AFLOW V 3.1.155

```
*******************************************************************************
*                                                                             *
*            Aflow STEFANO CURTAROLO - Duke University 2003-2018               *
*               High-Throughput ab-initio Materials Discovery                 *
*                                                                             *
*******************************************************************************
```

AFLOW_COMPARE_STRUCTURE
David Hicks (d.hicks@duke.edu) and Carlo de Santo

```
*******************************************************************************
```
GENERAL OVERVIEW

This algorithm takes two crystal structures and determines their level of similarity
based on criteria specified by H. Burzlaff (Acta Cryst., A53, 217-224 (1997)).

The code returns a level of misfit between the two structures. The results of the
misfit characterize the level of similarity:
  - misfit <= 0.1   : structures are similar/compatible
  - 0.1 < misfit <= 0.2 : structures are within the same family
  - misfit > 0.2   : structures are not compatible/no match found

This codes has the ability to compare material comprised of the same species (find
a duplicate material) or compare based on structure only (find underlying
structure, no requirement on matching alike atomic species).

There are two manners in which to compare materials/structures.
  1) Two at a time
  2) Entire directory: There is a scheme to compare an entire directory. Note: The only
                       files which can reside in the directory are position files.
                       (xstructure will fail otherwise.)


```
*******************************************************************************
```
AFLOW COMMANDS:

Two at a time comparison:

      aflow --compare_material=POSCAR_1,POSCAR_2 [--np=xx] [--print]
            Description: Compares POSCAR_1 to POSCAR_2 if the ratios are commensurate
                         and types of atoms are the same (i.e. same material).
                         POSCARs must be in the same directory you are running in.
                         [--np=xx] : Number of processors. Algorithm is thread-friendly.
                         [--print] : (For two at a time only.) Prints comparison data.

      aflow --compare_structure=POSCAR_1,POSCAR_2 [--np=xx] [--print]
            Description: Compares POSCAR_1 to POSCAR_2 (no requirement on type of atoms,
                         just stoichiometry ratios). POSCARs must be in the same
                         directory you are running in.
                         [--np=xx] : Number of processors. Algorithm is thread-friendly.
                         [--print] : (For two at a time only.) Prints comparison data.

Directory comparison:

      aflow --compare_material_directory|--compare_material_dir [-D "path"] [--np=xx]
            Description: Determines the unique materials (same atomic species)
                         within a given directory. Returns a JSON and TXT file with the
                         results. Default Directory: ".".
                         -D "PATH" : User can specify a specific directory to compare.
                                     Output will be placed there also.
                         [--np=xx] : Number of processors. Algorithm is thread-friendly.

      aflow --compare_structure_directory|--compare_structure_dir [-D "path] [--np=xx]
            Description: Determines the unique structure prototypes within a given directory.
                         Returns a JSON and TXT file with the results. Default Directory: "."

```
                    -D "PATH" : User can specify a specific directory to compare.
                                Output will be placed there also.
                   [--np=xx] : Number of processors. Algorithm is thread-friendly.


******************************************************************************
NOTE ON STRUCTURE SIZE AND SPEED

This algorithm becomes expensive if you have many atoms in your system (due to 3x3x3
supercell expansion to find commensurate lattices). It is mainly dependent on you
least frequently occuring atom (LFA). If the LFA>20 atoms, then the comparison may
take quite a bit of time (~hours). The algorithm quits immediately if a match is found.
However, the time consuming aspect arises if a match cannot be found.

WAYS TO ENHANCE COMPUTATION SPEED:
  1) Convert structures to primitive representation
  2) Increase the number of processors (--np=xx)


******************************************************************************
OVERVIEW OF COMPARISON ALGORITHM

Steps:
    1) Input two structures: structure1 and structure2
    2) Scale volumes of two structures to be commensurate
    3) Determine the least frequently occuring atom (LFA)
    4) Shift LFA for both structures to their respective origins;
       these will be used to indicate our lattice
    5) Create a 3x3x3 supercell of structure2
    6) Search for all possible quadruplets (sets of 4) of LFA atoms in the
       supercell and see if we can match it with structure1
       using Burzlaff's criteria. (This part is parallel if more than one
       processor is specified.)
    7) Once possible quadruplet/lattice is found, check contents
       of lattice (i.e. atoms). Check we can have a one-to-one
       mapping. Check that the lattice chosen preserves periodicity.
    8) Of the best match (smallest misfit (mis)):
       If mis <= 0.1:
         Structures similar. Print out new representation of
         structure2 and the figure of misfit
       else if 0.1 < mis <=0.2:
         Structures in the same family (possible symmetric
         group-subgroup relation). Print out new representation
         of structure2 and the figure of misfit
       else mis>0.2:
         Structures are not the same. Print "No match"


******************************************************************************
OVERVIEW OF DIRECTORY COMPARISON SCHEME

Steps:
    0) Create a directory of geometry files ONLY.
    1) Read in all files in directory and convert to xstructures.
    2) Obtain stoichiometry/Pearson symbol/space group (grouping in a smart way
       limit comparisons).
    3) Compare structures that have same stoichiometry, Pearson symbol, and space group
       by picking a master structure and comparing it with the other structures.
    4) If any of the structures did not match the master, create another group of
       comparisons for the ones that did not match. Continue until all structures are
       matched or until all comparisons are exhausted.
    5) Print out results in both a .json file (for easy python manipulation) and .txt file.


For additional information contact: David Hicks (d.hicks@duke.edu)
// ******************************************************************************
```