EPMR: Molecular Replacement by Evolutionary Search

User's Guide for EPMR Version 16.07.1

Introduction

EPMR is a general-purpose crystallographic molecular replacement program. Unlike traditional molecular replacement programs, it does not divide the problem into separate rotation and translation searches. Instead, it uses an evolutionary search algorithm to simultaneously optimize the orientation and position of a search model (1, 2). The program operates as follows:

- An initial set of random solutions (random orientations and positions for the search model) is generated. * The correlation coefficient between Fo and Fc is calculated for each trial solution.
- A fraction of the highest scoring solutions are retained and used to regenerate a complete set of new trial solutions. This is done by applying random alterations to the orientation angles and translations for each “surviving” solution. * The correlation coefficients for the new population are calculated, the population is again regenerated from the top scoring solutions, and this procedure is repeated for a specified number of cycles.

In this way, the algorithm provides broad, stochastic, initial sampling of the search space while gradually focusing in on the most promising regions. It allows for efficient searching of the six-dimensional (or higher) space. In general, it is several orders of magnitude faster than a brute-force, systematic, 6-D search. At the end of the evolutionary optimization, a local minimization is performed on the best solution. This is simply a rigid-body refinement of the search molecule.

The program calculates structure factors rapidly by indexing into a molecular transform using the method of Huber and Schneider (3). A traditional structure factor calculation is done only once, for the search model set at the origin of a P1 cell. Subsequent structure factor calculations are done by transforming reflection indices according to the rotations and translations applied to the model and the relationship between the P1 and real cells, interpolating into the grid of P1 structure factors and summing over the symmetry operators of the crystal. This is

much faster than an FFT calculation. A simple Babinet-type solvent correction is applied to the calculated structure factors. The values of the solvent correction parameters (k, B) are optimized during the search.

Because of the stochastic nature of the evolutionary optimization process, the correct solution will not be obtained on every run, even with a very good search model. The success rate is dependent on the quality of the model (2). By default, 20 optimization attempts are done, and more may be required if you have a difficult problem. For search models that are poor and at the limit of detection, the search efficiency can be quite low. If you have a molecular replacement problem that has not yielded a solution by any other means, a reasonable last resort is to set up EPMR to do as many runs as your patience and computing resources will allow. As long as the true solution represents the global maximum in the correlation coefficient between Fo and Fc, even if by the slimmest of margins, the algorithm should eventually find it.

EPMR includes the following features: * the ability to automatically search for multiple copies of a molecule in the asymmetric unit, either sequentially or concurrently * the ability to search with multiple models, either sequentially or simultaneously (i.e., in competition with each other) * the ability to use multiple coordinate sets as parts of an “assembly” that comprises the complete search model * an option to search over all related space groups, either sequentially or simultaneously * rotation-only and translation-only search modes * an option to provide static, partial structure * independent optimization of each segment of a search model during the final rigid-body refinement step * an option to bypass the evolutionary search and do only local, rigid-body optimization of the model

Feedback is welcome. E-mail concerning the program can be sent to ckissinger@epmr.org.

If you publish results obtained using EPMR, please cite:

Charles R. Kissinger, Daniel K. Gehlhaar & David B. Fogel, “Rapid automated molecular replacement by evolutionary search”, *Acta Crystallographica*, **D55**, 484-491 (1999).

Usage

EPMR is run from the command line. The program requires either two or three input files depending on the types of files used. The first file specified on the command

line generally will be either a Scalepack (*.sca*) file or a CCP4 MTZ file. Alternatively, a text file specifying only the cell constants and space group number can be used. In this case the file should contain that information in the following order:

```
a b c alpha beta gamma space_group_number
```

The exact formatting of this file is not critical. An example of appropriate contents of the text file for a cell in space group C2 is:

```
40.76 18.49 22.33 90 90.61 90 4
```

All 230 space groups are available for use in the program, in their standard settings. (For rhombohedral space groups, the hexagonal setting is expected, and your data need to be indexed accordingly.)

The second file on the command line should be a standard PDB format file containing the search model. Any lines in this input file that are not ATOM or HETATM records are ignored. The program expects correctly formatted PDB ATOM records. If element symbols are not present in columns 77-78, the atom type can usually be inferred correctly from the atom name, but the names of atoms with two-letter element symbols (*e.g.*, Fe, Ca, Se) should be appropriately left-shifted to ensure they are correctly recognized. If the PDB file contains CNS-style segment identifiers in columns 73 to 76, these will be used to subdivide the search model during the final rigid-body refinement. Each of these segments will be optimized independently during that step.

If the first file on the command line is a Scalepack or an MTZ file, and you also want to use it as the source of your reflection data, it is not necessary to provide a third input file on the command line. For Scalepack files, the program expects a *.sca* file with the final, merged reflection intensities (with Bijvoet pairs either combined or preserved as separate I+ and I-). For MTZ files, the data column to use for the Fobs or intensity data can be specified on the command line using the *-Z* option. Otherwise, the program will read the data from any column labeled “F”, “FP”, “FNAT”, "I" or "IMEAN" or from the first column with a label beginning with “F_” or “FP_” or "I_". If no such column is present, the first data column in the MTZ file with an appropriate data type will be used.

If you want to provide the reflection data separately from the unit cell and space group information, a third input file can be specified, which must be a Scalepack file, MTZ or a simple text file containing the observed structure factors. The only requirement for the text file is that H, K, L, and Fobs are the first four items on each line, separated by spaces or tabs.

The command line:

```
epmr example.sca example.pdb
```

or:

```
epmr example.mtz example.pdb
```

or:

```
epmr example.cell example.pdb example.hkl
```

will run the program in its default mode. In this case, the program will look for a single copy of the search molecule in the asymmetric unit. It will run the evolutionary search procedure twenty times. Data in the resolution range between 80 and 3 Ångstroms will be used in the search. Each of the 20 solutions found will be written to a file named *epmr.**N**.pdb*, where **N** is the sequential attempt number.

If you wish to provide multiple search models, put the list of file names in a text file, and supply that file to the program in place of a coordinate file, with a "@" symbol before it:

```
epmr example.mtz @example.filelist
```

If you provide a file list in this way, it is important to keep in mind that the complete path to the files from your working directory is required. It is recommended to specify the full, absolute path to each file so your file list is not dependent on the directory in which it is used.

A single search model can be read from the standard input by using a "-" (dash) in place of the coordinate file name on the command line:

```
epmr example.cell - example.hkl <example.pdb >example.log
```

When the program is started, it will print some information about the input data and program settings, do the initial FFT structure factor calculation, and then start the optimization runs. At the end of each evolutionary search, a local, rigid-body refinement is performed on the result.

The final orientation for each run will be reported on a line that begins with the word "Solution" followed by the run number, correlation coefficient, R-factor, rotation angles (alpha, beta, gamma as defined for the CCP4 programs), translation (in fractional coordinates for the model after it has been centered at the origin), the model number and space group. In most cases, a solution will be transformed through symmetry and/or alternate origins so that it lies as close as possible to the origin. (More limited transformation is done when static structure

has been applied or if only a local search is done.) Thus if the same solution is found by more than one optimization attempt, nearly identical rotation angles and translations will be reported, with no transformations necessary to compare solutions. If you intend to make use of the rotation and translation values outside of the program, they must be applied to the original search model after it has been moved so that its centroid is at the origin.

You can use the Linux/UNIX command:

```
grep Solution epmr.log
```

to view just the solutions. On UNIX-like systems, the command:

```
grep Solution epmr.log | sort -n -k 3r,3 -k 4
```

will list the solutions sorted by correlation coefficient, then R-factor.

The output coordinates are written in standard PDB format. The TITLE record includes the correlation coefficient, R-factor and number of clashes. A CRYST1 record is included. The original atom information from the input coordinates, such as chain identifiers and residue numbers, is retained in the output coordinates except when a search is done using multiple search models in an "assembly" (-a or -A options) or when a search is done for multiple copies of the search model (-m or -M options). In those cases, new chain identifiers are assigned, one for each "piece" of the assembly or copy of the search model, to make it easier to identify the separate components. Only ATOM or HETATM records from the input coordinates are stored and written to the output files. All other lines are ignored.

Options

The operation of the program is controlled by a set of command line options. (If the program is run without any command line arguments, a brief summary of the available options will be listed.) The possible options are:

-A Treat search models as an **assembly**, search with all simultaneously

This option is off by default. It causes each input model to be treated as an independently positioned and oriented segment of the complete search model. (By default, each input model is treated as an entirely separate, alternative search model.) When this option is turned on, the program will attempt to optimize the orientation and position of all search models simultaneously.

This is an experimental option. It is primarily useful when the individual segments that comprise the assembly would be too small to be effective when trying to build the solution incrementally using the -a option.

-a Treat models as an **assembly**, search with each sequentially

This option is off by default. It causes each input model to be treated as an independently positioned and oriented segment of the complete search model. (By default, each input model is treated as an entirely separate, alternative search model.) When this option is turned on, the program will find the optimal solution for the first model, keep that as static structure, search for the optimal solution for the next model, store that as static structure, and so on.

Both this option and the "-A" option can be combined with "-m" or "-M" to search for multiple copies of an assembly.

-C Search simultaneously over all space groups in the same point group and **crystal** class as the input space group

This option is off by default. The space group is treated as an additional variable in the evolutionary search (i.e., the alternative space groups compete with each other during the evolution). This option works well for routine molecular replacement calculations with a good search model. For difficult cases, it is safer to use the "–c" (lowercase) option instead, which will search each of the candidate space groups separately.

-c Search sequentially over all space groups in the same point group and **crystal** class as the input space group

This option is off by default. Each space group is tried in turn. The number of optimization runs specified by the "–n" option will be completed for each space group choice, and searches will be done for all copies of all input models in a space group before moving to the next.

-e integer Set the **seed** value for the random number generator to a specific value

By default (or if the seed value is set to a value of zero using this option), the seed is generated from random system information at run time. Explicitly setting the seed value is not necessary for normal operation of the program, but can be useful for testing purposes and allows you to reproduce a previous run. Two separate runs of the program that use the same seed value will produce identical results.

-g integer The number of **"generations"** (cycles of optimization)

The default value is 50. It is not normally necessary (or recommended) to change this value. However, setting this value to zero (-g0) will bypass the evolutionary search and feed your input model directly to the local optimizer. This allows you to use EPMR as a convenient rigid-body refinement program.

-h number High resolution limit for diffraction data used in the search (in angstroms)

The default value is 3.0 Å. Although it can be useful to try different high-resolution limits, the search efficiency is generally higher if data to 3.0 are included. It is rarely effective to set this to a value greater than 5.0.

-l number Low resolution limit for diffraction data (Ångstroms).

The default value is 80.0 Å, which generally results in no data being excluded. MR calculations can be highly sensitive to the inclusion or exclusion of the lowest resolution data, and also to the accuracy of that data. For EPMR it is highly recommended to include all low resolution data. In some cases, however, poorly measured low-resolution data can negatively impact the search. If you are having problems, you could try excluding the lowest resolution data using this option with, for instance, a low-resolution limit of 15 Å.

-M integer The number of copies of the **molecule** in the asymmetric unit to find simultaneously

The default value is 1. Values greater than one cause multiple orientations and positions to be optimized for the search model.

This option is primarily useful when there is a large number of molecules in the asymmetric unit. In other cases an incremental search for multiple molecules (option -m) is typically more effective.

-m integer The number of copies of the **molecule** in the asymmetric unit to find sequentially

The default value is 1. A value of 2 would cause the program to search for one copy of the molecule, save the solution as partial structure and continue searching for a second solution.

-n integer The **number** of independent optimization attempts

The default value is 20, which is intended for moderately difficult cases. For some very difficult MR problems, values of 100 or more are worth trying.

-o name The file name prefix for the **output** coordinate files

The default is “epmr”. If you run multiple jobs in the same directory, you will have to use this flag to avoid writing over files from other runs. If you specify “-” (a dash) as the file name prefix, the coordinates will be written to the standard output of the program instead of a separate file.

If you specify the option "-w1", the coordinate file containing the top solution from all runs will be named ***prefix****.best.pdb*. If you specify the command line option “-w2”, the program will name the output file for each optimization attempt as

prefix.N.*pdb* (*e.g.*, epmr.1.pdb). If you are searching sequentially over multiple space groups, search models, and/or copies in the asymmetric unit, the file name will have additional numeric indicators for those (*e.g.*, *epmr.3.4.2.5.pdb* would indicate the third space group choice, fourth model, second copy, fifth optimization attempt).

-p integer The **population** size (number of trial solutions evaluated in each cycle of optimization)

The default value is 300. Increasing this value beyond 300 will increase the search efficiency, but you will get more benefit from increasing the number of optimization attempts (-n) instead.

-R Rotation search only

This option is off by default. It will cause the program to search only rotation space, keeping the position of the search model unchanged.

-S Try all **search** models simultaneously

This option is off by default. The choice of the model becomes an additional variable in the evolutionary search. This is an experimental option still under development.

-s filename Read **static** structure from the specified file

If you have partial structure to input, include this flag and follow it with the name of the PDB file containing the correctly positioned partial structure. You can separate the partial structure into as many files as you wish and use this flag multiple times on the command line.

-T Translation search only

This option is off by default. It will cause the program to search only translation space, keeping the orientation of the search model unchanged. This could be useful, for instance, when you have a search model that has been pre-oriented by another program or through knowledge of non-crystallographic symmetry. Unlike in earlier versions of this program, the orientation is held fixed during final rigid-body optimization as well as during the evolutionary search.

-V Print **version** information and exit

-v integer The **verbosity**, which controls the amount of information written to the standard output of the program The default value is 1. A value of 0 (zero) results in nothing being written (except the coordinates, if they are written to the standard output using the option, “-o -”). A value of 2 causes more detailed information on the progress of the evolutionary search to be written.

-w integer The quantity of solutions you want to **write** out to PDB files

The default value is 2. A value of 0 (zero) here results in no coordinates being written out. A value of 1 means only the top solution from all of the runs will be written out. A value of 2 means all solutions will be written out. (The -o flag controls the naming of the output PDB files.)

-Z label The label of the MTZ data column to use. The data column can contain either structure factor or intensity data.

Acknowledgements

The original EPMR program was written by Chuck Kissinger and Dan Gehlhaar at Agouron Pharmaceuticals. David Fogel (Natural Selection, Inc.) contributed numerous ideas that were essential to the original development of the program. Bradley Smith (Agouron/Pfizer) was responsible for rewriting and improving the original program and testing numerous alternative algorithms and ideas. SGX Pharmaceuticals and Anadys Pharmaceuticals supported the development of the open-source version of this program.

References

1. Kissinger, CR, Gehlhaar, DK & Fogel, DB, “Rapid automated molecular replacement by evolutionary search”, *Acta Crystallographica*, **D55**, 484-491 (1999). 2. Kissinger, CR, Smith, BA, Gehlhaar, DK & Bouzida, D, “Molecular replacement by evolutionary search”, *Acta Crystallographica*, **D57**, 1474-1479 (2001). 3. Huber, R. & Schneider, M., “A group refinement procedure in protein crystallography using Fourier transforms”, *J. Appl. Cryst.* **18**, 165-169 (1985).

Examples

Example 1

Use EPMR in default mode. Read reflection intensities, cell constants and space group from a Scalepack file. Read coordinates for a single search model from a PDB file. Search for one molecule in the asymmetric unit, perform 20 attempts of the evolutionary search procedure, and write out all 20 solutions found:

```
epmr intensities.sca search_model.pdb > example.log
```

Example 2

Do a translation search, using static partial structure and perform 50 optimization attempts. Use the data labeled 'F_NAT1' in the MTZ file:

```
epmr -T -s static_model.pdb -n50 -Z F_NAT1 data.mtz search_model.pdb >
example.log
```

Example 3

Search for one molecule in the asymmetric unit (default), use data from 20 to 2.5 Å, do fifty runs, write out the solution found in each attempt to a file with the prefix *example_solution*. Read the cell constants and space group from a text file, read the reflections from a free-format file containing h, k, l, Fo and use static structure:

```
epmr -l 20 -h 2.5 -s partial.pdb -w2 -o example_solution -n 50
example.cell example.pdb example.hkl > example.log
```

Example 4

Search sequentially for three identical molecules in the asymmetric unit, write out just the top scoring solution from all attempts after 20 attempts (default) for each of three copies of the search molecule. Input two fragments of partial structure:

```
epmr -m3 -w1 -s partial1.pdb -s partial2.pdb example.cell example.pdb
example.hkl > example.log
```

Example 5

Do a rigid-body refinement of a molecule:

```
epmr -g0 example.sca example.pdb > example.log
```

Example 6

Use EPMR in a program pipeline. The search model is read from the output of the mythical program *search_model_generator* and the solution coordinates are written to the input of *solution_evaluator*. Cell and reflection data are read from an MTZ file. Do 100 optimization attempts, writing the coordinates for each solution found to the standard output. No log file information is written:

```
search_model_generator | epmr -o- -v0 -n100 example.mtz - |
solution_evaluator
```