SOMoRe
Search and Optimization for Molecular Replacement
Version 0.93 β
March 2003

Diane C. Jamrog*
(revised May 12, 2003)

Development of this program was supported in part by the W. M. Keck Center for
Computational Biology (NSF RTG BIR-94-13229, NLM RTG 2T15LM07093, NLM 5
T15 LM07093), NSF Grant DMS-9973339 and NIH GM64589.

*Department of Computational and Applied Mathematics, Rice University, Houston, Texas,
USA, djamrog@alumni.rice.edu.
Contents

1 A Very Brief Overview of MR Approaches ................................................. 4
2 The New Global Optimization Strategy ................................................. 4
3 SOMoRe ................................................................................................. 4
   3.1 Input ................................................................................................. 5
   3.2 Output ................................................................................................. 5
   3.3 Compiling the code ........................................................................... 6
4 Suggested Use of SOMoRe ................................................................. 6
5 Auto Mode .............................................................................................. 6
6 Key Words ............................................................................................... 7
   6.1 Key words from Qs ................................................................. 7
   6.1.1 Target structure keywords ..................................................... 7
   6.1.2 Observed data selection keywords .......................................... 8
   6.1.3 Internal calculus keywords .................................................... 9
   6.1.4 Some miscellaneous keywords ............................................. 11
   6.2 Key words specific to SOMoRe .................................................... 11
   6.2.1 Grid search keywords ............................................................ 11
   6.2.2 Local optimization keywords ............................................... 14
   6.2.3 Packing check keyword ........................................................... 14
   6.3 Keyword rules ................................................................................ 15
7 Example Input File ................................................................................. 15
8 More on Grid Searches .......................................................................... 15
   8.1 Estimating the global search run time ......................................... 15
   8.2 Global grid searches ........................................................................ 15
   8.3 Fixed model searches ..................................................................... 17
   8.4 Sequential searches ....................................................................... 18
   8.5 Local searches ................................................................................ 18
9 More on Multi-start Local Optimization ........................................... 19
   9.1 The local optimization method BFGS ......................................... 19
   9.2 Convergence results ....................................................................... 19
10 Caveat About Scaling .......................................................................... 20
11 Additional Codes ................................................................................ 21
   11.1 Creating additional PDB files: writepdb ....................................... 21
   11.2 Generating packing results: pack ................................................. 22
12 Future Work

A Determining the closest symmetry mate

B Unix Script Files
1 A Very Brief Overview of MR Approaches

In 1962, Rossman and Blow proposed that the molecular replacement (MR) problem be solved first by a three-dimensional search for optimal orientation and then a three-dimensional search for optimal translation of the oriented model protein [16]. However, there are drawbacks to separately optimizing the orientation and position of the model protein. Specifically, there are classes of MR problems that either defeat or seriously challenge such traditional methods; see [1, 4, 7, 17], for example.

To avoid the drawbacks associated with separately optimizing the MR variables, six-dimensional (6D) optimization approaches are now being proposed because improvements in computational resources have allowed such approaches to be feasible. In general, if there is one molecule in the asymmetric unit, then the MR problem is a 6D optimization problem. SOMoRe is another 6D approach. However, its methodology is quite different from current 6D approaches, which include parallelized, fine, 6D grid searches [17] and stochastic optimization approaches that include simulated annealing [7] and genetic algorithms [4, 12].

2 The New Global Optimization Strategy

The new strategy has two major components: a coarse, 6D global search and multi-start optimization. First, a coarse global search is performed on a surrogate function that is computed from primarily low-resolution intensities. As a result, the surrogate function will be smooth in comparison to a target function computed from primarily high-resolution intensities; see [10] for more information. During the grid search, SOMoRe keeps track of the lowest valued grid points to identify regions of the MR variable space where solutions are likely to occur. Next, multi-start local optimization is performed using these lowest valued grid points as starting points. If a starting point is close enough to a global minimum, then the sequence of iterates generated by the local optimization algorithm will converge to the minimum. Finally, all 6D local minima are ranked according to their function values and packing checks can be performed. If there is significant contrast between the function values of the lowest valued local minima and the remaining local minima, then this is a good indication that a global minimum has been found.

3 SOMoRe

To implement the new strategy, a C program called SOMoRe was developed by modifying the MR program Queen of Spades (Qs) that was developed by Glykos and Kokkinidis [7, 8]. Qs is freely available software which can be downloaded from Nicholas Glykos’ web page:

http://origin.imbb.forth.gr:8888/~glykos/Qs.html
In the same spirit, SOMoRe is also freely available for non-profit use.

Basically, the global optimization approach of Qs (the simulated annealing component) was removed and replaced with our deterministic, global optimization approach. We did retain the components of Qs that are responsible for calculating the fast Fourier transform (FFT) of the model protein, calculating the model's intensities, evaluating the target functions, and reading in the three essential input files: a file containing keywords and parameters, a file containing the observed data, and a file containing the coordinates of the model protein. Therefore, much of the front end of SOMoRe is nearly identical to Qs. In addition, as for Qs, SOMoRe uses the FFTW library of functions [5], which is available for downloading from the web page: http://www.fftw.org.

3.1 Input

To run SOMoRe three files are required: an input file containing keywords that will determine SOMoRe's actions, a data file containing the magnitudes of the observed structure factors, $|F^o|$ (not the intensities $I^o = |F^o|^2$), and a file containing the coordinates of a model protein.

Note: Currently, when electron density is calculated for the model protein, the elements recognized are carbon, oxygen, nitrogen and sulfur. Phosphorus atoms are assigned to be sulfur, and all other "unknown" atoms are assigned to be carbon.

3.2 Output

The main output of SOMoRe from a global search or multi-start optimization is a PDB file and a list of six-dimensional points that can be used to create other PDB files using a separate program, called writepdb. For a global search, a PDB file is written for the lowest valued grid point found during the search. However, usually this PDB file will not be a solution to the MR problem, especially if the user takes our advice and uses a large, high-resolution cut-off to allow a coarse grid search and faster run time. Rather, the solution to the MR problem should appear upon examination of the local minima that result from optimizing the lowest valued, global search grid points.

After multi-start local optimization, a PDB file is written for the lowest valued local minimum. This should be a solution to the MR problem, provided that the model protein is reasonably close to the target structure. Of course writepdb can be used to create PDB files for any local minima or grid points. See section 11.1 for instructions on how to use writepdb.

The output for a global search will have the format:

$$\theta_1 \ \theta_2 \ \theta_3 \ x \ y \ z \ \text{fv}$$

or

$$\theta_1 \ \theta_2 \ \theta_3 \ x \ y \ z \ \text{fv} \ \text{free},$$

where $\text{fv}$ are target function values and $\text{free}$ are free function values. In comparison, if only function values are requested, then the output for a local optimization will be

$$\#a \ \theta_1 \ \theta_2 \ \theta_3 \ x \ y \ z \ \text{fv}$$

$$\#b \ \theta_1 \ \theta_2 \ \theta_3 \ x \ y \ z \ \text{fv} \ N,$$

5
where \# is some integer. The starting point is designated by an \textbf{a} and the local minimum by a \textbf{b}. The integer \textit{N} is the number of points generated by the optimization algorithm.

3.3 Compiling the code

After the FFTW library has been installed, read the comments in the \texttt{Makefile} provided and edit this file as directed. Then type ‘make all’ to create the executable programs for SOMoRe.

4 Suggested Use of SOMoRe

Based on the results presented in [10] and [11], we recommend that a global search be performed using all available data from the low resolution limit of the measured data set to 8Å, followed by multi-start local optimization using all data to a high resolution limit of 4Å. We also recommend using the objective function \( C(|F_o|, |F_c|) \).

For every test problem so far an 8Å search using \( C(|F_o|, |F_c|) \) was successful. In other words, the global grid search led to starting points that were sufficiently close to a global minimum to converge to it using the local optimization method BFGS. Of course, there are bound to be limitations to any approach as the model becomes more inaccurate.

If a MR solution cannot be found using \( C(|F_o|, |F_c|) \), then, of course, you should try using \( C(P_o, P_c) \). If both 8Å searches are not successful, then a 10Å global search followed by multi-start optimization can also be tried. Finally, if all else fails and computing time is not estimated to be prohibitive, then a higher high-resolution cut-off can be used to include more intensities in the computation of the low-resolution surrogate function.

5 Auto Mode

SOMoRe has an auto mode. In this mode, an input file is generated for the user, based on the general recommendations just given.

To run SOMoRe in this mode, type

\begin{verbatim}
  somore -auto 1
\end{verbatim}

where \texttt{1} denotes that only one copy of the model will be positioned in the asymmetric unit and used to calculate the model’s structure factors. To find the positions of multiple copies of the model in the asymmetric unit, see section 8.4 on sequential searches.

If SOMoRe is run in auto mode, then it will create the input file \texttt{somore_auto.in}, and then read it. This input file calls for a coarse global search. The model PDB file must be named \texttt{model.pdb} and the data file (containing structure factor magnitudes) must be named \texttt{data.hkl}.

After the global search, the user can then change the keyword \texttt{GSEARCH} to \texttt{OPTIMIZE GlobalTopN} and change the resolution range (from 500 to 8Å to, for example, 500 to 4Å). And then type
to run multi-start local optimization on the lowest-valued grid points found during the coarse global search.

6 Key Words

Because much of the front end of SOMoRe is nearly identical to Qs, many of the keywords recognized by Qs are also recognized by SOMoRe. Below is a list of these keywords have the same effect in SOMoRe as in Qs.

6.1 Key words from Qs

We list the keywords below along with additional information, as well as suggestions about their use with SOMoRe. All of Qs's target structure related keywords, reflection selection keywords, and "internal calculus" related keywords are recognized by SOMoRe. For further clarification of these keywords, please see the manual for Qs [6]. (None of the "time", or "annealing schedule and move size" keywords used by Qs are recognized by SOMoRe.)

6.1.1 Target structure keywords

MOLEcules $n$ : specifies the number of search models in the asymmetric unit that are used to calculate $|F_h|^k$ that are compared to $|F_h|^k$. This number should be 1, unless the orientations and positions of the other copies of the model are specified by the keyword ORIENT (as many times as is needed).

MODEL <string> : specifies the name of the file that contains the atomic coordinates of the model, expressed in real space Cartesian coordinates not fractional coordinates.

DATA <string> : specifies the name of a data file containing columns of $h, k, l, |F_{hkl}|$, and $\sigma(F_{hkl})$.

CELL $a b c \alpha \beta \gamma$ : specifies the unit cell of the target structure from which the observed data was collected.

GROUP $n$ : specifies the space group number. This must be an integer, not the Hermann-Maugin symbol. Note that, if you are using a non-standard setting, you can define your own group operators using the SYMM keyword (as many times as needed, do not include identity), and using the TLIN keyword, you may define the translational search domain for the appropriate asymmetric unit or Cheshire cell.

SYMMetry : allows the user to specify the symmetry operators rather than using the keyword GROUP. For example, for the operator $1/2 + x, 1/2 - y, z$ use
SYMM

<table>
<thead>
<tr>
<th></th>
<th>1.0</th>
<th>0.0</th>
<th>0.0</th>
<th>0.50</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>-1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.50</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>-1.0</td>
<td>0.0</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The numbers must be floating point numbers (not, for example, 1/2). Also, the identity operator should not be given using this command.

6.1.2 Observed data selection keywords

RESolution low high : This keyword sets the low and high resolution cut-offs to select the data which will be used to compute the target function.

Note: For a global search, we suggest using 500 to 8Å (to define the surrogate function), that is, a large (or “low”), high-resolution cut-off. For multi-start local optimization, we suggest 500 to 4 Å. As a result, all the low-resolution magnitudes will be used to compute the target functions. We used these ranges for each test problem we investigated.

FREE f : This keyword instructs SOMoRe to randomly select \((100 \cdot f)\)% of the magnitudes that satisfy all other selection criteria and use these magnitudes to compute free values. In general, free values are used to cross validate low target function values. In this setting, we believe FREE should only be used to discriminate between local minima or possible solutions. For global search runs, when only low resolution data are being used, we do not recommend using FREE but rather to keep all the data in the working set. (The manual for Qs warns that free values are often computed from a small fraction of the overall data set measured and therefore, should not always be taken at “face value” [6].)

Note: we used a value of 0.10 when performing local optimization of global search results.

SIGMa_cutoff \(c_\sigma\) : All magnitudes such that \(|F^\sigma_h|/\sigma(|F_h^0|) < c_\sigma\) are excluded from the set of magnitudes used to compute the target function.

AMPLitude_cutoff \(c_a\) : All magnitudes such that \(|F_h^0| < c_a\) are excluded from the data set used to compute the target function. The default value for \(c_a\) is 1.0.

RANDOM_select f : This keyword instructs SOMoRe to randomly select \((100 \cdot f)\)% of the magnitudes that satisfy all other selection criteria and use these magnitudes to compute the target function.

Note: We did not need to use this keyword because the local optimization run times were reasonable even though all intensities between 500 and 4Å were used to compute the target function.

KEEP f : This keyword instructs SOMoRe to select the “brightest” or largest \((100 \cdot f)\)% of the magnitudes read in from the data file. The manual for Qs warns that
because this is a magnitude cut-off not an E-value cut-off that this keyword may effectively reduce the resolution of the target function [6].

Note: We do not recommend using this keyword because important information may also be contained in the weakest or smallest magnitudes.

USEValues $E_{\text{limit}}$: This keyword instructs SOMoRe to compute normalized structure factor amplitudes, and then use only those amplitudes such that $E_{\sigma} > E_{\text{limit}}$. In the Qs manual, Glykos notes that the E-value computations can be slow [6].

Note: We did not use E-values to solve any of the test problems.

RESFree $r_{\text{low}}$: This keyword instructs SOMoRe to exclude from the free set (the set of magnitudes used to compute free values) any magnitudes such that the resolution is lower than $r_{\text{low}}$. The Qs manual notes that very strong low-resolution magnitudes, for example, those that occur for 4-Å-helical bundles may be very useful to keep in the “working” set of magnitudes rather than the “free” set, but that this keyword may be “dangerous” [6].

For each test problem, I did not use any reflection selection keywords other than RESolution, FREE 0.1, and AMPLiitude_cutoff 0.0.

6.1.3 Internal calculus keywords

TARGET < CORR-1, CORR-2, R-FACTOR, >: Defines the target function used. CORR-1 is one minus the correlation coefficient between magnitudes, $1 - C(|F^o|, |F^c|)$, while CORR-2 is one minus the correlation coefficient between intensities, $1 - C(F^o, F^c)$. The correlation coefficient is

$$ C(F^o, F^c) = \frac{\sum_h (F^c_h - \langle F^c \rangle)(F^o_h - \langle F^o \rangle)}{\sqrt{\sum_h (F^c_h - \langle F^c \rangle)^2} \cdot \sqrt{\sum_h (F^o_h - \langle F^o \rangle)^2}} $$

$$ = \frac{\langle w^c \rangle^T w^o}{\|w^c\| \cdot \|w^o\|}, $$

where $F^o$ and $F^c$ are either magnitudes or intensities, $\langle \cdot \rangle$ denotes the average value of the observed or calculated data, and $w^c = F^c - \langle F^c \rangle \in \mathcal{R}^m$ and $w^o = F^o - \langle F^o \rangle \in \mathcal{R}^m$.

SCALE s: This keyword determines the size of the model’s artificial unit cell. The larger the scale factor, the larger the unit cell is, and the more finely the FFT of the model protein is sampled.

After the center of mass of the model is moved to the origin, SOMoRe reorients the model via an iterative process which attempts to minimize the volume of an imaginary box containing the model protein. Next, a scale factor is applied to this box to determine the dimensions of the model’s artificial unit cell, which are

$$ s (x_{\text{max}} - x_{\text{min}}) \times s (y_{\text{max}} - y_{\text{min}}) \times s (z_{\text{max}} - z_{\text{min}}), $$

9
where $x_{\text{max}}$ and $x_{\text{min}}$ are the largest and smallest Cartesian $x$-coordinates and the other variables are similarly defined.

*Note*: We used a scale factor of 4.0 for each test problem.

**MAXGridSpacing** $d$ : The maximum allowable space between grid points that are assigned electron density values. The electron density of the model is sampled on a grid. This grid of values is the input to the FFT function.

*Note*: We used 1.0 for each test problem.

**GRAuto** : This causes SOnMoRe to automatically determine the parameter $s$, which scales an imaginary box containing the model protein, and the maximum spacing, $d$, between grid points within the imaginary box that are assigned electron density values of the model protein.

**INTERpolation_scheme** < Linear, None, Polynomial > : This defines the interpolation scheme used by SOnMoRe. Glykos notes that Polynomial will slow the code down by a factor of two [6].

*Note*: We used Linear to generate all of our results so far.

**SCMode** < Sum, Wilson > : This is an optional keyword; if the keyword is not used, then no scaling will be applied to the calculated data. Sum-based scaling also known as a linear scale factor ensures that the average value of the observed data are equal to the average value of the calculated data. This scale factor is

$$K = \sum_h |F_h^o|^k / \sum_h |F_h|^k,$$

where $k = 1$ or 2, depending on whether magnitudes or intensities are used to compute the target function.

*Note*: **Sum-based scaling should be used during local optimization** even if the correlation coefficient is used, and **Wilson scaling should not be used when predominantly low resolution data are used**, for example, during a coarse global search. For more information, see section 10.

**We always used sum-based scaling for each test problem.**

**GLOBAL b $B_{\text{overall}}$** : This keyword defines the global (isotropic) temperature factor that is assigned to every atom in the model protein, which affects the computed electron density and molecular transform calculation. The default is 20Å.

**BULK** < $k_{\text{SOL}} B_{\text{SOL}}$ > : This keyword specifies that $|F_h^c|^k$ should be multiplied by the exponential factor:

$$1 - k_{\text{SOL}} \exp \left( B_{\text{SOL}} \frac{(d^*(h))^2}{4} \right),$$

where $d^*(h)$ is the distance from the origin of reciprocal space to the lattice point $h$. The default values for the parameters are $k_{\text{SOL}} = 0.785$ and $B_{\text{SOL}} = 205.0$. 
Note: Applying a bulk solvent correction to the calculated intensities may be beneficial, especially if low-resolution structure factors are calculated. An in-depth study on the effect of the correction was not conducted, but we did use a bulk solvent correction for every test problem.

6.1.4 Some miscellaneous keywords

WRITE : This keyword instructs SOMoRe to write those structure factor magnitudes selected to be part of the free set (those used to calculate the free values) to the file Free.hkl.

SEED n : Sets the seed for the random number generator so that any random results may be reproduced, such as selection of the free data set.

PACKall : This keyword instructs SOMoRe to write out the coordinates of ALL atoms of the symmetry mates occurring in eight neighboring unit cells, rather than just the CA atoms. A PDB file contain the contents of 2x2x2 unit cells is written for the lowest valued grid point (Packing_best.moll.pdb) or the lowest valued local minima (Packing_bestopt.moll.pdb).

POSTscript < Monochrome, Color > : For your viewing pleasure, this keyword causes SOMoRe to create the postscript files 001.ps, 010.ps, 100.ps “containing the projections of electron density of the orthogonal unit cell that will be used for calculating the molecular transform” [6]. After the FFT is finished, SOMoRe will create the files mod-001.ps, mod-010.ps, mod-100.ps, “containing the modulus of the transform on the three orthogonal central sections that correspond to the electron density projections” [6].

LATTice : This keyword instructs SOMoRe to plot the locations of the reflections on mod-001.ps, mod-010.ps, mod-100.ps.

NOISE f : This is a keyword can be used for testing purposes. It instructs the code to apply a random perturbation to the magnitudes read in. The offset should range uniformly from $-f|F^0_n|$ to $f|F^0_n|$. Although we didn’t use it, we kept it (just in case).

6.2 Key words specific to SOMoRe

6.2.1 Grid search keywords

The Grid searches

Three different grid searches can be performed. Only one of these keywords can be selected at one time.
GSEARCH : This keyword instructs SOMoRe to perform a global grid search were the coarseness of the search is a function of the the high-resolution cut-off of the data set used to compute the “surrogate” target function. The lowest valued grid points found during the global search are written to the file Global TopN. For more information, see section 8.2.

Note: Currently, the domain of the global search is automatically defined for 65 space groups in the standard settings needed for protein crystallography. If the space group of the MR problem you are solving is not one of these 65, then we suggest that you use the keywords SYMM and TLIM to define your own symmetry operators and asymmetric unit.

FSEARCH : This keyword instructs SOMoRe to perform a local, constant Eulerian grid search about the initial position of part of the model molecule, while the remainder of the molecule remains fixed or immobile during the search. (A constant Eulerian grid search is described in more detail in section 8.3 on fixed model searches.) The fixed part of the molecule is specified by keyword FIXED, while the part of the molecule that is allowed to move is specified by the keyword MODEL. First, SOMoRe randomly reorients the model (to minimize an imaginary box that fits about the model to minimize the size of the FFT). During this reorientation, SOMoRe keeps track of the rotation and translation, that is, the point \( p = (\theta_1, \theta_2, \theta_3, x, y, z) \), necessary to move the MODEL back to its original position. Then, a local six-dimensional search is performed about this point \( p \). The lowest valued grid points are written to the file Fixed Local TopN.

The default domain for the local search is \( \pm 30^\circ \) about the angles and \( \pm 10 \) A about \( x, y, \) and \( z \). These defaults can be changed using the keyword LRANGE.

LSEARCH < filename > : This command instructs SOMoRe to perform a local, constant Eulerian grid search about points read in from a file. Each line of this file should be of the format:

\[
\theta_1 \quad \theta_2 \quad \theta_3 \quad x \quad y \quad z \quad f_v
\]
or

\[
\theta_1 \quad \theta_2 \quad \theta_3 \quad x \quad y \quad z \quad f_v \quad \text{free}
\]

where \( f_v \) is the target function value and \( \text{free} \) is the free value. The default range is \( \pm 30^\circ \) about the angles and \( \pm 10 \) A about \( x, y, \) and \( z \). These defaults can be changed using the keyword LRANGE. The results of the local searches are written to files Local TopN#, where # corresponds to the point’s position in the file.

Note: We did not need to use this search option to solve any of the test problems, but we did use it to investigate regions about known solutions to the MR test problems.

Keywords for these grid searches
TOPN n: This keyword specifies the number of the lowest valued grid points that SOMoRe should keep track of during the either a global search (GSEARCH) or local search (LSEARCH or FSEARCH). The default is $n = 1,000$ for GSEARCH or FSEARCH, and $n = 50$ for LSEARCH.

TLIM $x_{\text{low}}$ $x_{\text{high}}$ $y_{\text{low}}$ $y_{\text{high}}$ $z_{\text{low}}$ $z_{\text{high}}$: This keyword may be used to define the global search domain in fractional coordinates that will be searched. A default domain is automatically defined by SOMoRe for the 65 space groups needed for protein crystallography using the appropriate asymmetric unit and in some cases Cheshire symmetry.

ALIM $\theta_{1\text{low}}$ $\theta_{1\text{high}}$ $\theta_{2\text{low}}$ $\theta_{2\text{high}}$ $\theta_{3\text{low}}$ $\theta_{3\text{high}}$: This keyword may be used to define the global search domain of the Eulerian angle space in degrees. The default search domains for the Eulerian angles are $0 \leq \theta_1 \leq 360^\circ$, $0 \leq \theta_2 \leq 180^\circ$, and $0 \leq \theta_3 \leq 360^\circ$.

ORIENT $\theta_1$ $\theta_2$ $\theta_3$ $x$ $y$ $z$: This keyword is used to fix the orientation and position of one copy of the model (specified by MODEL) and its symmetry mates. This option will allow the user to search for the positions of multiple copies of the protein model in asymmetric unit when there is more than one in the asymmetric unit.

Note: This keyword may be used more than once, but so far we have only used it to solve a 12D test problem. Also, remember when you use this keyword, MOLEcules must be equal to the number of times ORIENT is called plus one.

FIXED < string >: specifies the name of the PDB file that contains the atomic coordinates that are to remain fixed during the fixed local search (FSEARCH).

LRANGE $w_{\theta_1}$ $w_{\theta_2}$ $w_{\theta_3}$ $w_x$ $w_y$ $w_z$: This keyword defines the size of the domain of a local search about each given search point. SOMoRe expects six floating point numbers to define the domains about the angles and translations that are either read in during a LSEARCH, or the angles and translation that are automatically determined for an FSEARCH. The range of $\theta_1$ is $[\theta_1 - w_{\theta_1}, \theta_1 + w_{\theta_1}]$ and similarly for the other variables. The default range is $\pm 30^\circ$ about the angles and $\pm 10\text{Å}$ about $x$, $y$, and $z$.

If the search is in fact four or five-dimensional because of the space group, then just type in dummy values for the appropriate range(s). The upper and lower bounds for these variables will be adjusted automatically before the search. For example, for P6 or spacegroup 168, $w_z$ can be any floating point number within reason because $w_z$ will be reset to zero before the local search.

However, if the search is a fixed local search (FSEARCH), then the dimension of the MR variable space will be 6D because the origin of the unit cell has been “fixed” by the fixed part of the model.
6.2.2 Local optimization keywords

**OPTIMIZE < filename >**: This keyword instructs SOMoRe to perform multi-start local optimization. The points within the file are used as starting points for the optimization algorithm (BFGS). The filename can be GlobalTopN or any other filename containing the lowest valued grid points of a global or local search.

The keyword ORIENT can be used to fix other copies of the model during optimization. Thus, the optimization will be 6D. The structure factors will be calculated using the position(s) of the model specified by ORIENT keyword(s), plus the contribution for the copy of the model being moved according to the 6D optimization.

**MULTidimensional**: The keyword instructs SOMoRe to perform 6n-dimensional optimization by optimizing multiple orientations and positions of more than one copy of the model protein. The 6n-dimensional starting point is given by the 6D point specified by the ORIENT keyword and the 6D points read in from the file specified by OPTIMIZE. For 6n-dimensional, the keyword ORIENT must be used n − 1 times.

As mentioned above, if MULTI is not used and ORIENT is used, then the optimization will be 6D.

*Note:* From the results presented in [10], 6D optimization may be just as good, if not better, than 6nD optimization. For 6nD optimization, the positions of the models specified by ORIENT are also optimized, but presumably the orientation indicated by ORIENT was obtained by 6D optimization. From the results presented in [10], often this orientation (specified by ORIENT) does not improve much, while the run time for multi-start 12D optimization was a bit longer.

6.2.3 Packing check keyword

**CHECK < CA or ALL, d >**: This keyword instructs SOMoRe to perform packing checks for the symmetry mates in the unit cell, whose positions are specified either by the lowest valued grid points (from GSEARCH) or the local minima found during multi-start optimization (from OPTIMIZE). The results are written either to Packing_Global, Packing_Fixed or Packing_Optimize.

The packing check can be between CA atoms or ALL atoms of the symmetry mates. The floating point number d is the distance cut-off. The output of the packing check is the maximum number of distance violations over all distance violations computed between each possible symmetry mate pairing. A distance violation is an inter-atomic distance that is less than the cut-off. For example, if d = 2Å, then every inter-atomic distance less than 2Å is counted for each symmetry mate pairing, and then the maximum number or worst case of inter-penetration is reported. The packing check is between symmetry mates in the unit cell and does NOT check mates related by Bravais lattice translations.
We suggest using only CA atoms, as the run time for the packing check is a function of the number of atoms. For more information, see [10].

*Note:* We suggest a CA packing check with a distance cut-off of 2Å. Also, during the packing check, the closest symmetry mates are compared, which may not be those necessarily produced directly by the symmetry operators. The pseudo code for determining the closest symmetry mate is given in Appendix A.

### 6.3 Keyword rules

The rules are pretty simple. You must use CELL, DATA, and MODEL to define the basics. (If GROUp is not used, then its assumed to be P1.) Next, you must also chose only one of the following: GSEARCH, FSEARCH, OPTIMIZE or LSEARCH. Other than that you can use any combination of the keywords, with the following exceptions below.

<table>
<thead>
<tr>
<th>If you used</th>
<th>You must use</th>
<th>You can’t use</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSEARCH</td>
<td>FIXED</td>
<td>ORIENT</td>
</tr>
<tr>
<td>MULTI</td>
<td>ORIENT</td>
<td>MOLECULES</td>
</tr>
</tbody>
</table>

### 7 Example Input File

The enclosed text in Figure 1 is an example of an input file containing keywords that define parameters used during the global search and local optimization.

### 8 More on Grid Searches

#### 8.1 Estimating the global search run time

The run time of a global search can be estimated fairly accurately by performing a very coarse grid search and noting the number of grid points \( f_1 \), number of observed data in the resolution range \( d_1 \), and the run time, which should be relatively short. For example, set the resolution range to be 500 to 25Å. Next, change the resolution cut-offs to be more reasonable, for example, use 500 to 8Å, and again note the number of grid points \( f_2 \) and observed data in the resolution range \( d_2 \), and then kill the run. The run time for the new search will increase linearly by

\[
\text{factor} = \frac{f_2}{f_1} \cdot \frac{d_2}{d_1}
\]

#### 8.2 Global grid searches

The global grid search evaluates the target function at a set of points, which we refer to as a grid points, that sample the MR variable space.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CELL</td>
<td>59.062 68.451 30.517 90.00 90.00</td>
</tr>
<tr>
<td>MOLECULES</td>
<td>1</td>
</tr>
<tr>
<td>GROUP</td>
<td>19</td>
</tr>
<tr>
<td>MODEL</td>
<td>2i1l.pdb</td>
</tr>
<tr>
<td>DATA</td>
<td>r1akisf.hkl</td>
</tr>
<tr>
<td>RESOLUTION</td>
<td>500.0 8.0</td>
</tr>
<tr>
<td>GSEARCH</td>
<td></td>
</tr>
<tr>
<td>TOPN</td>
<td>1000</td>
</tr>
<tr>
<td>CHECK</td>
<td>CA 2.0</td>
</tr>
<tr>
<td>TARGET</td>
<td>CORR-1</td>
</tr>
<tr>
<td>BULK</td>
<td></td>
</tr>
<tr>
<td>GLOBAL_B</td>
<td>20.0</td>
</tr>
<tr>
<td>SCMODE</td>
<td>sum</td>
</tr>
<tr>
<td>INTERPOLATION</td>
<td>linear</td>
</tr>
<tr>
<td>SCALECELL</td>
<td>4.0</td>
</tr>
<tr>
<td>MAXGRIDSPACING</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Figure 1: An example input file.
The grid points are just $p_i = (\Theta_i, t_i)$, where $\Theta_i = (\theta_i^1, \theta_i^2, \theta_i^3)$ are Eulerian angles that are used to rotate the reciprocal lattice, and $t_i = (x_i, y_i, z_i)$ are translations (in fractional coordinates) of the model. The sampling of orientation space is in terms of Lattman angles [13]. During the search, the Lattman angles are converted to Eulerian angles (to rotate the reciprocal lattice) because symmetry relationships among Eulerian angles are not difficult to identify.

Lattman angles $(\theta^+, \theta_2, \theta^-)$ and the optimal Lattman sampling are used because they sample angular space more uniformly than Eulerian angles and because the number of $\Theta_i$ decreases by a factor of $2/\pi$ in comparison to a constant Eulerian sampling [13]. Lattman defines the optimal sampling or step lengths in $\theta^+$ and $\theta^-$ to be

$$\Delta \theta^+ (\theta_2) = \Delta \theta_2 / \cos (\theta_2/2),$$
$$\Delta \theta^- (\theta_2) = \Delta \theta_2 / \sin (\theta_2/2),$$

where $\Delta \theta_2$ is a function of the high-resolution cut-off of surrogate function's data set:

$$\Delta \theta_2 = 2 \arcsin (r_{\text{high}} / (2(a + b + c)/3)),$$  \hspace{1cm} (7)

where $r_{\text{high}}$ is the high resolution cut-off. The definition for the step size of $\Delta \theta_2$ or (7) is the same one used by CNS Version 1.0 [3].

The sampling of translation space is defined in terms of fractional coordinates. The step sizes for translational variables are also functions of the high-resolution cut-off:

$$\Delta x = r_{\text{high}} / (3a), \quad \Delta y = r_{\text{high}} / (3b), \quad \text{and} \quad \Delta z = r_{\text{high}} / (3c).$$ \hspace{1cm} (8)

The larger $r_{\text{high}}$ is, the larger the step size is (and the lower the frequency of the surrogate function; for more information, see [10, 11]). These are the same step sizes implemented and justified by Brünger in X-PLOR Version 3.1 [2].

Currently, we have coded translation ranges for the 65 space groups of most interest to X-ray crystallographers. They are either the asymmetric unit or use the Cheshire group unit cell [9]. If the space group you wish to use is not one of the 65 we have currently coded for, then you should use the keyword TLIM to define the asymmetric unit for this space group. SOMoRe will still be able to find a solution even if you do not use TLIM, but the run time will not be the shortest possible one. (Please email us if you find a problem with any of the translation ranges.)

8.3 Fixed model searches

A fixed model search allows the user to hold part of the "model" fixed and allow the remaining part to move locally about its initial position (which should have been found by a global search followed by multi-start optimization). First, the PDB file for the fixed part is read in, the FFT calculated, and the structure factors computed and stored. Next, the PDB file for the variable part of the model is read in, another FFT is calculated, and the set of structure factors are also stored so that interpolation may be used to determine the structure factors for any rotation and translation of the variable
part of the model. Thus, two FFTs are performed for a fixed model search. In contrast, when the keyword ORIENT is used, only one FFT is performed, only that of the model.

We found this type of search very useful in determining the structure of an adenylate kinase where the flexible lid of the protein crystallized in a different orientation than the lid of the model protein, but such that the orientation was in the neighborhood of the orientation of the model's lid.

Finally, a constant Eulerian sampling of the orientation space is used, that is, $\Delta \theta_1 = \Delta \theta_3 = \Delta \theta_2 = c$, where $\Delta \theta_2$ is determined by the high resolution cut-off defined by (7). A constant Eulerian search is used for ease of programming.

### 8.4 Sequential searches

As mentioned, the ORIENT keyword allows the user to perform a sequential search when there is more than one molecule in the asymmetric unit that needs to be positioned. Based on the 12D results presented in [10], we suggest that the orientation and position specified by ORIENT be one of the lowest valued local minimizers found after optimizing the lowest valued grid points of the initial 6D global search. For more information, see [10, Chp. 7].

### 8.5 Local searches

Finally, if the user has an idea of the location of a global minimum based on a global search, then a local 6D search can also be performed. However, when searching a local neighborhood about a 6D point indicated by a global search, the same model used for the global search must be used for the local search. The lowest valued 6D points, of course, correspond to the specific model used. For example, if the polyalanine model is used during the global search, then the same polyalanine model must be used during the local search.

As a side note, if the coarse global search does not find any points close to a global minimum, it may be possible to find a solution using a rotation search. During an unsuccessful global search, its likely that the translations are more accurate than the orientations [10, Chp. 7]. As a result, local rotation searches or searches for optimal orientation may be performed while fixing the translations of the symmetry mates. To fix the translation, just use LRANGE to set $w_{tx} = w_{ty} = w_{tz} = 0$, and then set appropriate ranges for the angles. To perform a search of the entire orientation space, set $w_{\theta_1} = 2\pi, w_{\theta_2} = \pi, w_{\theta_3} = 2\pi$, the asymmetric unit of Eulerian angle space.

It should be noted that if there are say 1,000 grid points with the lowest function values, then the number of unique translations among these 1,000 points is likely to be smaller than 1,000. To avoid unnecessary computation select a set of points with unique translations. This selection can be accomplished with a simple Matlab or C code.
9 More on Multi-start Local Optimization

9.1 The local optimization method BFGS

In general, a local optimization method is designed to generate a sequence of points that will converge to a local minimizer. The algorithm is stopped or the sequence terminated once a convergence criterion or criteria are met. Often a criterion is that the norm of the gradient is small because theoretically at a local minimum the norm of the gradient is zero. To see the specifics of implementation of BFGS in SOMoRe, including BFGS algorithm, the line searches, the parameter choices, and stopping criteria, see [10].

Why did we choose BFGS despite the fact that the local optimization method often chosen in MR applications is a conjugate gradient method? First, we chose BFGS because its convergence rate is superlinear, which is faster than the linear convergence rate of a conjugate gradient method. Put another way, the search directions for BFGS are often more accurate than the conjugate gradient method, thus allowing for fewer iterations in order to converge to a local minimum (hence the faster convergence rate). (We are unable to use a Newton method, which has an even faster convergence rate, because an exact representation of the Hessian matrix is not available due to the interpolations required to calculate the structure factors of the rotated and translated model.) For more information about BFGS, see [15], for example.

In addition, often conjugate gradient methods are chosen when the variable space is quite large because the amount of storage required is smaller and the computations per iteration are fewer in comparison to BFGS; see [14], for example. However, in this application typically the variable space is quite small, well under thousands of variables. Thus, the space savings of a conjugate gradient method is not required. It is also well known that conjugate gradient methods may stall before finding a solution; see [14], for example. Though we did not experiment to determine whether stalling might take place, we decided to avoid this possibly by implementing BFGS.

In summary, we wish to emphasize that all methods mentioned so far are local; that is, they require the starting point to be within a neighborhood of a local minimum in order to converge to it. Hence, whether any local method will be successful will be determined by a starting point’s closeness to a local minimum. In the case of MR, we are looking for the global minimum. Thus, we need to find many starting points that may be close to local minima, one of which we hope is a global one. Thus, the object of SOMoRe’s global search is to identify starting points with lower function values than the remainder of the grid points sampled in hopes that one or several of these points are within the region of convergence of a global minimum.

9.2 Convergence results

After local optimization SOMoRe prints some statistics on the optimization just performed. The rows entitled Convergence, Success, and Stagnation provide statistics for those final iterates that meet the first convergence criterion, the second convergence criterion and the third stopping criterion, respectively; see [10].
The first column is the average number of iterations the algorithm performed, that is, the number of points in the sequence of points generated by the algorithm. The second column is the average value of the norms of the final gradients. When the norm of the gradient is small, this is a good indication that a local minimum has been found. The third column is average percentage of full BFGS steps taken, that is, those steps such that the step length parameter $\alpha = 1$. The fourth column reports the average number of times the Hessian update was skipped. This provides an indication of how the algorithm is doing. If the Hessian update is skipped many times, then BFGS method may be stalling, that is, the iterates may not be moving very far. The fifth column reports the average number of times that the function value does not change, thereby providing an indication of whether or not the algorithm is converging or stalling. The last column is the norm of the gradient for the starting point. (This is not particularly useful, but interesting.)

10 Caveat About Scaling

Finally, we suggest that sum-based scaling of the calculated intensities should always be used rather than either Wilson scaling or no scaling at all, especially during local optimization. For example, Wilson scaling prevented test problem 2MBW, which has a perfect model protein, from being solved; see [10].

We have observed that using the linear scale factor or sum-based scaling has a positive effect on the local optimization method BFGS, even though the correlation coefficient is theoretically scale invariant to such a linear scale factor. This discrepancy in theory may be due to the current choice for the perturbation parameter $h_\epsilon$ used to compute the finite difference gradient. Several parameter values were tested, while the linear scaling factor was in effect, and we chose the best one based on number of iterations until convergence was achieved and norm of the final gradient (the smaller, the better). It is also possible that sum-based scaling introduces, less round off error in the finite precision computer arithmetic involved in computing the correlation coefficient.

The effect of linear scaling and no scaling can be observed by comparing the average value of the final gradient, which is printed out at the end of the optimization under the column labeled “avg. $||\text{grd}||$”. The smaller the norm of the final gradient, the better. The norm of the gradient should be close to zero at a local minimum.

In addition, when the linear scale factor is used, the number of iterations needed to satisfy a convergence criterion is often fewer, due to the fact that the gradient is generally larger when linear scaling is not used.

But most importantly, when this scaling is used, better local minima were found for the test problems. In other words, local minima that led to lower RMSDs between the known target protein and the reoriented model. When no scaling is used, the RMSD associated with the final iterate or local minima is typically larger, and the final norm of the gradient is typically quite large.
11 Additional Codes

11.1 Creating additional PDB files: writepdb

The executable writepdb generates PDB files from a file of containing either local minima or grid points. These PDB files are named Writepdb_.pdb, where # will either be the name of the points (when they are local minima), for example, 1b, 2b and so on or the order in which the points were read in, for example, 1, 2 and so on (when they are the results of a grid search). The first few lines of the PDB files are REMARK lines that specify the 6D point (three angles and three translations) used to generate the coordinates.

The syntax for calling writepdb is

    writepdb  inputfile.in  file-of-points.

A SOMoRe input file is also required to specify the model, the dimensions of the unit cell, and the space group.

The first few lines of output of writepdb are similar to those of SOMoRe because writepdb recalculates the coordinates of Somore.pdb (which are written to the file WriteSomore.pdb just in case the user wishes to compare these two files). Recall, that the 6D points output by SOMoRe apply to Somore.pdb and not directly to the PDB file of the model. Also, while reading in the points from the file, writepdb creates the file WritePoints that reprints these points.

Before the PDB files are written, the user will be asked three questions. First, are the points preceded by a name, for example, 1b (Y/N)? This will be true if the points are local minima and not true if they are the results of a grid search. Second, do the angles apply to reciprocal space (Y) or real space (N)? Currently, all angles output by SOMoRe apply to reciprocal space. Third, would you also like a packing PDB file (2x2x2 unit cells worth) (Y/N)? We discuss this option after the following section.

PDB files for higher-dimensional (greater than 6D) minima

For higher dimensional optimization results, writepdb will create a PDB file for each 6D component of the minimum (or starting point), that is, a PDB file for each copy of the model in the asymmetric unit. For example, SOMoRe prints a higher dimensional minima in the following format

    1b  θ₁  θ₂  θ₃  x  y  z
    1b  θ₁  θ₂  θ₃  x̃  ỹ  z̃.

The 6D component of the minimum on each line determines the position of one copy of the model. Now, because each line begins with the point’s name, the input file must contain the keyword MULTIdimensional so that writepdb expects such points, and the PDB files will be properly named. In the case above, the PDB files will be named Writepdb_1b1.pdb and Writepdb_1b2.pdb. (If two points are read in with the same name and MULTIdimensional has not been used, then the program will terminate so that PDB files are not given the same name and overwritten.)
Packing PDB files

In addition to writing a PDB file for each point, another file containing the coordi-
nates of eight adjacent unit cells can also be written. These files are named Writepdb_packing#.pdb. 
The user might use this to see if there is any inter-penetration of symmetry mates. How-
ever, there are programs that will easily generate such pictures using a single PDB file 
and the symmetry operators.

11.2 Generating packing results: pack

If SOMoRe was not instructed to perform a packing check, a packing check can still 
be performed after the fact by using pack. The executable pack will perform the same 
check and write the results to the file Packing. For a review of SOMoRe's packing 
check, see 6.2.3.

The syntax for calling pack is

pack file-of-points.

(The executable pack expects the angles of the points to apply to reciprocal space, 
which is the current output format of SOMoRe.)

Before the packing checks are performed, the user will be asked five questions. First, 
do you want to compute packing checks using CAs only (Y/N)? Packing checks can be 
performed using all atoms in the model, but this takes longer. Second, what should the 
distance cut-off (in Angstroms) be? The distance cut-off is used to determine how many 
atoms of the neighboring symmetry mates are within this distance. If this distance is, 
say, 2 Å and many atoms of neighboring symmetry mates are within this distance, then 
the mates are likely to be inter-penetrated. Third, are the points preceded by a name, 
for example, 1b (Y/N)? This will be true if the points are local minima and not true 
if they are the results of a grid search. Fourth, is this a 1D problem (Y/N)? If it is, 
then another file is written, called Packing1D, to summarize the results of Packing. 
Fifth, enter the name of the PDB file whose coordinates will be rotated and translated 
according to the points in the file. This should be the file Somore.pdb (whose CRYS-
TLINE will specify the unit cell dimensions and space group). Recall, that the 6D points 
output by SOMoRe apply to Somore.pdb and not directly to the PDB file of the model.

12 Future Work

Because SOMoRe's global search is deterministic, consisting of many independent func-
tion evaluations, the global search can be easily parallelized on multiple processors. A 
modification of this type in currently underway at the University of Wisconsin-Madison.
A Determining the closest symmetry mate

A pseudo code for determining the closest symmetry mate is below. Let the center of mass of a molecule be $c = \left( \sum_{j=1}^{N} x_j/N, \sum_{j=1}^{N} y_j/N, \sum_{j=1}^{N} z_j/N \right)$, where $N$ is the number of atoms, and $(x_j, y_j, z_j)$ are the orthogonal coordinates of the $j$th atom.

**Pseudo code for determining the closest mate**

**Step 1.** Compute the centers of mass of symmetry mates $i$ and $j$; call these $c_i$ and $c_j$.

**Step 2.** Compute $d_c = c_i - c_j$.

**Step 3.** Convert $d_c$ to fractional coordinates, that is, compute $d_f = A^{-1}d_c$, where $A$ is the real space orthogonalization matrix.

**Step 4.** Add the translation $b_t$ (which is in terms of fractional coordinates) to the fractional coordinates of symmetry mate $j$.

A pseudo code for determining the appropriate basis vector translation, $b_t$, is below. Let $d_f(k)$ and $b_t(k)$ refer to the $k$th component of $d_f$ and $b_t$, and let floor(), ceil(), and mod() be the standard floor, ceiling, and modulo functions.

**Pseudo code for determining $b_t$**

for $k = 1$ to 3
  if $d_f(k) > 0$ (mate $i$ is to the right of mate $j$ with respect to basis vector $k$)
    $b_t(k) = \text{floor}(d_f(k))$
    if mod($d_f(k), 1.0) > .5$
      $b_t(k) = b_t(k) + 1$
  else
    $b_t(k) = \text{ceil}(d_f(k))$
    if mod($d_f(k), 1.0) < -.5$
      $b_t(k) = b_t(k) - 1$
end

end
somore 1aki.in >& 1aki-500-8F
mv GlobalTopN TopN-1aki-500-8F

somore 1aki-opt.in >& 1aki-500-8F-500-4F
mv Optima Optima-1aki-500-8F-500-4F

Figure 2: A sample script file.

B Unix Script Files

It is often worthwhile to print the output of SOMoRe to a file as it runs. This can
be easily accomplished, for example, in a UNIX operating system by a script file. In
addition, the script file can also call SOMoRe to perform multi-start local optimization
immediately following the global search. Figure 2 is an example of a script file.

If the above file, is called 'runsomore', then simply type 'runsomore &' to execute
the script file and run the job in the background. However, the permissions for the
script file will likely need to be changed beforehand using the command 'chmod 700
runsomore'.
References


