z. Wang, M. C. Nguyen, X. Zhao, K. Umemoto, R. M. Wentzcovitch, and K. M. Ho, "An adaptive genetic algorithm for crystal structure prediction," *J. Phys.: Condens. Matter*, vol. 26, pp. 035402 (1–6), 2014.

[28] G. Trimarchi and A. Zunger, "Global space-group optimization problem: Finding the stablest crystal structure without constraints," *Phys. Rev. B.*, vol. 75, pp. 104113 (1–8), 2007.

[29] M. d'Avezac and A. Zunger, "Identifying the minimum–energy atomic configuration on a lattice: Lamarckian twist on darwinian evolution," *Phys. Rev. B.*, vol. 78, pp. 064102 (1–15), 2008.

[30] N. L. Abraham and M. I. J. Probert, "A periodic genetic algorithm with real-space representation for crystal structure and polymorph prediction," *Phys. Rev. B.*, vol. 73, pp. 224104 (1–6), 2006.

[31] A. Fadda and G. Fadda, "An evolutionary algorithm for the prediction of crystal structures," *Phys. Rev. B*, vol. 82, pp. 104105 (1–8), 2010.

[32] https://opensource.org/.

[33] http://cryst.ehu.es/.

[34] http://openbabel.org.

[35] G. Kresse and J. Hafner, "Ab initio molecular dynamics for liquid metals," *Phys. Rev. B*, vol. 47, pp. R558–R561, 1993.

[36] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison, "Open babel: an open chemical toolbox," *J. Cheminf.*, vol. 3, pp. 1–14, 2011.

[37] S. Curtarolo, W. Setyawan, G. L. W. Hart, M. Jahnatek, R. V. Chepulskii, R. H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M. Mehl, H. T. Stokes, D. O. Demchenko, and D. Morgan, "Aflow: an automatic framework for high-throughput materials discovery," *Comput. Mater. Sci.*, vol. 58, pp. 218–226, 2012.

[38] S. R. Bahn and K. W. Jacobsen, "An object-oriented scripting interface to a legacy electronic structure code," *Comput. Sci. Eng.*, vol. 4, pp. 56–66, 2002.

[39] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. Chevrier, K. A. Persson, and G. Ceder, "Python materials genomics (pymatgen) : A robust, open-source python library for materials analysis," *Comput. Mater. Sci.*, vol. 68, pp. 314–319, 2013.

[40] J. D. Gale, "Empirical potential derivation for ionic materials," *Philos. Mag. B*, vol. 73, pp. 3–19, 1996.

[41] J. D. Gale and A. L. Rohl, "The general utility lattice program (gulp)," *Mol. Simul.*, vol. 29, pp. 291–341, 2003.

[42] J. D. Gale, "GULP: a computer program for the symmetry-adapted simulation of solids," *J. Chem. Soc., Faraday Trans.*, vol. 93, pp. 629–637, 1997.

[43] S. M. Woodley, P. D. Battle, J. D. Gale, and C. R. A. Catlow, "The prediction of inorganic crystal structures using a genetic algorithm and energy minimisation," *Phys. Chem. Chem. Phys.*, vol. 1, pp. 2535–2542, 1999.

[44] S. M. Woodley and C. R. A. Catlow, "Structure prediction of titania phases: Implementation of darwinian versus lamarckian concepts in an evolutionary algorithm," *Comput. Mater. Sci.*, vol. 45, pp. 84–95, 2009.

[45] D. C. Lonie and E. Zurek, "Identifying duplicate crystal structures: Xtalcomp, and open–source solution," *Comput. Phys. Commun.*, vol. 183, pp. 690–697, 2012.

[46] https://github.com/psavery/randSpg.

[47] http://stokes.byu.edu/iso/findsym.php.

[48] H. T. Stokes and D. M. Hatcha, "Findsym: program for identifying the space-group symmetry of a crystal," *J. Appl. Crystallogr.*, vol. 38, pp. 237–238, 2005.

[49] http://valgrind.org/.

[50] https://opensource.org/licenses/BSD-3-Clause.