JOURNAL OF
APPLIED
CRYSTALLOGRAPHY

ISSN: 1600-5767
journals.iucr.org/j

Recent improvements in *DSR*

Daniel Kratzert and Ingo Krossing

J. Appl. Cryst. (2018). **51**, 928–934

JOURNAL OF APPLIED CRYSTALLOGRAPHY

ISSN 1600-5767

Recent improvements in *DSR*

Daniel Kratzert* and Ingo Krossing

Institut für Anorganische und Analytische Chemie, Albert-Ludwigs University of Freiburg, Albertstrasse 21, Freiburg, Baden Württemberg 79104, Germany. *Correspondence e-mail: dkratzert@gmx.de

Received 15 February 2018
Accepted 17 March 2018

Edited by S. Sasaki, Tokyo Institute of Technology, Yokohama, Japan

Keywords: geometric restraints; refinement; modelling; graphical user interfaces; *DSR*; *SHELXL*.

CCDC reference: 1822173

Supporting information: this article has supporting information at journals.iucr.org/j

The *DSR* computer program has received many minor and major updates over the past two years. This publication describes some new features, such as disorder modelling on special positions, error detection for restraints and trifluoromethyl group modelling. Most importantly, the graphical user interfaces (GUIs) make *DSR* a lot easier to use, especially in modelling disorder on special positions that would have been difficult to implement without a GUI. In addition, generating and editing of new fragments in the database is now much easier.

1. Introduction

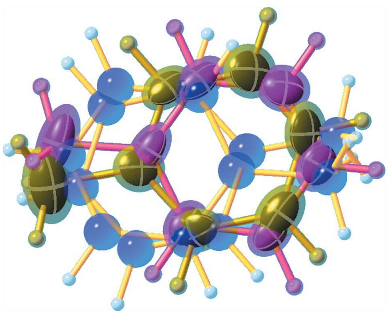
In small-molecule crystallography, one of the most complex tasks is the modelling of disorder. The goal of developing the computer program *Disordered Structure Refinement* (*DSR*; Kratzert *et al.*, 2015) is to provide a simpler and less error-prone way than to build a model of disorder from each individual atom. Rather, the *DSR* method builds up a fragment-by-fragment model, with predefined molecular fragments taken from a database and then transferred into the crystal structure. This procedure has already proved to be useful in numerous cases for our group and others. For example, one structure had 1400 atoms in 101 residues and would have been almost impossible to refine without a modelling program (Lichtenthaler *et al.*, 2015).

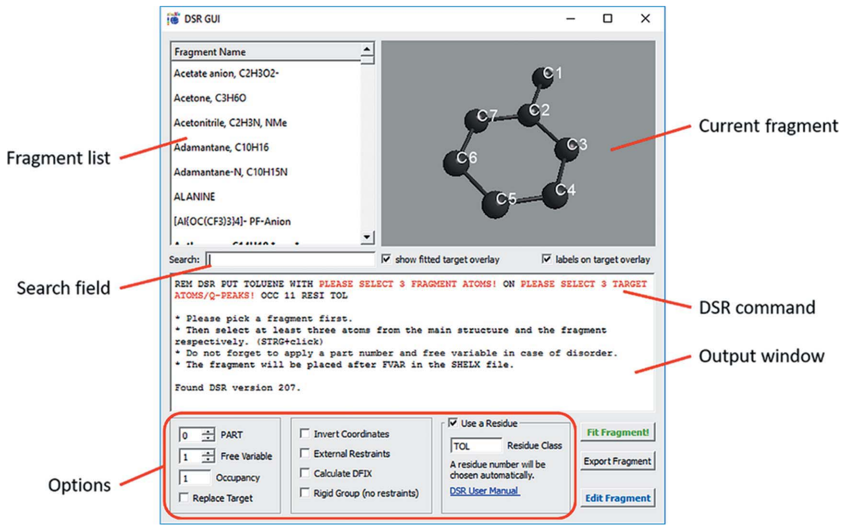
The first version of the *DSR* program was released towards the end of 2013. Since then, about 60 more fragments have been added to the database of the program. Although this version has saved a lot of painstaking work in modelling large disordered structures, the current version is even easier to use. The first versions had a relatively simple text interface, which was sufficient to define where and under what conditions a molecular fragment should be transferred into a crystal structure to be refined using *SHELXL* (Sheldrick, 2015), but today's crystal structures are mostly refined using *SHELXL* with the help of a graphical user interface (GUI) instead of a pure text editor (Dolomanov *et al.*, 2009; Hübschle *et al.*, 2011; Farrugia, 2012; McArdle, 2017). Although *DSR* should work well in combination with any refinement interface using text mode, there was a need to integrate *DSR* into the most common GUIs, *ShelXle* (Hübschle *et al.*, 2011) and *OLEX2* (Dolomanov *et al.*, 2009).[1]

2. General implementation

DSR itself was written in pure Python (https://www.python.org/) and is executed *via* the system command line. The

[1] It would also be beneficial to have *DSR* or something similar being implemented in *CRYSTALS* (Cooper *et al.*, 2010) or *JANA* (Petříček *et al.*, 2014).





```
REM DSR PUT TOLUENE WITH PLEASE SELECT 3 FRAGMENT ATOMS! ON PLEASE SELECT 3 TARGET
ATOMS/Q-PEAKS! OCC 11 RESI TOL

* Please pick a fragment first.
* Then select at least three atoms from the main structure and the fragment
respectively. (STRG+click)
* Do not forget to apply a part number and free variable in case of disorder.
* The fragment will be placed after FVAR in the SHELX file.

Found DSR version 207.
```

[✓] show fitted target overlay   [✓] labels on target overlay

[0] PART   [1] Free Variable   [1] Occupancy   [ ] Replace Target

[ ] Invert Coordinates   [ ] External Restraints   [ ] Calculate DFIX   [ ] Rigid Group (no restraints)

[✓] Use a Residue   [TOL] Residue Class

A residue number will be chosen automatically.

Figure 1
The main window of the *DSR* GUI in *ShelXle*.

interface to *DSR* in *ShelXle* was implemented as a *QWidget* from the Qt Library (http://doc.qt.io) in C++ with Qt4. The interface in *OLEX2* called *FragmentDB* is also implemented with Python but using the wxWidgets Library (https://www.wxwidgets.org). All source code is freely available on Github at https://github.com/dkratzert. The code base and ideas are similar for both programs, but some details are completely different, for example the fitting procedure.

3. Implementation in *ShelXle*

When we created the graphical interface, our aim was that *DSR* should be changed as little as possible. Nevertheless, users should be able to install *DSR* without having to perform further configuration. To obtain a simple machine-readable interface, general text output readable by GUIs has been implemented in *DSR*. In the other direction, the GUI talks to *DSR* using the command-line options and the special *DSR* command line. The special *DSR* command is written in the *SHELXL* `.res` file: see Kratzert *et al.* (2015) for more details. Whatever option is chosen, the GUI always shows the *DSR* command at the top of its output window (Fig. 1).

The GUI also needs a way of displaying a list of the structures contained in the database. The command line option `-lc` changes the format of the list of structures to a machine-readable format. A mouse click on a name in this list executes *DSR* with the option `-ah`, which in turn generates a machine-readable output of the database entry to be displayed as a three-dimensional representation (Fig. 1). It is also possible for the GUI to send information to *DSR*. Option `-shx [path]` tells *DSR* where the *SHELXL* executable is located in the file system and option `-target [[x y z], ...]` takes coordinates as a target for the fragment fit. The latter is important for target atoms or maxima in the difference electron density map (*q* peaks) that are around special positions (see §4). Finally, `-x` searches for fragments and displays the result with the same syntax as `-lc`. In this way, search results can be displayed instantly.

3.1. Fragment fit

To customize a fragment in *ShelXle*, the user must first select a fragment in the list. Then it has to be decided which atom of the fragment should go to which position in the crystal structure. For this purpose, it is necessary to select three atoms (which must not be collinear) in the structure and in the fragment by mouse clicks. If this condition is met, the *ShelXle DSR* GUI displays a preview of the fit, showing the target atoms as a wireframe and the fragment as a ball-and-stick model (Fig. 2).

The button 'Fit Fragment' (Fig. 1) writes the *DSR* command into the *SHELXL* `.res` file directly after FVAR. The GUI launches *DSR*, a list of errors or successes is displayed in the output window, and finally *ShelXle* reloads the current `.res` file to get the results from *DSR*.

Because *DSR* introduces many restraints in a short time, it is necessary to have good visual feedback about the quality of the model. Therefore, a helpful new feature that has been introduced is the prominent display of the list of the most disagreeable restraints from the *SHELXL* output in *ShelXle* ('Visualize Output' option) and *OLEX2*. During the *DSR* or *FragmentDB* modelling operations, the user always has feedback about the compliance of the restraints with the applied standard deviations and distances. This possibly prevents the blind application of restraints that do not match the data.

3.2. Fragment editor

The fragment editor helps with the editing and creation of new molecular fragments. Existing fragments can be modified

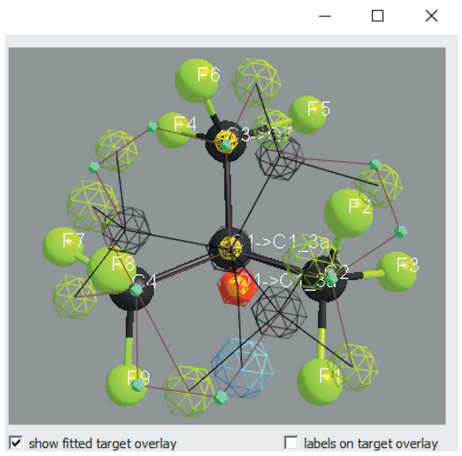
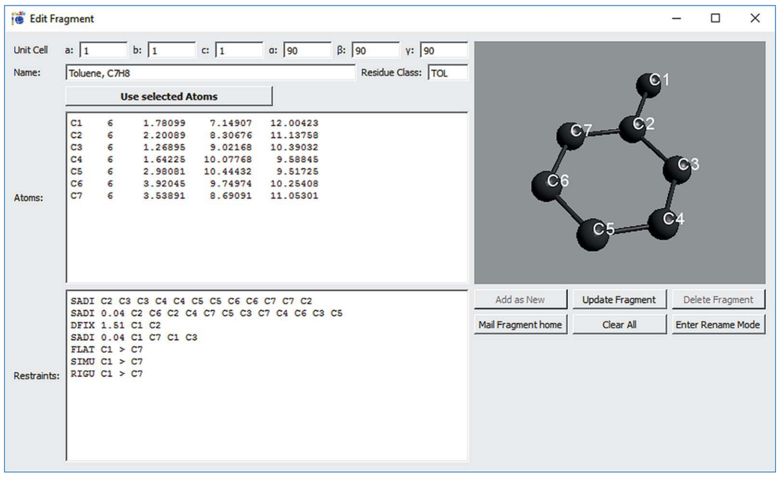


Figure 2
Preview of a fragment fit (upper right corner of Fig. 1) prior to the actual transfer. The $[OC(CF_3)_3]^-$ fragment (balls and sticks) is placed on the second position caused by rotational disorder along the C—O bond of the perfluoro-*tert*-butyl alcohol group (wireframe model with target position environment).



```
C1    6    1.78099    7.14907   12.00423
C2    6    2.20089    8.30676   11.13758
C3    6    1.26895    9.02168   10.39032
C4    6    1.64225   10.07768    9.58845
C5    6    2.98081   10.44432    9.51725
C6    6    3.92045    9.74974   10.25408
C7    6    3.53891    8.69091   11.05301
```

```
SADI C2 C3 C3 C4 C4 C5 C5 C6 C6 C7 C7 C2
SADI 0.04 C2 C6 C2 C4 C7 C5 C3 C7 C4 C6 C3 C5
DFIX 1.51 C1 C2
SADI 0.04 C1 C7 C1 C3
FLAT C1 > C7
SIMU C1 > C7
RIGU C1 > C7
```

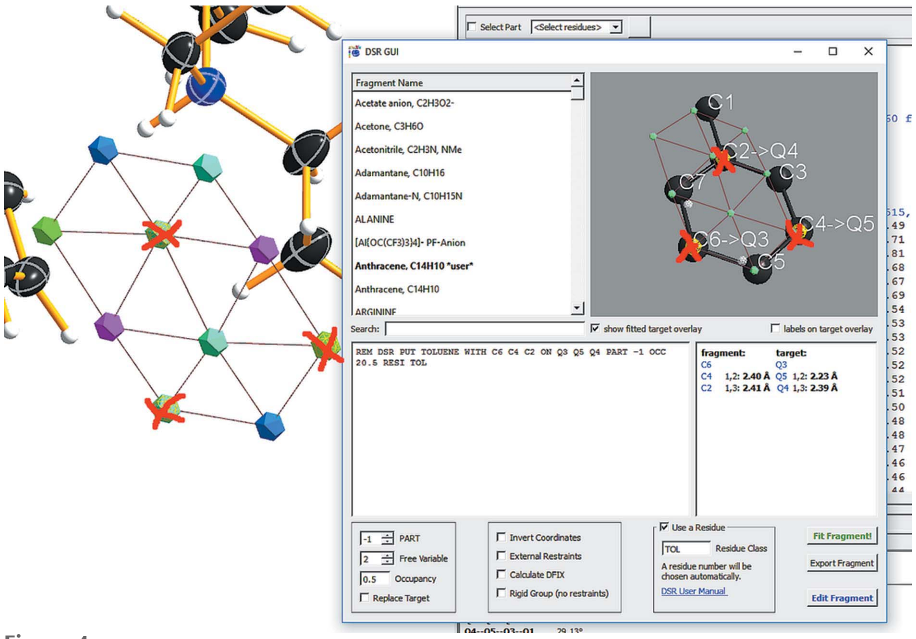
Figure 3
The fragment editor window.

by clicking on ‘Edit Fragment’. The editor window (Fig. 3) allows the user to manipulate the atoms, with one atom in each row written as name, atomic number, *x*, *y* and *z*, separated by white space. The restraints field allows the creation of all restraints allowed in *SHELXL* but with the limitation that the atom names mentioned have to be in the fragment atoms list. The unit-cell input takes the unit-cell parameters of the structure data from which the fragment comes. Upon saving the new or edited fragment, the coordinates are simultaneously converted to Cartesian coordinates. Therefore, it is also possible to introduce Cartesian coordinates with a unit cell of 1 1 1 90 90 90.

To create a completely new fragment, open any fragment and click ‘Clear All’. New atoms can then be added manually or with the ‘Use Selected Atoms’ button. The minimum requirements for a new fragment in the editor are the cell parameters, an atom and at least one restraint.

The button ‘Enter Rename Mode’ invokes a mode in which atoms are renamed simultaneously in the list of restraints in a consistent manner. The precondition for this mode is an unmodified atom list.

With the ‘Mail Fragment home’ button the user can email the currently opened fragment to the author if he or she wants to have it in the official database. The only prerequisite for this is a fully configured email program. The fragment should be from a high-quality low-temperature measurement and should not be involved in disorder, or it should be calculated using quantum chemical programs with reasonable quality. The absolute geometry of the fragment should not be overstrained, because the relative restraints applied by *DSR* let the data decide on the resulting bond lengths and angles. The alternative, a model from residual density peaks, would often be even worse.



```
REM DSR PUT TOLUENE WITH C6 C4 C2 ON Q3 Q5 Q4 PART -1 OCC
20.5 RESI TOL
```

```
fragment:          target:
C6                 Q3
C4   1,2: 2.40 Å   Q5  1,2: 2.23 Å
C2   1,3: 2.41 Å   Q4  1,3: 2.39 Å
```

-1 PART
2 Free Variable
0.5 Occupancy
TOL Residue Class

Figure 4
(Left) Selection of the target position for the fragment (red crosses). (Right) Preview of the fragment fit (red crosses).

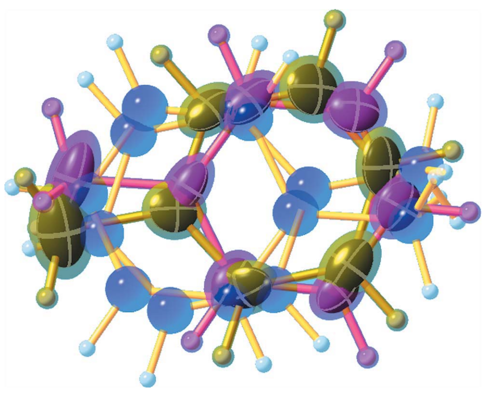


Figure 5
The finished disorder model of toluene around an inversion centre (symmetry-generated atoms in pale blue).

4. Disorder on special positions

One of the more subtle changes since the previous publication concerns the modelling of disorder around special positions. It often happens that solvents co-crystallize on special positions. For example, toluene is often located on an inversion centre, but toluene has no inversion centre (Müller *et al.*, 2006). In addition, these positions are often voids between larger

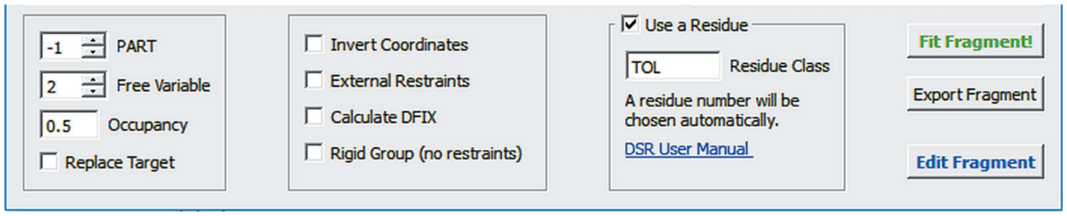

-1 PART
2 Free Variable
0.5 Occupancy
☐ Replace Target
☐ Invert Coordinates
☐ External Restraints
☐ Calculate DFIX
☐ Rigid Group (no restraints)
☑ Use a Residue
TOL Residue Class
A residue number will be chosen automatically.
DSR User Manual

Figure 6
Options used for the first part of the toluene disorder model.

molecules, where the solvent molecule is not densely packed and can move a little. To describe the model of a molecule around a special position with *SHELXL*, one always has two options. The first is to model only a fraction of the molecule in the asymmetric unit with an occupancy of 1. Often, however, solvent molecules do not follow the symmetry of the special position exactly. In this case, bond lengths from atoms generated by symmetry elements become inconsistent. It is then advisable to model the whole molecule with fractional occupancy according to the local symmetry.

The following example explains the modelling of a toluene molecule, which shows disorder around an inversion centre in a head-to-tail arrangement and additionally performs a rotational motion in the ring plane (Kratzert & Böttcher, 2018; CSD refcode HERJUM).

Owing to the inversion centre, the Fourier density map and the resulting electron-density peaks (q peaks) appear as the characteristic arrangement shown in Fig. 4, provided that the 'Grow q peaks' option is activated in *ShelXle* (available since version 877). Being able to select symmetry-generated q peaks as the target was crucial in allowing modelling of such disorder. Otherwise, sometimes too few q peaks would be displayed for a target position of the fragment placement on special positions. To model this disorder, it is now necessary to choose three of the q peaks as target (red crosses in Fig. 4) and three corresponding atoms of the fragment. These six coordinates will later be fitted onto each other during the fragment transfer. Inversion symmetry also implies that the modelled fragment has to have an occupancy of 0.5 (because we model one complete molecule instead of one half).

To describe the additional rotational movement of the toluene near its ring centre, displayed in Fig. 5, we also have to tie the occupancy to a so-called free variable in *SHELXL* (Fig. 6) and add a second fragment to describe two distinct positions disordered over the whole crystal lattice.

The 'PART' option in Fig. 6 controls the treatment of the connectivity table in *SHELXL*. A negative part means that special-position constraints are disabled; otherwise, some of the toluene atoms would be forced to lie on the inversion centre. Moreover, each part of the toluene gets different part numbers of −1 and −2 to prevent *SHELXL* from assuming bonds between neighbouring atoms of the different fragments overlapping with each other and also to prevent bonds to symmetry-generated atoms.

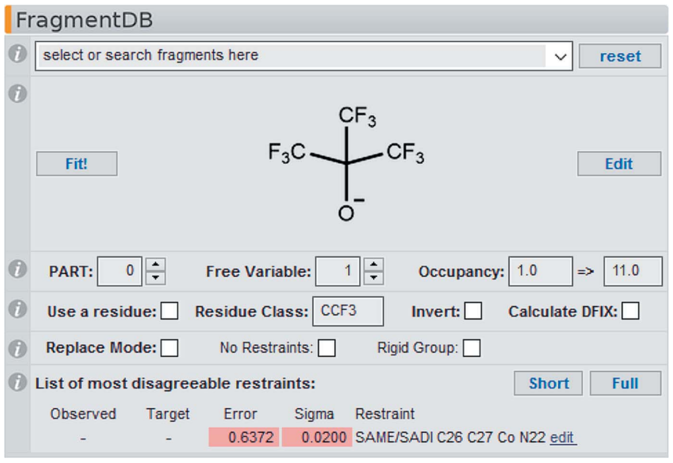
During each fragment transfer, *DSR* also applies stereochemical restraints to stabilize the model in a physically meaningful minimum. The more diffuse the disorder, the more essential are restraints for the least-squares refinement to find the correct minimum. For toluene in this example, the following restraints would be introduced:

```
SADI_TOL C2 C3 C3 C4 C4 C5 C5 C6 C6 C7 C7 C2
SADI_TOL 0.04 C2 C6 C2 C4 C7 C5 C3 C7 C4 C6 C3 C5
DFIX_TOL 1.51 1 C2
SADI_TOL 0.04 1 C7 C1 C3
FLAT_TOL C1 > C7
SIMU_TOL C1 > C7
RIGU_TOL C1 > C7
SAME_TOL C1 > C7
```

Modelling the toluene molecule of the above example in an atom-by-atom manner would require many more steps and would be much more error prone. However, the restraints from *DSR* cannot always be perfect. Therefore, the example model can be improved by replacing DFIX with a SADI restraint and `SIMU_TOL C1 > C7` with `SIMU C1_1 > C7_1 C1_2 > C7_2` in this case (_1 and _2 mean residues 1 and 2, respectively).

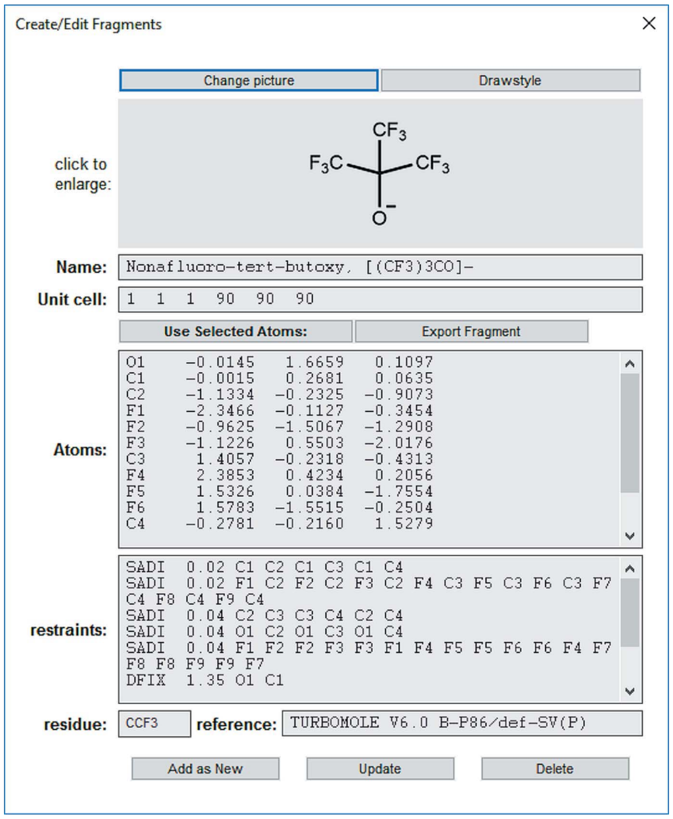
5. *OLEX2* plugin (*FragmentDB*)

Although *DSR* will work together with *OLEX2*, this does require a native plugin in *OLEX2* (Fig. 7). Thanks to the excellent code base of *OLEX2*, porting *DSR* is easy to achieve. The functionality is similar but has three main differences. In contrast with the text database of *DSR*, the *FragmentDB*


FragmentDB
select or search fragments here
PART: 0 Free Variable: 1 Occupancy: 1.0 => 11.0
Use a residue: ☐ Residue Class: CCF3 Invert: ☐ Calculate DFIX: ☐
Replace Mode: ☐ No Restraints: ☐ Rigid Group: ☐
List of most disagreeable restraints:

Observed	Target	Error	Sigma	Restraint
-	-	0.6372	0.0200	SAME/SADI C26 C27 Co N22 edit

Figure 7
The graphical interface of *FragmentDB* in *OLEX2*.



Name: Nonafluoro-tert-butoxy, [(CF3)3CO]-

Unit cell: 1 1 1 90 90 90

Atoms:

```
O1    -0.0145   1.6659   0.1097
C1    -0.0015   0.2681   0.0635
C2    -1.1334  -0.2325  -0.9073
F1    -2.3466  -0.1127  -0.3454
F2    -0.9625  -1.5067  -1.2908
F3    -1.1226   0.5503  -2.0176
C3     1.4057  -0.2318  -0.4313
F4     2.3853   0.4234   0.2056
F5     1.5326   0.0384  -1.7554
F6     1.5783  -1.5515  -0.2504
C4    -0.2781  -0.2160   1.5279
```

restraints:

```
SADI  0.02 C1 C2 C1 C3 C1 C4
SADI  0.02 F1 C2 F2 C2 F3 C2 F4 C3 F5 C3 F6 C3 F7
C4 F8 C4 F9 C4
SADI  0.04 C2 C3 C3 C4 C2 C4
SADI  0.04 O1 C2 O1 C3 O1 C4
SADI  0.04 F1 F2 F2 F3 F3 F1 F4 F5 F5 F6 F6 F4 F7
F8 F8 F9 F9 F7
DFIX  1.35 O1 C1
```

residue: CCF3 reference: TURBOMOLE V6.0 B-P86/def-SV(P)

Figure 8
The fragment editor window of the *OLEX2 FragmentDB* plugin.

plugin stores the fragment data in an SQLite (http://www.sqlite.org/copyright.html) database. The second difference is the representation of the fragment. *OLEX2* has no way of displaying a second three-dimensional representation of a molecule. Therefore, the molecular fragment is represented by a previously generated image from the fragment database.

The third prominent difference between *FragmentDB* and *DSR* is the fitting procedure of the fragment from the database to the desired position in the crystal structure. *OLEX2* has always had a graphical functionality to perform this fit. Instead of defining pairs of atoms beforehand, the user clicks on pairs of atoms, and the fragment moves into the desired position after every pair of clicks on fragment and target atoms.

The fragment editor of *FragmentDB* displayed in Fig. 8 has similar functionality to the above-mentioned editor in *ShelXle*, but it cannot check the validity of the user input while it is being typed in and there is no renaming mode for simultaneous renaming of atoms and restraints.

6. Restraints

The main purpose of *DSR* is to fit molecular fragments into a crystal structure, but when disorder is involved it is important to consider restraints. A restraint is incorporated into the least-squares refinement as if it were an additional experimental observation. Information about the molecular geometry is added to the sum of the weighted square difference between the observed and calculated intensities. It can be either an absolute value (DFIX, DANG and FLAT restraints) or a relative distance in the form of a SADI restraint (Watkin, 2008, 1994; Becker & Müller, 2017; Müller, 2009; Müller *et al.*, 2006). In particular, when disorder is involved and the exact position of each atom becomes uncertain, the additional geometric information from restraints is crucial. To suppress model bias as much as possible, *DSR* mostly uses relative restraints. Either restraints are taken from the database, which are carefully generated by hand, or in the case of residues it also introduces a `SAME_name [first atom] > [last atom]` restraint to make the 1,2- and 1,3-distances between equal residues similar. The assumption here is that residues with the same name should have similar bond lengths and bond angles.

If a model using relative restraints is not stable enough, *DSR* can generate restraints from the geometry of the fitted fragment. These restraints should only be used as last resort, for example in the case of low resolution. In order to generate these restraints, *DSR* uses the *NetworkX* library to build a molecular graph from the fragment coordinates. The graph is then searched for 1,2- and 1,3-distances, as well as ring systems. Rings are cut into overlapping atomic pieces of four atoms and their flatness is assessed by calculating the volume of the tetrahedron built up from each four-atom piece. Sufficiently flat rings (those with a near-zero volume, <0.085 Å^3) are then restrained to be flat with the FLAT restraint. If directly connected atoms lie on the ring plane, they are also added to the FLAT restraints.

In the automatic generation of restraints, however, it must be clear that they can only be as good as the geometry of the basal fragment specified. This disadvantage does not apply to the relative restraints. But in any case, users should not blindly assume any restraint to be valid in every case.

6.1. Restraint validation

Applying relative restraints is a good way to stabilize a structure model during refinement. However, what happens if the applied restraints have logical errors? It is very likely that a handwritten list of restraints, *e.g.* for C_{70} fullerene, contains typing errors. That is why *DSR* has some internal checks to avoid errors in restraints and it has enough information within itself to do so. The SADI restraints contain information on which atoms are pairs of equal distance. Thus, their distance and the corresponding standard deviation are also known from the coordinates. There are now two possibilities for error. The first is that the user has mistyped either one or more atom names. This error will result in a low accuracy, because there will be an unusually long distance or just a non-existing atom. On the other hand, if the overall distribution of the distances is broad, the precision is low.

DSR assumes (from empirical tests) that the mean distances of atom pairs in one SADI restraint should be the same within 2.5σ and the standard deviation of the distances should be below 0.065 Å. If the standard deviation is too large, a Nalimov test (Nalimov & Williams, 1963) is performed to warn the user about the suspicious atom pair. To give direct feedback about wrong restraints in the database, these precautions run if the GUI opens a database entry or if the command-line version shows the list of fragments. For example, if the user invents restraints for benzene to make the 1,2-distances equal inside a 0.02 Å standard deviation, the corresponding restraint would be

```
  SADI 0.02 C1 C2 C2 C3 C3 C4 C4 C5 C5 C6 C6 C1
```

Accidentally changing the first C2 to C3 implies a much longer distance for C1 to C3 instead of C1 to C2 (2.4 Å instead of 1.4 Å). This potential error would generate a warning in *DSR* and can then be the subject of inspections:

```
  *** Suspicious deviation of atom pair 'C1 C3'
(2.373 A, median: 1.368) after line 192 in
dsr_user_db.txt ***
  *** SADI 0.02 C1 C3 C2 C3 C3 C4 C4 C5 C5 C6 C6 C1
... ***
```

7. Log file

In the case of an error occurring during the run of *DSR*, it is useful to have a log file showing the last few actions. Therefore, *DSR* creates a log file during every run in the current working directory. The log file is always called `dsr-log.lst` and contains the Python version in its first line and the complete program output in the subsequent lines:

```
Python version: 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59)
[MSC v.1500 32 bit (Intel)]

------------------------- D S R - v207 --------------------------
No residue number was given. Using residue number 2.
Inserting toluene into res File.
Source atoms: C4, C6, C2
Target atoms: Q3, Q4, Q5
RESI instruction is enabled. Leaving atom numbers as they are.
Fragment atom names: C1, C2, C3, C4, C5, C6, C7

Already existing restraints for residue "TOL" were not applied
again.
-------------------------------------------------------------------
 Running SHELXL with "C:/bn/SXTL/shelxl_2018_1.exe -b3000 P21n_a"
and "L.S. 0"
 SHELXL Version 2018/1
 wR2  = 0.1926
 GooF = 1.6170
 R1   = 0.0701
Runtime: 0.5 s
DSR run complete.
```

8. Trifluoromethyl groups

The Cambridge Structural Database (CSD; Groom *et al.*, 2016) contains roughly 38 000 structures with CF_3 groups, of which about half are rotationally disordered. This observation is hardly surprising considering the low rotational barrier of $X{-}CF_3$ groups ($\sim$4–25 kJ mol^{-1}; Iijima *et al.*, 1992; Wang *et al.*, 2006). The large number of disordered CF_3 groups implies that it is useful to have a tool to help with this task. The current version of *DSR* is able to generate a model for disordered CF_3 groups automatically; the user only chooses the disorder type and position of the group. In general, CF_3 groups have three common types of rotational disorder (Fig. 9): staggered with 60° rotation, staggered with additional movement of the entire group and thus a split of the central carbon atom over two positions along the principal axis of its thermal ellipsoid, and finally a rotation of 30°, where three rotational positions are modelled.

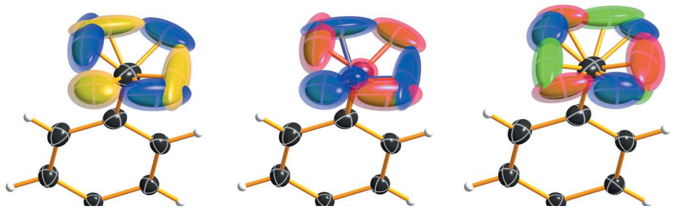


Figure 9
Rotational disorder of a CF_3 group, modelled with the three possibilities of *DSR*. (Left) The staggered model on two positions. (Centre) Two positions with a split of the central carbon atom. (Right) Rotation of 30° on three positions.

In *ShelXle*, the fitting procedure is now performed by selecting the central carbon atom of a CF_3 group and selecting the desired CF_3 model in the fragment list. The modelling process may then delete previously attached fluorine atoms, detect the positions of the fluorine atoms and place a model with a complete set of restraints for the disorder (Fig. 9).

9. Software updates

The development of *DSR* is a constant evolution. Therefore, it is necessary to inform users about updates and to make them as easily accessible as possible. The solution is a self-updating process: *DSR* asks a web server for the latest version during start-up. The update procedure retrieves the new version from the web server and overwrites the files in the current installation directory. Checksums make sure the files are intact. Nevertheless, it is always possible to install the latest setup file obtained from the web site.

10. Conclusions

The addition of graphical user interfaces to *DSR* has simplified and improved its usage. Large structures with a large amount of solvent or ligand disorder that were challenging before are now only a matter of computing time. Nevertheless, there is still room for improvement, *e.g.* by implementing ways to generate conformers of the fragment while it is being transferred. Significant further improvements of the method might only be achieved with a completely new approach. We would be happy to consider any external contribution or cooperation.

DSR can be obtained free of charge at https://www.xs3.uni-freiburg.de/research/dsr.

Acknowledgements

DK is very grateful to all bug reporters, who gave numerous helpful hints for improvement. We also thank the referees for valuable comments.

References

Becker, S. & Müller, P. (2017). *Chem. Eur. J.* **23**, 7081–7086.

Cooper, R. I., Thompson, A. L. & Watkin, D. J. (2010). *J. Appl. Cryst.* **43**, 1100–1107.

Dolomanov, O. V., Bourhis, L. J., Gildea, R. J., Howard, J. A. K. & Puschmann, H. (2009). *J. Appl. Cryst.* **42**, 339–341.

Farrugia, L. J. (2012). *J. Appl. Cryst.* **45**, 849–854.

Groom, C. R., Bruno, I. J., Lightfoot, M. P. & Ward, S. C. (2016). *Acta Cryst.* B**72**, 171–179.

Hübschle, C. B., Sheldrick, G. M. & Dittrich, B. (2011). *J. Appl. Cryst.* **44**, 1281–1284.

Iijima, K., Tanaka, Y. & Onuma, S. (1992). *J. Mol. Struct.* **268**, 315–318.

Kratzert, D. & Böttcher, T. (2018). Personal communication (refcode 1822173). CCDC, Union Road, Cambridge, UK.

Kratzert, D., Holstein, J. J. & Krossing, I. (2015). *J. Appl. Cryst.* **48**, 933–938.

Lichtenthaler, M. R., Stahl, F., Kratzert, D., Heidinger, L., Schleicher, E., Hamann, J., Himmel, D., Weber, S. & Krossing, I. (2015). *Nat. Commun.* **6**, 8288.

McArdle, P. (2017). *J. Appl. Cryst.* **50**, 320–326.

Müller, P. (2009). *Crystallogr. Rev.* **15**, 57–83.

Müller, P., Herbst-Irmer, R., Spek, A. L., Schneider, T. R. & Sawaya, M. R. (2006). *Crystal Structure Refinement – A Crystallographer's Guide to SHELXL*, IUCr Texts on Crystallography, Vol. 8. Oxford University Press.

Nalimov, V. V. & Williams, M. (1963). *Application of Mathematical Statistics to Chemical Analysis:* Oxford: Pergamon Press.

Petříček, V., Dušek, M. & Palatinus, L. (2014). *Z. Kristallogr.* **229**, 345–352.

Sheldrick, G. M. (2015). *Acta Cryst.* C**71**, 3–8.

Wang, X., Mallory, F. B., Mallory, C. W., Beckmann, P. A., Rheingold, A. L. & Francl, M. M. (2006). *J. Phys. Chem. A*, **110**, 3954–3960.

Watkin, D. (1994). *Acta Cryst.* A**50**, 411–437.

Watkin, D. (2008). *J. Appl. Cryst.* **41**, 491–522.