MUSE Users Manual

Multi-algorithm-collaborative Universal Structure-prediction Environment

VERSION: 1.5.0

Zhong-Li Liu

E-mail: zl.liu@163.com

Contents

1 Introduction

1.1 About MUSE

MUSE is short for Multi-algorithm-collaborative Universal Structure-prediction Environment, which was developed for easy use in structure prediction of materials under ambient or extreme conditions, such as high pressure. It was written in Python and organically combined the multi algorithms including the evolutionary algorithm, the simulated annealing algorithm and the basin hopping algorithm to collaboratively search the global energy minima of materials with the fixed stoichiometry. After introduced the competition in all the evolutionary and variation operators, the evolution of the crystal population and the choice of the operators are self-adaptive automatically, i.e. the crystal population undergoes the self-adaptive evolution process. So, it can very effectively predict materials' stable and metastable structures under certain conditions only provided the chemical information of the material. Its success rate is almost 100%.

Presently, MUSE uses VASP, SIESTA, CASTEP, PWSCF, and LAMMPS as the local optimization tools. With the help of Spglib, it determines the space group number of each optimized structure immediately after the optimization is done. More importantly, it generates the random structures of the first generation with symmetry constraints, which largely decreases the optimization time of the first generation. The random structure is created according to the randomly chosen space group number between 1 and 230. It also can pick up an interrupted search from wherever the search stopped.

1.2 Multi algorithms collaborative

To enhance the search ability, MUSE uses ten evolutionary and variation operators, including cross over, mutation, permutation, cross over plus mutation, mutation plus cross over, permutation plus mutation, cross over plus permutation, random move, ripple and mutation plus ripple. Different systems have different optimal operators to achieve the possible largest diversity. So, from the second generation these ten evolutionary operators will compete with each other in their success rates starting from the initial value 100% of each operator. The operators are randomly used according to their historical behaviors from the third generation. That is to say, if an operator has more positive contributions (produced lower enthalpy structures or increased the diversity) to the whole population, it has more chances to be called to breed offspring. This is called the self-adaptation of evolutionary operators for different systems, which largely increases the diversity of structures and hasten the convergence of the global search. Whether the new optimized offspring in the main evolutionary loop is kept or not is determined by the Metropolis algorithm [1] based on the decreasing temperature, which is actually coupled the evolutionary algorithm and the simulated annealing algorithm together. Furthermore, to prevent prematuration, the main evolutionary loop is also coupled with basin hopping algorithm which has advantage of global optimization. After the main evolutionary loop is converged, MUSE will enter into the pure basin hopping loop to try to escape the possible local minimum. For technique details, please see Ref. [2].

2 Installation and run

Its installation and use are very easy, just as 'plug and play'. MUSE is based on Python. To increase the speed we also use Numpy. The Spglib is used to determine the space group number of each optimized structures and then MUSE determines the search direction according to the space group. For the local optimization, we presently use VASP, SIESTA, PWSCF, CASREP, and LAMMPS.

2.1 Installation requirements

The following packages are required:

a. Python 2.7 or later.

b. NumPy.

c. Spglib.

d. VASP or other local optimization code.

As for Spglib, please install pyspglib for structrues' space group determination.

2.2 Installation

The installation of MUSE is easy. When one has the python setuptools module installed. The first method is just type `python setup.py install` in the main directory. One can also specify clear directory to install using additional "`--prefix=/path/to/install/`". If one has no python setuptools, then the second method to install is: first put the following `PATH` and `PYTHONPATH` environment variable in your `~/.bashrc` file:

```
export PATH=$PATH:/path/to/MUSE
export PYTHONPATH=$PYTHONPATH:/path/to/MUSE
```

or in your `~/.cshrc` file:

```
setenv PYTHONPATH /path/to/MUSE:$PYTHONPATH
setenv PATH /path/to/MUSE/src:$PATH
```

Then, execute `source ~/.bashrc` or `source ~/.cshrc`.

2.3 Run

Just type `muse` or `python /path/to/MUSE/src/muse` to run the search in your work directory. The parallel run of local optimizations can be set in `muse.in`, see below.

2.3.1 Interface with VASP

To run MUSE with VASP, one need provide `muse.in`, `INCARs`, and `POTCAR-XX` files for each kind of atom, where `XX` is the short name of each atom. For example, you can use `POTCAR-Mg`, and `POTCAR-O` for MgO.

2.3.2 Interface with SIESTA

MUSE needs `muse.in`, `siesta.fdf`, and `XX.psf` files to perform the crystal structure search using SIESTA. The `XX.psf` are pseudopotential files of SIESTA code, where `XX` is the short name of each atom.

2.3.3 Interface with PWSCF

MUSE needs `muse.in`, `pwscf.in`, and `XX.UPF` files to perform the crystal structure search using PWSCF. The `XX.UPF` are pseudopotential files of PWSCF code, where `XX` is the short name of each atom.

2.3.4 Interface with CASTEP

MUSE needs `muse.in`, `sys.param` and `sys.cell` files to perform the crystal structure search using CASTEP. The pseudopotential file of each kind of atom is specified in the `sys.cell` and read from the CASTEP default directory.

2.3.5 Interface with LAMMPS

MUSE needs `muse.in`, `lammps.in`, and potential files to perform the crystal structure search using LAMMPS. The potential files named as `XX.pot` are specified in the `lammps.in` file.

3 Input examples

The following example is the structure search for MgO. The main input file is named as `muse.in`, in which the lines started with '#' are omitted. Each parameter and its value(s) are placed in the same line, in any order and with any number of blank lines. At least, values are separated by whitespace and placed after '='. The following is the `muse.in` file for MgO. The parameters are all explained in their comment line(s).

```
# The string to describe the system.
NameSys = MgO

# Element symbols of the different atoms
NameOfAtoms = Mg O

# Atomic number of each atom
AtomicNumber = 12 8

# The Number of each atom in the primitive cell in accord with
# AtomicNumber
Num_atomsInChemForm = 1 1

# Initial Guessed coarse primitive cell volume used for the first
# generation.  Proper value saves time for the first generation.
V_guess = 30.0

# The Minimum number of the primitive cell in the supercell for
# strarting with small system.  After the small system is fully
# converged, it is tricky to quikly converge the larger system based
# on the information of the small system.
MinNum_PrimitiveCell = 4

# The Maximum number of the primitive cell in the supercell (the
# largest system)
MaxNum_PrimitiveCell = 16

# The increment of number of the primitive cell from Min to Max
# number of primitive cell in the supercell
StepNum_PrimitiveCell = 2

# Minimum distance between atoms or ions
```

```
MinDist = 1.3

# Minimum vector of the primitive cell
MinVect = 2.0

# Pressure in GPa.
Pressure = 100

# If pick up the optimized structures from the last run for
# continuing it (1 for yes, 0 for no)
IfPickUp = 0

# If use Symmetry Constraints in the first generation.(1 for yes, 0
# for no)
IfSymmConstr = 1

# Population Size of each generation
PopSize = 30

# Percentage of each population for generating next generation
Perc4NextGen = 0.75

# Number of the best individuals directly passed to the next
# generation
Num_Keep = 2

# If the offsprings directly from parents are reoptimized (1 for yes,
# 0 for no)
IfReOptKept = 0

# The number of maximum try of each operator for the selected
# individual(s) until it has positive
# contribution (produce lower enthalpy str.  or changed the space
# group to increase the diversity).
MaxTry_EachOperator = 1

# If compete between generations (parents and offspring) (1 for yes,
# 0 for no)
IfCompeteBetweenGens = 0

# Initial Gaussian standard deviation for individuals
InitGaussStdDev = 0.5

# The number of continuous generations with best enthalpy and the
# same structure used to stop the search
Num_GenBest = 6

# Inital temperature for annealing
InitTemp_anneal = 100

# Maximum generation
```

```
MaxGen = 30

# Algorithm for evolution, EA: evolutionary algorithm, SA: simulated
# annealing, BH: basin hopping, MAC: multi algorithm collaboration.
Algorithm = MAC

# The precision for determining the space group
SymPrec = 0.1

# Optimization Code (1:  vasp; 2:  siesta; 3:  pwscf; 4:  castep;
# 5:  lammps)
OptCode = 1

# The parallel submiting commmd
ParSubCom = mpirun -np 8 vasp > log.vasp
#ParSubCom = mpirun -np 8 siesta <siesta.fdf >OUT
#ParSubCom = mpirun -np 8 pw.x <pwscf.in >OUT
#ParSubCom = RunCASTEP.sh -np 8 sys
#ParSubCom = mpirun -np 8 lmp_openmpi <lammps.in >OUT

# K point resolution in each local optimization.  The number is in
# accord with the optimization number in input file.
K_res = 0.08 0.07 0.05 0.03

# The maximum time for every local optimization in hour.  Local
# optimization exceeded this time value will be killed and the next
# will begin.
MaxHour = 1.0
```

4 Output files

All the output files are in the `running` directory. When a new calculation starts, this directory will be renamed to `doneN`, where N is an integer. The `poscars_1st` file is the collection of POSCAR files for the randomly generated structures with symmetry constraints in the first generation. The `poscars-N` is the collection of POSCAR files for all the generations with N primitive cells after optimizations. The `all-enthalpy-N` file contains the enthalpies of the optimized structures along with the symmetry information and the volumes. The `best-enthalpy-N` file collected the enthalpies of best enthalpies of each generation. The `kgrids-N` file collected the k-point meshes for each optimization.

5 Tools

The `spg.py` is a tool for determining the symmetry of a structure with POSCAR format and reduces it to the primitive cell. The `findsym.py` is for the determination of symmetry of a structure using FINDSYM code. The `strana.sh` is the tool for re-analysis of symmetry for all the structures using Spglib and/or FINDSYM. The `hor` script can list the structural order according to energy, e.g., `hor all-enthalpy-2`.

References

[1] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, J. Chem. Phys. 21 (1953) 1087.

[2] Z. L. Liu, Comput. Phys. Commun. 185 (2014) 1893.