CSD Conformer Generator User Guide

CSD Conformer Generator User Guide

2022.3 CSD Release

Conditions of Use

Introduction

Overview of the CSD Conformer Generator

Conformer generation is important in computer aided drug design and discovery, and many programs have been developed that attempt to reproduce biologically relevant conformations from initial basic chemical models. The CSD Conformer Generator is a new knowledge-based program that uses the wealth of knowledge in the CSD to explore the conformational space of small molecules.

Given an input 3D molecule with all hydrogen atoms present, which is optionally minimized, pre-determined CSD rotamer and CSD ring distributions are incrementally applied to a fragmented view of the molecule and the generated conformers assigned scores based on sample relative frequencies (i.e. approximate probabilities) of the geometric parameters assigned. Conformers are monitored for clashes during the incremental build up procedure allowing early rejection of clashing conformations. A final diverse set of up to n conformers (default n=200) clustered according to conformer similarity is returned. Each conformer of the final set is locally optimized in torsion space (Reference 1).

As a CSD user, you may be familiar with the program Mogul which contains four data libraries, one each for bond lengths, valence angles, torsion angles, and unfused, unbridged rings. Although Mogul was the obvious starting point for the knowledge base required by the conformer generator algorithm, some changes were made to improve search speeds, produce torsion distributions that correctly represented the influence of any chirality present, extend the ring library to cover fused rings, improve handling of symmetry, and pre-cluster ring distributions so that only distinct conformational minima were retained.

A new type of fragment, the rotamer, has been introduced and rotamer libraries are used instead of torsion libraries. The reason can be explained with an example rotatable bond X–Y in the context of a fragment $R_A(R_B)X–Y(R_C)R_D$. Its conformation can be defined by any of the torsion angles $R_A–X–Y–R_C$, $R_A–X–Y–R_D$, $R_B–X–Y–R_C$, and $R_B–X–Y–R_D$. Each of these is considered a separate torsion-angle fragment in Mogul. Therefore, if X–Y is in a CSD crystal structure, it contributes to four distributions in the Mogul torsion-angle library. If it is in a query molecule, four distributions are retrieved. Furthermore, the keys used in the Mogul torsion library capture more information about the atoms defining the torsion angle than the other atoms bonded to X and Y: for example, there is more information about R_A and R_C than about R_B and R_D for the torsion-angle fragment $R_A–X–Y–R_C$. As a consequence, the four separate distributions are based on different crystallographic observations, though some overlap is likely. This would create extra work for a conformer-generation algorithm using the library because the probability of any hypothesized geometry around X–Y would be a function of all four distributions. On the contrary, in a rotamer library, each rotatable bond in the CSD contributes to just one distribution, and only one distribution per rotatable bond is retrieved in a search. In 2014, Taylor et al. published an in-depth description about the creation of these new rotamer libraries (Reference 2).

In 2018, Cole et al. published a comparison of the performance of the improved version of the CSD Conformer Generator (available from release 1.2 and above) with previously published evaluations (Reference 3). Its performance was found to be significantly improved in reproducing the molecular conformations of structures from the Cambridge Structural Database and the Protein Data Bank, as compared to other published methods of a similar speed. Please use this latest publication when citing the CSD Conformer Generator.

There are three ways of using the program:

- Via a command line utility which is described in the remainder of this document.
- Interactively via the **Mercury** graphical user interface (please see the dedicated user guide for more information).
- Via the CSD Python API. Please see the CSD Python API documentation for details.

Release notes for the 2022.1 CSD Release

There are no major changes in the 2022.1 release of the CSD Conformer Generator.

Installation

The software is installed as part of the CSD Software Portfolio.

On Windows, the directory containing the executable is automatically added to your system PATH.

On Linux, you have to set the environment's default Python installation to Python 3.7. For example, for bash shell:

```
export PATH=/path/to/python3.7:$PATH
```

On macOS, the directory containing the conformer generator executable should be added to the user PATH. This can be achieved by opening a terminal and typing:

```
export PATH=/path/to/conformer_generator/bin:$PATH
```

Ligand input

The program requires 3D molecules with all hydrogen atoms present in either .mol2 or .sdf format.

Generating conformers

Running for the first time

At the simplest level, the conformer generator can be used with an input file and an output file, with no further options. The command:

```
conformer_generator input_file.mol2 conformers.mol2
```

will read the file `input_file.mol2` and write out a file called `conformers.mol2` containing conformers for each molecule specified in the input file. By default, the program will generate at most 200 conformers for each molecule in the input file. The conformers will be sorted into their order of likelihood as predicted by the program.

On both Linux and Windows, the program should be run from a directory where you have permission to write, so that the output file may be created. For Windows in particular, this means it should not be run from within the installation directory.

If the output file already exists, the program will stop with an error. You can override this behaviour using the `-f` option, for example:

```
conformer_generator input_file.mol2 conformers.mol2 -f
```

Controlling the output

You can change the behaviour of the program by altering a variety of parameters.

The output format written is controlled by the file extension given to the output filename. The command:

```
conformer_generator input_file.mol2 conformers.sdf
```

will write out an SD file called `conformers.sdf` containing the conformers, instead of a .mol2 file. It is possible to write out the conformers in more than one file by specifying multiple output files. This option is useful for generating the output in multiple formats simultaneously.

You can, alternatively, write the conformers to "Standard Out" (SD) which can be useful for piping data out of the program into other processes. For example:

```
conformer_generator input_file.mol2 STDOUT -ot sdf
```

will write SD format conformers to stdout and redirect usual stdout to a file called `redirected_stdout.txt`.

It is also possible to generate separate output files of conformers per input molecule. This is performed by specifying a subdirectory to contain the output files, for example:

```
conformer_generator input_file.mol2 -ot sdf -od
output_directory
```

This will create a new directory called `output_directory` and write the files to it. Note that the program will stop with an error if the output directory already exists to avoid accidental overwriting of files. Again, as with files, the `-f` option can be used to override this behaviour.

The following example illustrates how to write out additional information on the number of flexible torsions and rings. The command:

```
conformer_generator input_file.mol2 conformers.sdf stats.csv
```

will write the conformers to `conformers.sdf` and additionally output some statistics to two comma separated files: `stats.csv` and `stats_per_conformer_scores.csv`. For more information, please see Output description.

What if I want more than 200 conformers?

By default, the CSD Conformer Generator generates a maximum of 200 conformers per input molecule. At the simplest level, running the program with the option `-nc` will change the maximum number of conformers allowed, for example:

```
conformer_generator input_file.mol2 conformers.mol2 -nc 500
```

will allow the program to generate up to a maximum of 500 conformers for a particular molecule. This, however, will not guarantee that the program will generate exactly 500 conformers for that molecule; for example, the search space may be exhausted before 500 distinct (according to the clustering algorithms used by the program) conformers are generated.

How can I change the conformer dissimilarity used for the clustering?

By default, the program automatically selects conformer dissimilarity metrics on a per-molecule basis. If the torsion dissimilarity of a conformer to an already accepted one is higher than a specific threshold, they are assumed to be dissimilar and the new conformer is kept. If the torsion dissimilarity value is less than the threshold, then the atom rmsd between the two conformers is checked. If the atom rmsd is below the threshold, the conformers are similar and the new conformer is discarded, otherwise it is added to the list of accepted conformers. You can set the tolerances of the clustering by using the `-adt` (atom dissimilarity) and `-tdt` (torsion dissimilarity) options to alter the definition of conformer similarity. Specifying either of the two dissimilarity parameters will disable the use of automatically derived parameters. You should note, however, that adjusting these options can currently incur a significant performance penalty. Here is a table of suggested combinations.

`-adt`	`-tdt`
0.5	100
0.75	225
1.0	350
1.25	450

Generating conformers for lots of molecules

A common usage of a conformer generator is to create conformers for many molecules. The CSD Conformer Generator supports usage of multiple threads on a single machine. By default, only one thread is used, but this can easily be overridden with the `-nt` option, for example:

```
conformer_generator input_file.mol2 conformers.mol2 -nt 4
```

would run the program utilizing 4 threads on a given machine.

Please note that all threads will share the memory available to the conformer generator process. Thus, it is not recommended to use more than 4 threads when running a 32 bit version of the conformer generator.

Output description

When the output format `.csv` is specified, two files will be written out. The first contains data for each molecule run through the generator. The properties and descriptions of the first file are as follows:

reader.pass.molecule_name
    Name of the molecule as read from the input file

reader.pass.molecule_filename
: Name of the input file

conf_gen.pass.max_log_prob_conf_theory
: The maximum possible log probability (natural logarithm) for a conformer of this molecule

conf_gen.pass.min_log_prob_conf_theory
: The minimum possible log probability (natural logarithm) for a conformer of this molecule

conf_gen.pass.adjusted_max_clash_value
: The clash value finally used to generate conformers for this molecule

conf_gen.pass.cluster_atom_rmsd
: The clustering atom RMSD value used for this molecule

conf_gen.pass.cluster_torsion_dissimiliarty
: The clustering torsion dissimilarity used for this molecule

conf_gen.pass.n_conf_gen_clust
: Number of conformers generated (after clustering)

conf_gen.pass.n_tors
: Number of torsions sampled by the conformer generator

conf_gen.pass.n_ring
: Number of flexible ring systems sampled by the conformer generator

conf_gen.pass.max_conf_reached
: A Boolean flag to indicate if the search generated the maximum number of conformers before sampling the whole conformer tree

conf_gen.pass.n_zero_obs_rotamers
: The number of rotamers that could not be treated using Mogul distributions due to a lack of observations for this molecule (uniform distributions are used instead)

conf_gen.pass.n_zero_template_rings
: The number of rings that could not be treated using templates due to a lack of observations (the input ring conformation is used instead)

reader.pass.molecule_id
: The index of the molecule

A second file is also written out that contains fields for each conformer that was generated. The fields in this .csv file are as follows:

reader.pass.molecule_name

  Name of the molecule as read from the input file

conf_gen_conformers.pass.prob

  The natural logarithm of the probability of this conformer

conf_gen_conformers.pass.clash

  The clash score measured for this conformer (after optimisation in torsion space)

conf_gen_conformers.pass.normalised_score

  A score that normalises the probability into the range of 0.0, 1.0 given by the formula: $\frac{\ln(maxp) - \ln(p)}{\ln(maxp) - \ln(minp)}$ where maxp and minp are the maximum and minimum possible probability scores for this molecule and p is the probability associated with this conformer. A zero score indicates the highest likelihood conformer. A score of 1.0 would indicate the lowest possible likelihood conformer. This value is limited by the `normalised_score_threshold` parameter.

conf_gen_conformers.pass.conformer_id

  The numerical index of the conformer

Conformer generation failures

By default, a first optional minimization step is performed before conformer generation. It optimizes bond length and bond angles of the input molecule according to CSD data. The geometrical optimization makes use of the Tripos force field functional forms and, where available, equilibrium bond distances and valence angles are parameterized using data obtained from the CSD. If the minimization fails with an unrecoverable error, the molecule will be skipped (for the list of skipped molecules see the `minimisation_failures.csv` file).

If no conformers are generated or the conformer generator exits abnormally, the input structure supplied to the conformer generator will be returned. This will be the minimized structure if minimization is turned on and the original input structure if minimization is deactivated. The molecules for which no conformers have been generated will be listed in the `conformer_generator.warn` file.

Usage reference

List of commands:

```
-h

-od <OUTPUT_DIRECTORY>

-ot <OUTPUT_TYPE>

-f

-nc <N_CONFORMATIONS>

-ng <N_CONFORMATIONS_IN_SEARCH>

-io

-im

-mut <MAXIMUM_UNUSUAL_TORSIONS_ALLOWED>

-mrp <MINIMUM_ROTAMER_PROBABILITY>

-tdt <TORSION_DISSIMILARITY_THRESHOLD>

-adt <ATOM_DISSIMILARITY_THRESHOLD>

-nst <NORMALISED_SCORE_THRESHOLD>

-nt <N_THREADS>

-sr

-sm

-sg

molecule_file

output_files

output_files ...
```

Optional arguments:

`-h`, `--help`: show this help message and exit.

File reading & writing options:

`molecule_file`: The input file to use (in SD file format or mol2 file format).

`output_files`: The output file names to use. Output files should have a suffix of .mol2, .sdf, or .csv to indicate format. You can specify more than one output file (e.g. a .csv file and a .mol2 file, but not both .mol2 and .sdf files at the same time). An output file called **STDOUT** will cause the program to write output to stdout and redirect other output to a log file; default format is .mol2. If `STDOUT` is specified, the user can provide the `<OUTPUT_TYPE>` argument (see `-ot`) to control the format written out.

`-od <OUTPUT_DIRECTORY>`, `--output_directory <OUTPUT_DIRECTORY>`: The output directory to write output files to. This argument can be used in tandem with an `<OUTPUT_TYPE>`.

`-ot <OUTPUT_TYPE>`, `--output_type <OUTPUT_TYPE>`: This argument specifies the output format to use. It can be used when writing each conformational ensemble for each input molecule to a separate file or when writing to stdout. `<OUTPUT_TYPE>` can be ‘sdf’ or ‘mol2’. If specified, then `output_files` should not be specified (except as `STDOUT`).

`-f`, `--force_overwrite`: Force output file overwriting.

Options to control how many conformations to generate and save:

`-nc <N_CONFORMATIONS>`, `--n_conformations <N_CONFORMATIONS>` : The number of conformers to keep after clustering (default: 200).

`-io`, `--include_original`: Include the original read-in conformation (default: off).

`-im`, `--include_minimised`: Include the initial minimised conformation (default: off).

Options to control when to accept or reject a given conformation:

`-mut <MAXIMUM_UNUSUAL_TORSIONS_ALLOWED>`, `--maximum_unusual_torsions_allowed <MAXIMUM_UNUSUAL_TORSIONS_ALLOWED>`: The maximum number of unusual torsions permitted in a molecule.

`-mrp <MINIMUM_ROTAMER_PROBABILITY>`, `--minimum_rotamer_probability <MINIMUM_ROTAMER_PROBABILITY>`: The minimum rotamer probability (how likely a rotamer has to be to be regarded as probable).

`-tdt <TORSION_DISSIMILARITY_THRESHOLD>`, `--torsion_dissimilarity_threshold <TORSION_DISSIMILARITY_THRESHOLD>`: The torsion threshold of dissimilarity: if two conformers have a torsion dissimilarity greater than this, then they are regarded as possibly different (depending on atom rmsd, see option `-adt`).

`-adt <ATOM_DISSIMILARITY_THRESHOLD>`, `--atom_dissimilarity_threshold <ATOM_DISSIMILARITY_THRESHOLD>`: The atomic threshold of dissimilarity: if two conformers have an atom dissimilarity greater than this, then they are accepted as being different conformers (provided they have first been regarded as dissimilar by the torsion dissimilarity threshold, see option `-tdt`).

`-nst <NORMALISED_SCORE_THRESHOLD>`, `--normalised_score_threshold <NORMALISED_SCORE_THRESHOLD>`: A threshold value between 0 and 1 describing the maximum allowed deviation from the conformer with the highest theoretical probability. If the normalised score of a conformer is less than this threshold, it will be kept otherwise it will be discarded. Threshold values closer to 0 will restrict the search to high probability conformers, while values closer to 1 will also allow low probability conformers to be sampled, where 1 represents the conformer with the lowest theoretical probability. The default value is 0.5.

Other options:

`-nt <N_THREADS>`, `--n_threads <N_THREADS>`: The number threads to use (default: 1).

`-sr`, `--skip_run`: Skip workflow run (just write out the workflow file).

`-sm`, `--skip_minimisation`: Skip pre-minimisation of input molecules.

`-sg`, `--skip_generation`: Skip conformer generation; means the input molecules just get minimized.

References

1. The Cartesian minimization and the conformer post-optimization make use of “libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS)”, http://www.chokkan.org/software/liblbfgs.

2. Taylor, R.; Cole, J. C.; Korb, O. and McCabe, P. Knowledge-Based Libraries for Predicting the Geometric Preferences of Druglike Molecules. J. Chem. Inf. Model. (2014) **54**, 2500-2514.

3. Cole, J. C.; Korb, O.; McCabe, P.; Read, M. G. and Taylor, R. Knowledge-Based Conformer Generation using the Cambridge Structural Database. J. Chem. Inf. Model. (2018) **58**, 615-629.