SHRY

Release 1.1

Genki Prayogo, and Kosuke Nakano

Jan 28, 2023

CONTENTS

SHRY (**S**uite for **H**igh-th**r**oughput generation of models with atomic substitutions implemented by p**y**thon) is a tool for generating unique ordered structures corresponding to a given disordered structure.

CHAPTER

ONE

INSTALLATION

SHRY is available via PyPI (pending)

```
pip install shry
```

1.1 Development

Installing from source

```
git clone https://github.com/giprayogo/SHRY.git
cd SHRY
pip install .
```

CHAPTER TWO

CONTENTS

2.1 How to use

Quick use

```
shry STRUCTURE_CIF
```

You can prepare CIFs with partial occupations by one of the ways below

2.1.1 Editing CIF file

The important part is the `_atom_site_occupancy` and `_atom_site_label`, which are typically grouped together in a loop

```
loop_
_atom_site_label
_atom_site_type_symbol
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_occupancy
_atom_site_U_iso_or_equiv
Sm1 Sm 0.000 0.00 0.00 1.000 0.0
Fe1 Fe 0.250 0.25 0.25 1.000 0.0
Fe2 Fe 0.278 0.50 0.00 1.000 0.0
```

Suppose that here we want to replace `Fe1` to a 40/60 mix together with Nb. Copy and edit the `Fe1` line, adjusting the labels and occupations.

```
loop_
_atom_site_label
_atom_site_type_symbol
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_occupancy
_atom_site_U_iso_or_equiv
Sm1 Sm 0.000 0.00 0.00 1.000 0.0
Fe1 Fe 0.250 0.25 0.25 0.400 0.0
```

(continues on next page)

(continued from previous page)

```
Nb1 Nb 0.250 0.25 0.25 0.600 0.0
Fe2 Fe 0.278 0.50 0.00 1.000 0.0
```

SHRY will automatically stop if the total occupancy of a site is either less or more than 1.0. To simulate vacancies, create a pseudatom with species X.

2.1.2 Using SHRY option to create partial occupation

The below example achieves the same modification. You can also choose to above by the below options.

```
shry [-f/--from-species] Fe1 [-t/--to-species] Fe0.4Nb0.6 STRUCTURE_CIF
```

Note that SHRY targets either `_atom_site_label` or `_atom_site_label`. If instead `Fe` is used in the first argument, all iron sites including `Fe2` will be replaced by `` `Fe0.4Nb0.6 ``.

2.1.3 Check total symmetry-inequivalent structures

```
shry --count-only STRUCTURE_CIF
```

This operation is based on Polya enumeration and takes much less time than a proper generation.

2.1.4 More command line options

Creating supercell

Sometimes a supercell is required to fit in finer concentrations. SHRY accepts either 3-digit (diagonal) or 9-digit (non-diagonal) format to specify the supercell's scaling matrix. For example a 2x2x1 supercell can be specified by either

```
shry -s 2 2 1 ...
```

or

```
shry -s 2 0 0 0 2 0 0 0 1 ...
```

Disorder only

If you just want to modify the CIF, without making the unique structures, you can add

```
shry --mod-only ...
```

Other options

Other options can be found in the help menu

```
shry -h
```

2.2 Using SHRY as a Python module

See `${SHRY_INSTALLDIR}/examples` for examples.

2.2.1 Basic function

(See `example1.py` for this section).

`Substitutor` is the main interface for using functions implemented in SHRY. It uses Pymatgen's `Structure` as the structure representation.

```
from pymatgen.core import Structure
from shry import Substitutor

...

structure = Structure.from_file(file_name)
substitutor = Substitutor(structure)
```

We recommend enumerating the unique structures using `Substitutor.count()` before generating them. It is an implementation of Polya enumeration, which is almost instant for most cases.

The structures (either as CIF or `Structure`), weights, configuration letters, etc. can then be obtained from `Substitutor.quantities(string_tuple)`. The `string_tuple` may contain any of these keywords:

- `cifwriter`. Pymatgen's `CifWriter` instances.
- `structure`. Pymatgen's `Structure` instances. Use this for writing CIF files.
- `weight`. How many configurations
- `letter`. Configuration letter ('aaa', 'bab', etc.) corresponding to the substitution.
- `ewald`. Ewald energy for the given structure.

`Substitutor.quantities(string_tuple)` is a generator of a dictionary with the previous keywords as keys. For example, if you want to get the CIFs and weights, do

```
for i, packet in enumerate(substitutor.quantities(("cifwriter", "weight"))):
    cifwriter = packet["cifwriter"]
    weight = packet["weight"]

    filename=f"cif_i{i}w{weight}.cif"
    cifwriter.write_file(filename=os.path.join(output_dir, filename))
```

There are also individual generators for each of the quantities:

- `cifwriters()`
- `structure_writers()`

- `weights()`
- `letters()`
- `ewalds()`

however these will invoke one full run for every call, so if more than one quantities is required, it will be slower.

2.2.2 Enumlib equivalent

See `example2.py` for comparison with equivalent `enumlib` functions through Pymatgen's `EnumlibAdaptor`.

2.2.3 Advanced use

LabeledStructure

`LabeledStructure` is a modified Pymatgen's `Structure`, but it tracks the CIF's `_atom_site_label`. This is useful if you want to group sites together regardless of supercell use, which can sometimes split the sites.

It also implements `replace_species` for substituting sites with a slightly more convenient syntax (see `example3.py`).

Saving substitutor instance

`Substitutor` can automatically "remap" pattern generated in other structure if the structures are symmetrically similar. This can save a lot of time if you are dealing with multiple concentrations or a set of symmetrically similar sytems.

See `example4a.py` for how to enable caching and pickle the `Substitutor` instance, and `example4b.py` for the later reloading.