An efficient treatment of ring conformations during molecular crystal structure determination from powder diffraction data

Supplementary information for "An efficient treatment of ring conformations during molecular crystal structure determination from powder diffraction data"
Spillman, Shankland & Shankland, *CrystEngComm* 2022

Contents

1. Implementation of restraints

Restraints are enforced during the local optimisation component of *GALLOP* by including them as penalty terms, which are minimised together with the intensity χ^2. No restraints are applied during the particle swarm optimisation step; this allows local minima that may be introduced by the addition of the restraints to be escaped.

Distances

The simplest restraint directs a pair of atoms to sit a defined distance apart. The atoms involved can be from any fragment within the asymmetric unit, allowing for both intra- and intermolecular restraints. The distance penalty term, D is defined as:

$$D = \left(\left|\vec{u}_{ij}\right| - \delta_{ij}\right)^2$$

Where $\left|\vec{u}_{ij}\right|$ is the magnitude of the vector pointing from atom i to atom j, and δ_{ij} is the distance provided as the restraint. The penalty term D can take values ranging from 0 to ∞.

Angles

The angle between any two interatomic vectors can also be used as a restraint. Generally, this will be a bond angle, though this is not a requirement: any two interatomic vectors within the asymmetric unit can be used as input. The angle penalty term, A is defined as:

$$A = \left(\frac{\vec{u}_{ij} \cdot \vec{u}_{kl}}{\left|\vec{u}_{ij}\right|\left|\vec{u}_{kl}\right|} - \cos\alpha_{ijkl}\right)^2$$

Where $\vec{u}_{ij}$ is the vector pointing from atom i to atom j, $\vec{u}_{kl}$ is the vector pointing from atom k to atom l and α_{ijkl} is the angle between the interatomic vectors supplied by the user. For computational efficiency, the cosine of the user-supplied angle α_{ijkl} is calculated in advance and stored, which allows for rapid comparison with the observed atomic coordinates. By setting i and k to be the same atom, normal bond-angles (*i.e.* the angle between bond $\vec{u}_{ij}$ and bond $\vec{u}_{il}$) can be used as the basis for restraints. The penalty term A can take values ranging from 0 to 4.

Torsions

The torsion angle between two intersecting planes can also be used as a restraint. In general, it will be a proper torsion angle, though this is not a requirement: any four atoms from any of the fragments in the asymmetric unit can be used as input. The torsion penalty term, T is defined as:

$$T = \left(\frac{\left|\vec{u}_{jk}\right| \vec{u}_{ij} \cdot \left(\vec{u}_{jk} \times \vec{u}_{kl}\right)}{\left|\vec{u}_{ij} \times \vec{u}_{jk}\right| \left|\vec{u}_{jk} \times \vec{u}_{kl}\right|} - \sin \tau_{ijkl} \right)^2 + \left(\frac{\left(\vec{u}_{ij} \times \vec{u}_{jk}\right) \cdot \left(\vec{u}_{jk} \times \vec{u}_{kl}\right)}{\left|\vec{u}_{ij} \times \vec{u}_{jk}\right| \left|\vec{u}_{jk} \times \vec{u}_{kl}\right|} - \cos \tau_{ijkl} \right)^2$$

Where $\vec{u}_{ij}$ is the vector pointing from atom i to atom j, $\vec{u}_{jk}$ is the vector pointing from atom j to atom k, $\vec{u}_{kl}$ is the vector pointing from atom k to atom l and τ_{ijkl} is the torsion angle, supplied by the user, between the planes formed by atoms i, j, k and atoms j, k, l. Again, for computational efficiency, the sine and cosine of the user-supplied angle are calculated in advance and stored. The penalty term T can take values ranging from 0 to 4.

2. Cost function

For best performance, the additional penalty terms in the cost function must be weighted appropriately relative to the intensity χ^2. Our approach is to globally scale the value of all restraint penalties by the value of χ^2 at each iteration, then further scale each individual restraint by a user-supplied weighting term. This allows the user to easily set the importance of each restraint relative to χ^2. A weight of 1 assigns equal importance to χ^2 and the restraint, weights less than 1 reduce the importance of the restraint relative to χ^2 and weights greater than 1 increase the relative importance.

Both the intensity χ^2 and the restraints are functions of the degrees of freedom of the structure. This must be taken into account when implementing this weighting scheme because *GALLOP* makes use of gradients for the local optimisation component of the algorithm. Gradients obtained from a naïve implementation would result in markedly different optimisation behaviour than expected[1]. To avoid this issue, our implementation of the cost function makes use of a gradient-free copy of χ^2, which acts purely as a numerical scaling factor.

In a scenario where a user has supplied *m* distance restraints, *n* angle restraints and *o* torsion restraints, the cost function *C* is defined as:

$$C(\boldsymbol{x}) = \chi^2(\boldsymbol{x}) + \chi^2_{copy}\left(\sum_{d=1}^{m} \omega_d D_d(\boldsymbol{x}) + \sum_{a=1}^{n} \omega_a A_a(\boldsymbol{x}) + \sum_{t=1}^{o} \omega_t T_t(\boldsymbol{x})\right)$$

Where $\boldsymbol{x}$ represents the structural degrees of freedom, χ^2_{copy} is a gradient-free copy of $\chi^2(\boldsymbol{x})$ and ω_d, ω_a and ω_t are the user-supplied weights for each of the distance, angle and torsion restraints respectively.

For the work reported here, we set all weights to 1.

[1] A naïve cost-function implementation that scales the restraints directly by $\chi^2(\boldsymbol{x})$ gives:

$$C(\boldsymbol{x}) = \chi^2(\boldsymbol{x})(1 + R(\boldsymbol{x}))$$

where $R(\boldsymbol{x})$ is the sum of all weighted restraint penalty terms. Via the product rule, the gradient of this function is:

$$\nabla C(\boldsymbol{x}) = \big(1 + R(\boldsymbol{x})\big)\nabla\chi^2(\boldsymbol{x}) + \chi^2(\boldsymbol{x})\nabla R(\boldsymbol{x})$$

As can be seen, whilst the gradient of the restraint terms has been scaled by χ^2 as desired, this also results in scaling the gradient of χ^2 by a factor of $1 + R(\boldsymbol{x})$. However, by using a gradient-free copy of χ^2 as defined above, the cost function and its associated gradient are given by:

$$C(\boldsymbol{x}) = \chi^2(\boldsymbol{x}) + \chi^2_{\text{copy}} R(\boldsymbol{x})$$

$$\nabla C(\boldsymbol{x}) = \nabla\chi^2(\boldsymbol{x}) + \chi^2_{\text{copy}} \nabla R(\boldsymbol{x})$$

This results in the desired optimisation behaviour.

3. Differences with other SDPD programs

Other programs for SDPD such as *FOX* and *TALP* also allow users to define restraints. However, there are some differences with the approach taken by *GALLOP* for both defining and refining the restraints.

TALP minimises the following cost function:

$$M = S_Y + kS_R$$

where S_Y is the residual measuring the difference between observed and calculated intensities, k is a scaling factor and S_R is the residual involving restraints, which are all defined in terms of distances. The restraint residual is given by:

$$S_R = \sum_j \sigma_j^{-2} \left(d_{obs,j} - d_j\right)^2$$

where σ_j is the estimated variance of distance j, $d_{obs,j}$ is the user supplied distance and d_j is the distance obtained during refinement. *TALP* uses interatomic distances to enforce bond angle and torsion angle restraints, and hence there is no requirement to calculate any angular values.

FOX takes a slightly different approach. For each type of restraint, the associated χ^2 is calculated as follows:

$$\text{if } r \in [r_0 - \delta; r_0 + \delta] \qquad \chi_r^2 = 0$$

$$\text{if } r \le r_0 - \delta \qquad \chi_r^2 = \left(\frac{r - (r_0 - \delta)}{\sigma}\right)^2$$

$$\text{if } r \ge r_0 + \delta \qquad \chi_r^2 = \left(\frac{r - (r_0 + \delta)}{\sigma}\right)^2$$

where r is either a bond length, bond angle or torsion angle, δ defines an inner range around the user-supplied value r_0 in which no penalties are applied, and σ defines how quickly the cost rises when r departs from the allowed range. As such, for bond angle and torsion angle restraints, the angular values must be computed, necessitating the use of trigonometric functions.

During optimisation, *FOX* decouples the optimisation of the restraints and the optimisation of the diffraction χ^2; a Boltzmann-type probability is used to accept or reject molecular changes via:

$$\exp\left(-\frac{\chi_r^2}{T}\right)$$

If a change is accepted, *FOX* then moves on to calculate the diffraction χ^2 value. The restraint temperature T is dynamically modified to accept 70 % of the new molecular configurations proposed during the optimisation procedure.

Unlike *FOX*, *GALLOP* does not directly calculate angles for use in restraints, but rather determines the sines and cosines of the angles directly from the interatomic vectors. This avoids the use of trigonometric functions, which are relatively slow to evaluate on GPUs.

However, unlike the *TALP* distance-based approach, users still supply angular values as the restraint inputs for bond angles and torsion angles, which may be more intuitive for users familiar with Z-matrix molecular descriptors.

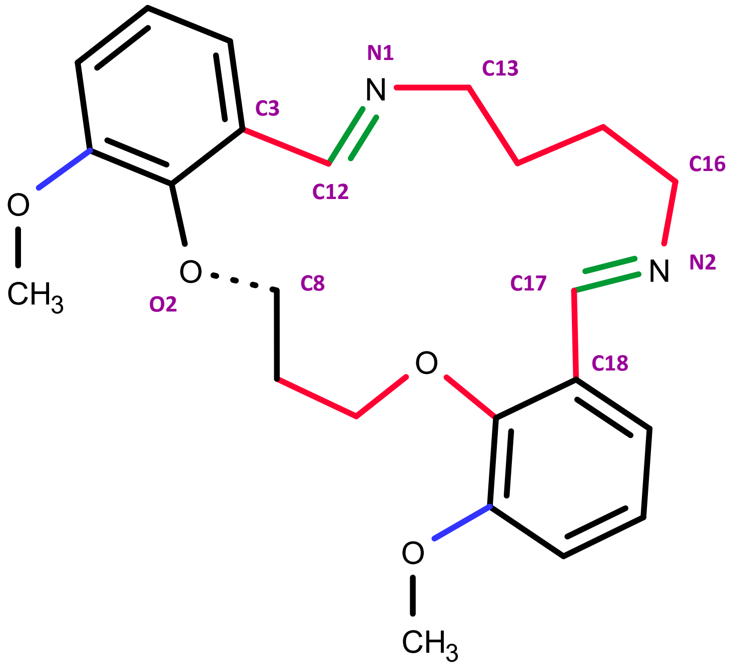
Finally, in *GALLOP* the restraints and the diffraction χ^2 are refined simultaneously. The *FOX* authors argue against this approach in a gradient-free global optimisation setting. However, as *GALLOP* makes use of gradient-based local optimisation of the cost function, these arguments do not necessarily apply to the approach described here.

<u>4. Restraints used in this work</u>

The tables below detail the atom labels and values used for the restraints reported in this work. Atom labels are in accordance with the CSD entry atom labels. Distances were measured using Mercury and rounded to two decimal places. Torsion angle restraint values for double bonds were set to 180°.

IJUXUI

The figure below shows the molecular structure of IJUXUI, with atom labels shown in purple for atoms involved in distance and torsion angle restraints. Atom labels are derived from the CSD entry.

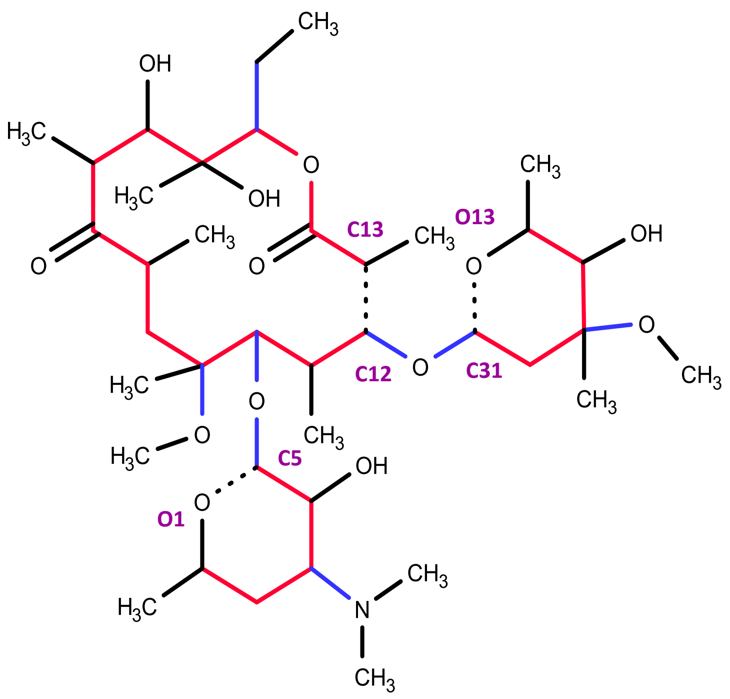


Distance / Å	Atom 1	Atom 2
1.44	O2	C8

Torsion / °	Atom 1	Atom 2	Atom 3	Atom 4
180	C13	N1	C12	C3
180	C16	N2	C17	C18

LAQSON01

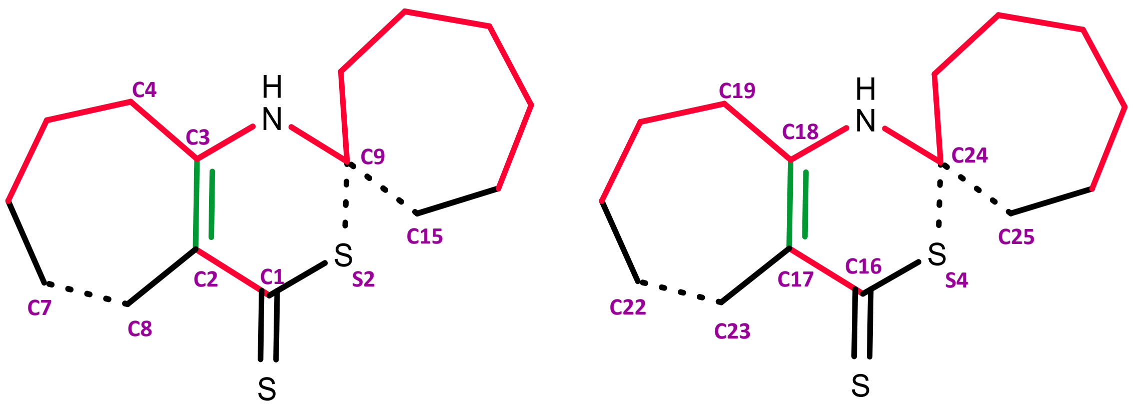
The figure below shows the molecular structure of LAQSON01, with atom labels shown in purple for atoms involved in distance restraints. Atom labels are derived from the CSD entry.



Distance / Å	Atom 1	Atom 2
1.42	O1	C5
1.56	C12	C13
1.42	O13	C31

YIXSII

The figure below shows the molecular structures of the two independent molecules in the asymmetric unit of YIXSII, with atom labels shown in orange for atoms involved in distance and torsion angle restraints. Atom labels are derived from the CSD entry.

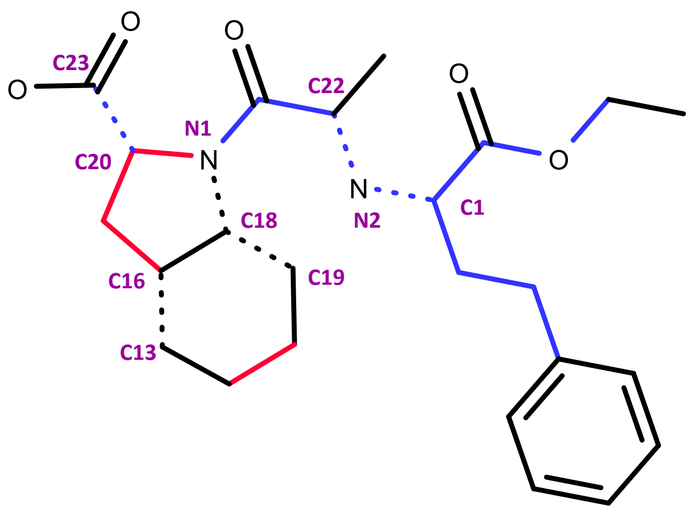


Distance / Å	Atom 1	Atom 2
1.49	C7	C8
1.83	S2	C9
1.55	C9	C15
1.52	C22	C23
1.84	S4	C24
1.56	C24	C25

Torsion / °	Atom 1	Atom 2	Atom 3	Atom 4
180	C1	C2	C3	C4
180	C16	C17	C18	C19

IQISAE01

The figure below shows the molecular structure of IJUXUI, with atom labels shown in purple for atoms involved in distance restraints. Atom labels are derived from the CSD entry.



Distance / Å	Atom 1	Atom 2
1.47	C1	N2
1.50	N2	C22
1.53	C20	C23
1.46	N1	C18
1.49	C13	C16
1.53	C18	C19

5. Results for all GALLOP runs: those for the 20 iterations are those reported in Table 2 of the publication

Structure	*GALLOP* Iterations	Cut	Restrained	Success / %	Time per run / mins	Average RMSD / Å	Best RMSD / Å
IJUXUI	10	✗	-	100	11.1	0.073	0.073
1 × *N*=100		✓	✗	43	12.3	0.150	0.110
		✓	✓	100	12.8	0.138	0.110
	20	✗	-	100	24.7	0.073	0.073
		✓	✗	48	24.7	0.138	0.108
		✓	✓	100	25.2	0.136	0.108
LAQSON01	10	✗	-	100	10.3	0.077	0.076
5 × *N*=20		✓	✗	0	10.3	-	-
		✓	✓	6	10.4	0.152	0.115
	20	✗	-	100	20.6	0.077	0.076
		✓	✗	0	20.6	-	-
		✓	✓	10	20.7	0.118	0.114
YIXSII	10	✗	-	75	8.7	0.047	0.046
3 × *N*=34		✓	✗	0	10.5	-	-
		✓	✓	93	10.6	0.142	0.109
	20	✗	-	86	17.3	0.047	0.046
		✓	✗	0	21.0	-	-
		✓	✓	100	21.2	0.142	0.109
IQISAE01	10	✗	-	39	8.3	0.121	0.119
4 × *N*=25		✓	✗	0	8.3	-	-
		✓	✓	8	8.3	0.178	0.168
	20	✗	-	44	16.3	0.120	0.119
		✓	✗	0	16.5	-	-
		✓	✓	17	16.6	0.187	0.165

Cut = did model incorporate cut bonds?; Restrained = did model incorporate restraints?; Success = frequency of success in achieving correct structure solution; Time per run = time taken for a single *GALLOP* run with *N* swarms – for three of the structures, multiple runs were required; RMSD = root mean square deviation of answer from known REFCODE structure, obtained by *Mercury* (15/15 molecules in common, 20% tolerances; H atoms ignored) rounded to 3 decimal places, *N*=number of independent swarms that were simultaneously accommodated in GPU memory.