



OpenEye
Scientific Software

OMEGA
Release 2.4.6

OpenEye Scientific Software, Inc.

February 14, 2012

CONTENTS

1	Front Matter	1
2	Installation and Platform Notes	3
2.1	Licenses	3
2.2	General Installation	3
2.3	PVM	5
2.4	Open MPI	6
2.5	Uninstallation	8
3	OMEGA Theory	11
3.1	Filtering	12
3.2	Stereochemistry Enumeration	12
3.3	Fragment Library Generation	12
4	OMEGA Application Usage	15
4.1	Command Line Interface	15
4.2	Further specifications	22
5	<i>makefraglib</i>: Creating a Fragment Library	23
5.1	<i>makefraglib</i> Theory	23
5.2	<i>makefraglib</i> Usage	23
6	<i>flipper</i>: Enumerating Stereocenters	27
6.1	<i>flipper</i> Theory	27
6.2	<i>flipper</i> Usage	27
7	<i>oeb2sdconf</i>	29
7.1	<i>oeb2sdconf</i> Usage	29
8	Release Notes	31
8.1	OMEGA 2.4.6 (<i>February 2012</i>)	31
8.2	OMEGA 2.4.3 (<i>August 2010</i>)	31
8.3	OMEGA 2.4.1 (<i>June 2010</i>)	32
8.4	OMEGA 2.4.0 (<i>November 2009</i>) (Toolkit only)	32
8.5	OMEGA 2.3.3 (<i>March 2009</i>) (Toolkit only)	33
8.6	OMEGA 2.3.2	33
8.7	OMEGA 2.3.1	33
8.8	OMEGA 2.3.0	33
8.9	OMEGA 2.2.2	34
8.10	OMEGA 2.2.1	34

Bibliography

35

Index

37

FRONT MATTER

Copyright 1997-2012 OpenEye Scientific Software, Santa Fe, New Mexico. All rights reserved.

All rights reserved. This material contains proprietary information of OpenEye Scientific Software. Use of copyright notice is precautionary only and does not imply publication or disclosure.

The information supplied in this document is believed to be true but no liability is assumed for its use or the infringement of the rights of others resulting from its use. Information in this document is subject to change without notice and does not represent a commitment on the part of OpenEye Scientific Software.

This package is sold/licensed/distributed subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out or otherwise circulated without OpenEye Scientific Software's prior consent, in any form of packaging or cover other than that in which it was produced. No part of this manual or accompanying documentation, may be reproduced, stored in a retrieval system on optical or magnetic disk, tape, CD, DVD or other medium, or transmitted in any form or by any means, electronic, mechanical, photocopying recording or otherwise for any purpose other than for the purchaser's personal use without a legal agreement or other written permission granted by OpenEye.

This product should not be used in the planning, construction, maintenance, operation or use of any nuclear facility nor the flight, navigation or communication of aircraft or ground support equipment. OpenEye Scientific Software, shall not be liable, in whole or in part, for any claims arising from such use, including death, bankruptcy or outbreak of war.

Windows is a registered trademark of Microsoft Corporation. Apple, OS X, and Macintosh are registered trademarks of Apple Computer, Inc. AIX and IBM are registered trademarks of International Business Machines Corporation. UNIX is a registered trademark of the Open Group. RedHat is a registered trademark of RedHat, Inc. Linux is a registered trademark of Linus Torvalds. SPARC is a registered trademark of SPARC International Inc.

SYBYL is a registered trademark of TRIPOS, Inc. MDL is a registered trademark and ISIS is a trademark of Accelrys, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrodinger, Inc.

Python is a trademark of the Python Software Foundation. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. and other countries.

Other products and software packages referenced in this document are trademarks and registered trademarks of their respective vendors or manufacturers.

INSTALLATION AND PLATFORM NOTES

2.1 Licenses

To run OMEGA you will need to obtain a license file for OMEGA from OpenEye Scientific Software (business@eyesopen.com). The license file should be pointed to by the environment variable `OE_LICENSE`.

If you intend on running multi-processor OMEGA via PVM or Open MPI, only the master machine needs access to the license file.

On Windows, the environment variables can be set under the system Control Panel.

2.2 General Installation

2.2.1 General Installation

By default, all OpenEye applications are installed into a single distribution directory tree on the specified machine. The default location for this tree is platform specific and will be detailed below.

The root of the tree (i.e. the `openeye` directory) contains the following subdirectories:

- admin** This directory is intended to contain any administrative scripts and tools associated with the installed applications. Currently, this directory is simply a placeholder on all platforms except for Microsoft Windows, where it contains the uninstaller executables.
- arch** This directory contains the collection of platform specific subdirectories. Each subdirectory contains the actual installed executables and support libraries for the associated platform. In the platform specific subdirectory there will be a subdirectory for each application. Within that will be another subdirectory for each version of that application.
- bin** This directory contains a startup script for each application that has been installed. This script determines, at run-time, what the current platform is and then calls the appropriate executable in the `arch`. This script enables the easy co-existence of multiple platforms and versions of any OpenEye application in the same distribution tree.
- data** This directory contains all of the associated data for the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

docs This directory contains all of the documentation associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

examples This directory contains all of the examples associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

The startup script discussed in the section on the `bin` directory above will have the same name as the installed executable with which it is associated. When the script is called, it will attempt to determine the current platform and run the appropriate executable if installed. If an appropriate executable cannot be found, the script will report that information, as well as a list of the currently installed platforms. The auto-detection can be overridden by setting one of two environment variables:

- **OE_ARCH** can be used to specify a colon separated list of compatible distributions for the current platform such as:

```
redhat-RHEL5-x64:redhat-RHEL4-x64
```

Specification of this environment variable overrides the auto-detection process, if it is present. If none of the compatible distributions listed are found, the script will fall back to the auto-detection process.

- **APPNAME_OE_ARCH** can be used to specify a colon separated list of compatible distributions for a specific application (as specified by changing the **APPNAME** text in the environment variable name) just like **OE_ARCH** as detailed above.

Specification of this environment variable overrides the **OE_ARCH** environment variable as well as the auto-detection process. If none of the compatible distributions listed are found, the script will fall back to the **OE_ARCH** list first and then to the auto-detection process.

Specifying this variable provides a simple way to customize the behavior for individual applications on non-standard platforms.

The startup script also supports a few commandline arguments including:

- | | |
|---------------------|--|
| -path | Specifying this argument will output the full path of the executable to be run. The executable will not be started if this argument is present. |
| -print_arch | Specifying this argument will output the details of the current platform as detected by the script as well as which platform-version of the executable is being run. The executable will be started if this argument is present. |
| -use_version | Specifying this argument followed by a specific version number allows the user to control which released version of the executable to run. |

2.2.2 Linux/Unix

Linux/Unix distributions are provided as a gzipped tarball of the distribution tree described above. Installation is performed by untarring the file in the desired location. Multiple distributions can be installed in the same location without any challenge.

To ensure that the installed applications can be called from the command line, be sure to add the full path of the `openeye/bin` subdirectory to the **PATH** environment variable. For instance, if the distribution was installed into `/usr/local/openeye`, the **PATH** environment variable should contain: `/usr/local/openeye/bin`.

2.2.3 Windows

Windows distributions are provided as a standard EXE installer. By default, all OpenEye applications will install into the `C:\OpenEye` directory.

An OpenEye group with an application specific subgroup will be added to the *Start* menu. The application specific subgroup will contain links to the documentation, the uninstaller, as well as to a Windows command shell which has the appropriate **PATH** settings already defined to allow the user to simply type the executable name at the prompt without concern for where the executable is actually installed.

For GUI applications, a link to the application will be created on the desktop as well as in the application specific subgroup of the *Start* menu.

2.2.4 Mac OS X

Mac OS X distributions are provided as a standard *pkg* installer delivered as a *dmg* disk image. By default, all OpenEye applications will install into the `/Applications/OpenEye` directory.

To ensure that the installed applications can be called from the command line in the *Terminal*, be sure to add `/Applications/OpenEye/bin` to the **PATH** environment variable.

For GUI applications, an application bundle which can be clicked on to start, will be present in the `/Applications/OpenEye` directory. This bundle cannot be moved independent of the `OpenEye` directory. For instance, the entire `OpenEye` directory can be moved as one piece, but moving the application bundle or the contents of any of the subdirectories in the `OpenEye` directory may cause the application to not start. However, the bundle can still be dragged into the Dock and run from there without any problem.

2.3 PVM

PVM or parallel virtual machine is a freely available library for running processes on more than one processor on one or more machines ([Geist-1994]). OMEGA can take advantage of PVM to distribute jobs over multiple processors. To do this PVM must be installed on all the machines OMEGA will be distributed over. The PVM source is freely available from http://www.csm.ornl.gov/pvm/pvm_home.html. However many Linux distributions, and some Unix versions, include PVM by default. OMEGA is built with the current PVM version 3.4.5, but should also work with PVM version 3.4.4. OMEGA does not support PVM under Windows.

To use OMEGA with PVM you must do one of the following.

- Place a link to the `omega` executable in `$PVM_ROOT/bin/$PVM_ARCH`
- Define the environment variable **PVM_PATH**, which points to the directory where the OMEGA executable resides.

The environment variables **PVM_ROOT** and **PVM_ARCH** should be defined globally as part of the PVM installation. **PVM_PATH** is generally a user-defined environment variable, and must be defined for all shells (*i.e.*, you can't just define it in the shell you're launching OMEGA in).

Note: There is no specific slave executable. The executable distributed for this program serves as both a master and slave PVM program as well as a single processor version.

2.3.1 PVM hosts file

After PVM and OMEGA are installed, each OMEGA job run via PVM must be supplied a file (via `-pvmconf`) that describes the hosts and number of CPU's to use. Each line in this file includes the word **host** followed by the hostname of the slave followed by the number of CPU's to use on that slave.

To use 10 slaves on a linux cluster (where nodes are named *linux1*, *linux2*, etc.)

```
host linux1 2
host linux2 2
host linux3 2
host linux4 2
host linux5 2
```

To use 32 CPU's on a single-image system like an SGI Altix, a single line can be used:

```
host altix 32
```

2.3.2 PVM Scaling

PVM jobs involve a certain of network traffic, sending multi-conformer molecules from the master to the slaves and sending results from the slaves back to the master. The single OMEGA slave jobs are relatively fast compared to the network I/O time, so for scaling beyond 32 CPUs, additional effort is required in setting up a job. For low numbers of CPUs or for divide-and-conquer style applications, using `.oeb.gz` as the dbase file format is fine.

2.4 Open MPI

As of version 2.4.1, OMEGA supports MPI-1, or Message Passing Interface 1, on all platforms except Microsoft Windows, Solaris & AIX. MPI, like PVM, enables distributing OMEGA runs over multiple machines and processors.

OMEGA uses the Open MPI implementation of MPI, found at <http://www.open-mpi.org>. Every version of OMEGA includes a full Open MPI install, so no additional software is needed.

There are two requirements to run OMEGA under Open MPI.

- Every machine in the cluster must have OMEGA installed.
- The path to the top level OpenEye `bin` directory must be in the **PATH** environment variable on all machines, and before any other locations that may contain MPI executables (`orted`, `mpirun`, etc).

Note: While the OpenEye MPI install must be first in the path, it will not conflict with other installs. OpenEye puts two MPI related scripts in the top level OpenEye `bin` directory. The first, `oempirun`, is a replacement for `mpirun`. Given their different names, there will not be conflict. The `orted` command in the OpenEye `bin` directory is a wrapper script around the `orted` daemon. If MPI was not run with `oempirun` and an OpenEye application, it will fall back to the next instance of `orted` in the path.

2.4.1 Using Open MPI

Running OMEGA under Open MPI makes use of a wrapper script `oempirun` located in the OpenEye `bin` directory.

To run OMEGA under Open MPI:

```
prompt> oempirun [Open MPI options] omega2 [OMEGA Options]
```

Begin by generating a text file that will include the MPI hosts you plan to use and the number of processes on each (for this example, we'll call this file `hosts`). The file should contain a line for each machine with the name of the machine, a space, then `slots=N`, where N is the number of processors for your run. For this example, you would want the file to look like:

```
c1 slots=4
c2 slots=4
c3 slots=4
c4 slots=4
c5 slots=4
```

Now the following OMEGA command-line will start a job with 1 master and 19 slaves.

```
prompt> oempirun -hostfile hosts omega2 [OMEGA Options]
```

The master will be on c1, where the job is being started. All the i/o for the run will be from the master machine, and all results and logging information will be combined.

The command line may also be used to specify hosts. For example, to run 3 OMEGA processes on the hosts larry, curly and moe:

```
prompt> oempirun -np 3 -host larry,curly,moe omega2 -in molecules.oeb -out database.oeb
```

The above command line will run 3 OMEGA processes on the hosts larry, curly and moe. The master machine, which needs a license, would be larry. It is important to note that the omega2 passed to oempirun does not take a path.

2.4.2 Open MPI License

Most files in this release are marked with the copyrights of the organizations who have edited them. The copyrights below generally reflect members of the Open MPI core team who have contributed code to this release. The copyrights for code used under license from other parties are included in the corresponding files.

```
Copyright (c) 2004-2008 The Trustees of Indiana University and Indiana
                        University Research and Technology
                        Corporation. All rights reserved.
Copyright (c) 2004-2008 The University of Tennessee and The University
                        of Tennessee Research Foundation. All rights reserved.
Copyright (c) 2004-2008 High Performance Computing Center Stuttgart,
                        University of Stuttgart. All rights reserved.
Copyright (c) 2004-2007 The Regents of the University of California. All rights reserved.
Copyright (c) 2006-2008 Los Alamos National Security, LLC. All rights reserved.
Copyright (c) 2006-2008 Cisco Systems, Inc. All rights reserved.
Copyright (c) 2006-2008 Voltaire, Inc. All rights reserved.
Copyright (c) 2006-2008 Sandia National Laboratories. All rights reserved.
Copyright (c) 2006-2008 Sun Microsystems, Inc. All rights reserved.
                        Use is subject to license terms.
Copyright (c) 2006-2008 The University of Houston. All rights reserved.
Copyright (c) 2006-2008 Myricom, Inc. All rights reserved.
Copyright (c) 2007-2008 UT-Battelle, LLC. All rights reserved.
Copyright (c) 2007-2008 IBM Corporation. All rights reserved.
Copyright (c) 1998-2005 Forschungszentrum Juelich, Juelich Supercomputing
                        Centre, Federal Republic of Germany
Copyright (c) 2005-2008 ZIH, TU Dresden, Federal Republic of Germany
Copyright (c) 2007      Evergrid, Inc. All rights reserved.
Copyright (c) 2008      Institut National de Recherche en
                        Informatique. All rights reserved.
Copyright (c) 2007      Lawrence Livermore National Security, LLC.
                        All rights reserved.
```

Copyright (c) 2007-2008 Mellanox Technologies. All rights reserved.
Copyright (c) 2006 QLogic Corporation. All rights reserved.

Additional copyrights may follow

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

2.5 Uninstallation

2.5.1 Linux/Unix

To uninstall a single distribution of a product the relevant subdirectories for that product and version simply need to be deleted from within the following directories:

arch In the `openeye/arch` directory is a platform specific subdirectory. Within this are directories for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled. For example, to delete or uninstall v1.0.0 of a product, delete the folder "`<product_name>/1.0.0`".

data In the `openeye/data` directory is a subdirectory for each installed product and within those are

subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

docs In the `openeye/docs` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

examples In the `openeye/examples` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

2.5.2 Windows

Installation of an OpenEye product on Windows causes an OpenEye group with an application specific subgroup to be added to the *Start* menu. One of the items in the application specific subgroup is a link to the uninstaller. Clicking on the uninstaller initiates a wizard which guides the user through uninstallation.

For GUI applications, uninstallation also removes the desktop link to the application as well as in the application specific subgroup of the *Start* menu.

2.5.3 Mac OS X

To uninstall a single distribution of a GUI application simply drag the application from `/Applications/OpenEye/bin` to the Trash can.

To uninstall a single distribution of a command line application you will need to delete the executable/folder from `/Applications/OpenEye/arch/osx-10.6-x64/`. For example, to delete or uninstall v1.0.0 of a product, delete the folder "1.0.0" located in `/Applications/OpenEye/arch/osx-10.6-x64/<product_name>`.

Associated documentation, data and example files for a single distribution can be uninstalled by deleting the subdirectory/folder from within the `/Applications/OpenEye/data`, `/Applications/OpenEye/docs` and `/Applications/OpenEye/examples` directories.

OMEGA THEORY

OMEGA is composed of two main components; model building and torsion driving. The components are independent and can be used separately from each other. Models can be generated without performing a torsion search. Model generation may be bypassed by importing structures from external sources.

OMEGA builds initial models of structures by assembling fragment templates along sigma bonds. Input molecules' graphs are fragmented at exocyclic sigma, and carbon to heteroatom acyclic (but not exocyclic) sigma bonds. Conformations for the fragments are either retrieved from pregenerated libraries built with makefraglib, or constructed on-the-fly using the same distance constraints followed by geometry optimization protocol that makefraglib uses. Molecule assembly is accomplished by simple vector alignment since all inter-fragment joints are along sigma bonds.

Once an initial model of a structure is constructed, or given as input, OMEGA generates additional models by enumerating ring conformations and invertible nitrogen atoms. Ring conformations are taken from the same fragment library used to build an initial model. OMEGA detaches all exocyclic substituents from a ring system, aligns and attaches them relative to the new ring conformation. OMEGA attempts to generate every possible combination of ring conformations possible for a given structure.

The next step in model generation is to detect and enumerate invertible nitrogens. Nitrogens that have pyramidal geometry, no stereochemistry specified, no more than one hydrogen, are three valent, and have no more than three ring bonds are considered by OMEGA to be invertible. Invertible in this context simply means that at room temperature a pyramidal nitrogen is likely to be able to rapidly (on an NMR timescale) interconvert between two puckered forms. All multiconformer ring models are further expanded by enumerating all possible nitrogen puckers. The resulting model set is the starting point for conformer search by torsion driving.

OMEGA begins the torsion search process by examining the molecular graph and determining the bonds may freely rotate. By default, OMEGA selects acyclic sigma bonds that have at least one non-hydrogen atom attached to each end of the bond. Hydrogen rotors (i.e. hydroxyl groups) are not altered during the torsion search. Doing so would make a combinatorially nasty problem even worse with no affect on the final ensemble. The final ensemble selection is based on heavy atom RMS distance which is unaffected by hydrogen positions. A list of possible dihedral angles are then assigned to each rotatable bond. The current mechanism for assignment is based on SMARTS matching, although alternate strategies for assigning angles based on experimental (i.e. X-ray) or theoretical (i.e. fragment optimization studies) are possible. The molecular graph is then subjected to pattern and geometric symmetry detection. Common patterns such as para-disubstituted benzene are used to reduce the number of symmetry equivalent dihedral angles that need to be searched. All torsions are altered by 120 and 180 degrees, and an RMS calculation is performed taking into account symmetry equivalent atoms in order to detect two and three fold symmetries. Torsions are then grouped into fragments of sets of up to five contiguous rotatable bonds. Exhaustive depth first torsion search is performed on each of the fragments, and the resulting conformers are placed into list sorted by energy. Entire structures are assembled by combining the lowest energy set of fragments, and then the next lowest set, until the search is terminated. Termination conditions include a limit on the total number of conformers that may be generated, fragment list may be exhausted, or the sum of the fragment energies exceeds the energy window of the global minimum structure. The best conformers identified in the torsion search are rank ordered base on energy. A final ensemble is selected by sequentially testing the conformers using the RMS distance cutoff. To be accepted in to the final ensemble, a conformer must have an

RMS distance to every other member of the ensemble that exceeds the user defined cutoff value. The final ensemble is populated up to the user defined maximum ensemble size limit, or until the list of low energy conformers is exhausted.

3.1 Filtering

Filtering based on graph (and possibly physical) properties should always be carried out prior to generating multi-conformer databases using OMEGA. Eliminating undesirable compounds prior to generating conformers will save execution time of both OMEGA and down stream applications, and space on disk. Large polypeptides or proteins, very flexible molecules, or simply molecules that would never be considered useful for the ultimate modeling application are best eliminated from a data set as early as possible.

The most important graph filter to apply is rotatable bond count. Although OMEGA may be able to generate conformers for molecules with more than 20 rotatable bonds, the results of such an exercise would be dubious at best for any conformer generation method. Number of rings, especially flexible rings, should also be used to exclude irrelevant molecules. OMEGA will handle molecules that have many thousands of possible ring combinations, although the time expenditure may be prohibitive and the results equally questionable to molecules with an unreasonably large number of rotatable bonds. Simple element filters may be useful as well, although OMEGA will discard compounds for which no force field parameters exist. Using element filters beforehand may simply aid in tracking the rationale for discarding compounds instead of searching through OMEGA log files for failure modes.

The *filter* program from OpenEye Scientific Software provides all of the functionality outlined above, and additional physical property filters. It is highly recommended that *filter* or a program similar in functionality be used for input file preparation of large datasets.

3.2 Stereochemistry Enumeration

Compounds that contain unspecified or ambiguous definitions of stereochemistry may be preprocessed before generating conformers to add explicitly specify stereochemistry. Input molecules that have three dimensional coordinates inherently have stereochemistry specified, but SMILES or two dimensional SD files may have atoms (R/S) or bonds (E/Z) for which the stereochemistry is unknown or unspecified. OMEGA will generate structures for compounds of unknown configuration, however, in virtual screening exercises it may be beneficial to simply enumerate possible stereoisomers and treat each stereoisomer as a separate compound.

OMEGA distributions contain a utility called *flipper* that enumerates unspecified stereochemistry within user defined limits. For an explanation of how to use *flipper*, consult the *flipper: Enumerating Stereocenters* chapter in the OMEGA application manual. Stereochemistry enumeration is an exponential task. For every atom or bond (N) in a molecule that has two possible stereochemical 'states', there are 2^N possible stereoisomers for the molecule. Enumerating all possible stereoisomers for molecules may be unreasonable in terms of CPU time or storage space. *flipper* has user defined limits to regulate the enumeration process and keep it practical for routine applications.

Enumerating stereoisomers provides an information gain that may aid in operations downstream from conformer generation. *flipper*, or a similar stereochemistry enumeration tool should be used prior to generating conformers with OMEGA.

3.3 Fragment Library Generation

Prior to conformer generation, OMEGA builds a set of three dimensional models of a compound that contain the bond lengths, angles, and ring conformations that will be held fixed during the torsion search. OMEGA is capable of generating fragment templates on-the-fly, however, using pregenerated templates is far more efficient and will speed execution of conformer generation. OMEGA distributions include a program called *makefraglib* which can be used to create libraries of molecular fragments and ring conformers that can be used by OMEGA to build three

dimensional models of molecules. For an explanation of how to use *makefraglib*, consult the *makefraglib: Creating a Fragment Library* chapter in the OMEGA application manual. In practice, extensive fragment libraries need only to be constructed a single time and rarely need to be altered. Corporate and vendor databases can be used to construct an extensive fragment library. Once built, the fragment libraries will rarely need to be updated, and will provide a significant enhancement in performance.

OMEGA APPLICATION USAGE

4.1 Command Line Interface

A description of the command line interface can be obtained by executing OMEGA with the `--help` option.

```
prompt> omega2 --help
```

will generate the following output:

```
Help functions:
  omega2 --help simple      : Get a list of simple parameters
  omega2 --help all        : Get a complete list of parameters
  omega2 --help <parameter> : Get detailed help on a parameter
  omega2 --help html       : Create an html help file for this program
```

4.1.1 Required Parameters

-in

File containing one or more molecular connection tables to be processed by OMEGA.

-out

File to write conformers generated by OMEGA. Gzipped OEBinary is the recommended output format.

Execute Options

-param

The argument for this flag is the name of a file containing control parameters. The control parameter file acts to either replace or augment the command line interface. All parameters necessary for program execution may be provided in the control parameter file, although any command given explicitly on the command line will supersede options found in the parameter file. OMEGA generates a new parameter file containing the full set of execution parameters upon every execution. The name of the parameter file written by OMEGA is created by combining the prefix base name with the `‘.parm’` extension.

-pvmconf

A text file specifying a PVM configuration. The `-pvmconf` option is only available on platforms where OMEGA supports PVM. For every host in the cluster it should contain a line:

```
host host_name n
```

where `n` is the number of processors on the host.

File Options

-commentEnergy

This flag causes the conformer energy in kcal/mol to be written in the comment field for each conformer. This is particularly useful when writing SD or MOL2 files that will be passed to software that anticipates strain energies written in the comment field. [default = false]

-includeInput

When true this boolean flag will include the input conformer in the output file. This requires that the input file format is a 3D format (eg: SDF, MOL2, OEB, etc) [default = false]

-log

The argument for this flag specifies the name of the log file. The level of detail for logfile information can be altered using the `-verbose` flag. Output can be directed to the terminal instead of a file by giving a hyphen '-' as the argument to the flag instead of a filename. Generation of an output log may be disabled by providing 'nul' or 'null' as an argument. [default = *prefix.log*]

-pendingFile

Filename used for pending molecules file.

-prefix

The argument for this flag defines the prefix to be used for various information and data files generated by OMEGA. Most important among these is the '*omega2.parm*' file which includes a copy of all the parameters used in the OMEGA run. The prefix is also used to generate a default log file name if not explicitly specified with the `-log` flag. [default = *omega2*].

-progress

Show progress on screen. Options are 'none', 'dots', 'log' and 'percent'. The 'dots' options will displays dots on screen to show molecules completed. The 'log' option will duplicate the log file on screen. The 'percent' option will track progress through the input file. [default = none]

-rotorOffsetCompress

This flag controls the behavior of writing the rotor offset compressed flavor of the OEBinary format. Rotor offset compressed files contain the same information as standard Cartesian OEBinary files, but require a small fraction of the storage space. Optimal compression of OMEGA output can be obtained by writing GZip compressed OEBinary files with the `-rotorOffsetCompress` flag turned on. [default = true]

-sdEnergy

This flag controls the behavior of writing strain energies as SD tags. This flag may be used with either OEBinary or MDL SD files. [default = false]

-status

Filename used for status report file

-verbose

This is a boolean flag that controls the level of detail written to the log file. By default OMEGA will only write minimal information to the log file. Molecule titles and warning messages constitute the bulk of logging at the default level. Verbose logging will cause more information to be written to the log file in order to follow behavior during program execution. [default = false]

-warts

This boolean flag is used to generate unique titles for conformers that reflect their position in an ensemble of conformations produced by OMEGA. The title given to each conformer will begin with the molecule title taken from the input file, and appended with an underscore and the integer corresponding to the rank order number of the conformer in the final ensemble. [default = false]

3D Construction Parameters

-addfraglib

The argument that follows this flag is one or more fragment files, normally generated by *makefraglib*. These fragments will supplement the built-in fragment library

-buildff

This flag sets the force field used for constructing fragments that are assembled to build an initial model of the input structure. Consult the description of Force Fields (see *Force Fields*) for an explanation of appropriate arguments for this flag. [default = mmff94s_NoEstat]

-canonOrder

This flag can be used to disable the automatic reordering of input molecules to a canonical atom and bond order. In order obtain consistent results from different file formats and different connection table orders, OMEGA reorders the input connection table. This behavior can be turned off using the *-canonOrder* flag, however, the resultant ensembles will likely be inconsistent in their composition. [default = true]

-deleteFixHydrogens

When a fixfile fragment is specified, all possible mappings (matches) of the fragment and the input molecule are considered for positioning the input molecule. This process begins with a substructure search of the fragment in the input molecule. If hydrogens are included on the fixfile fragment, the number of possible matches will grow exponentially with the number of equivalences. Each geminal hydrogen pair on the fixfile produces a non-productive match multiplied by all other non-productive matches. This flag can be used to prevent the explosion of non-productive matches by deleting hydrogens from the fixfile fragment prior to the substructure search. [default = false]

-dielectric

The argument to this flag allows the user to specify the dielectric applied to the coulomb term of the force field. This flag may only be used with a version of the search force field that includes the coulomb term, such as the MMFF94 and MMFF94s variants. [default = 1.0]

-exponent

The argument to this flag allows the user to specify the exponent applied to the inverse distance calculation of the coulomb term of the force field (i.e. $(1/r)^X$ where X is the argument to the *-exponent* flag). This flag may only be used with a version of the search force field that includes the coulomb term, such as the MMFF94 and MMFF94s variants. The legal values are 1 or 2. [default = 1]

-fixfile

The argument that follows this flag is a molecule file used to specify the coordinates for a substructure of the input molecules. An initial structure is generated for every input molecule, and then a substructure search is performed using the molecule or fragment provided in the fixfile as a query molecule. Every instance of the fixed substructure found in the input molecule up to a predetermined limit (see *maxmatch* and *umatch*) is used to replace the coordinates of the atoms that match the substructure. The input molecule coordinates are aligned relative to the substructure prior to fragment replacement, and then the coordinates are taken from the fixed fragment and assigned to the corresponding atoms of the input molecule. A separate alignment, replacement, and then conformer search is carried out for every matching substructure in the input molecule.

-fixrms

Fixfile fragments taken from crystallographic sources may differ in their geometry relative to optimal MMFF geometries. OMEGA attempts to superimpose built structures onto fixfile fragments and, if the geometry differs too greatly, OMEGA considers the superposition a poor match and will fail to build a structure using the fixfile. This flag can be used to loosen the default RMS superposition criteria to allow suboptimal superpositions to succeed in spite of the poor geometric complementarity. [default = 0.15]

-fixsmarts

Another way to specify a fixed portion of a molecule. The SMARTS pattern is used to fix a portion of the *-fixfile* primarily, or fix or portion of the input molecule if *-fromCT* is set to *false* secondarily.

-fromCT

This boolean flag determines whether OMEGA should generate an initial set of 3D coordinates using only the connection table of the input molecule. Initial model generation is always necessary for molecule file formats devoid of coordinates (i.e SMILES). Bond lengths and angles taken from molecule files containing coordinates may be retained by setting this flag to false. [default = true]

-maxmatch

This flag is used to limit the number of fixfile substructure matches in the input molecule. Each match will result in replacement of the matching substructure with coordinates taken from the fixfile fragment. The number of matches may need to be limited using this parameter for a substructure where many matches are possible. [default = 10]

-setfraglib

The argument that follows this flag is an OEBinary molecule file containing the coordinates of pre-built acyclic fragments, and multiple conformations of cyclic systems. Starting with OMEGA2.2, this flag is no longer required, as a default fragment library has been built into the program. Although OMEGA can generate fragment 3D geometries on the fly, building them in advance speeds execution significantly. Multiple fragment files may be provided as a comma delimited list of filenames. The fragment file(s) chosen should be constructed using a force field corresponding to the one being used for the torsion search. For example, the MMFF94s variant of the Merck Molecular Force Field should be used both for fragment construction and torsion search. Normally this file is built using the *makefraglib* auxiliary program.

-strictfrags

This flag sets how OMEGA generates fragments that are not in the fraglib. The default is a faster and less rigorous fragment generation than the makefraglib program, whereas a 'true' value will require OMEGA to be identical to the makefraglib program. The most noticeable difference is the time limit for fragment generation, which is increased from 30 seconds to 300 seconds with *-strictfrags* set to `true`. [default = false]

-strictatomtyping

Use strict atom typing for MMFF94 or allow 'close enough' atom typing. A *true* setting will fail any molecule that contains an atom type that does not have specified parameters. A *false* setting will allow parameters from a similar atom type to be used. [default = true]

-umatch

The *-umatch* boolean flag determines whether only the unique substructure matches of the fixfile are used for coordinate replacement. A unique substructure match is defined as a match that does not cover the identical set of target atoms as any other substructure match in a set. For example, a benzene substructure will match a benzene ring 12 times. Only one substructure match constitutes a unique match, while the other 11 matches are duplicates. If the flag is set to false then all possible substructure matches may be used for coordinate replacement. This behavior is usually unnecessary as non-unique matches will frequently lead to duplication. [default = true]

Structure Enumeration

-enumNitrogen

The *-enumNitrogen* string flag controls the behavior of OMEGA with respect to enumeration of non-planar nitrogens. Any nitrogen with pyramidal geometry in the initial model of the input molecule, and having no more than two ring bonds is considered by OMEGA to be 'invertible'. OMEGA will enumerate all possible puckers if the *-enumNitrogens* flag is set to `true`. OMEGA will only enumerate unspecified invertible nitrogens if the flag is set to *unspecified*. [default = true]

-enumRing

The *-enumRing* boolean flag controls the behavior of OMEGA with respect to ring conformations. If this flag is set to true, OMEGA will generate all possible combinations of all ring conformations in a molecule. Ring systems with only a single conformation will be replaced with a conformation taken from a fragment file, or generated on the fly by OMEGA. If this flag is set to false then no ring conformer enumeration or replacement

will occur. Initial geometries provided in by an input file (see *-fromCT*) may therefore be preserved by setting *-enumRing* to false as well. [default = true]

Torsion Driving Parameters

-addtorlib

Takes a filename as a parameter. Torsion rules in the file are placed above any previous torsion rules and therefore are matched first. (see *Torsion Library Format*)

-erange

The *-erange* flag sets the energy cutoff used as an accept or reject criteria for conformers depending on the number of rotatable bonds in the structure. Any conformer that has a calculated strain energy less than the sum of the energy window and the energy of the global minimum conformer will be accepted. Conformers with strain energies above this threshold are rejected. The energy range is given as a comma separated list of values that correspond to the *-rangeIncrement* parameter. For example, *-erange "5.0, 10.0, 15.0, 20.0"* used with *-rangeIncrement 3* sets the energy window to 5.0 Kcal/mol for structures with zero to two rotatable bonds, 10.0 Kcal/mol for structures with three to five rotatable bonds, and so on. The energy window for structures with more rotors than the highest *-erange* value specified will be taken as the highest specified value.

-ewindow

The *-ewindow* flag sets the energy window in kcal/mol used as an accept or reject criteria for conformers. Any conformer that has a calculated strain energy less than the sum of the energy window and the energy of the global minimum conformer will be accepted. Conformers with strain energies above this threshold are rejected. [default = 10.0]

-maxConfRange

This string argument to this flag allows the user to specify the maximum number of conformers to be output for a structure based on the number of rotatable bonds in the structure. For example, *-maxConfRange "100,200"* used with *-rangeIncrement 5* will cause OMEGA to output 100 conformers for structures with zero to 4 rotors, and 200 conformers for all structures with more than 4 rotors.

-maxconfs

The *-maxconfs* flag sets the maximum number of conformations to be generated. Conformers are assembled in energy sorted order of the constituent fragments. The default value in most cases will vastly exceed the number of conformers that need to be generated in order to select the best possible ensemble based on the RMS distance and energy criteria. As a special case, setting *-maxconfs 0* will result in OMEGA skipping the duplicate removal step and it will write all generated conformers to the output file. Note that this implies *-rms 0* is also used. [default = 200]

-maxrot

The *-maxrot* flag sets the maximum number of rotatable bonds cutoff. Molecules that have equal to or fewer rotors than the *-maxrot* cutoff will be processed by OMEGA. OMEGA will not search for conformers of molecules that have more rotors than the *-maxrot* cutoff. By default, OMEGA does not apply a number of rotatable bonds cutoff. Instead, a desired cutoff must be supplied by the user. [default = -1]

-maxtime

This flag limits the amount of time (in seconds) spent generating conformers for each molecule. [default = 120.0]

-rangeIncrement

The *-rangeIncrement* is used to control the number of rotatable bonds range used with the *-maxConfRange*, *-rmsrange*, and *-erange* flags. The preceding flags are used to control the maximum number of conformers, RMS cutoff, and energy windows used that are dependent on the number of rotors in a given structure. [default = 5]

-rms

The *-rms* flag sets the minimum Root Mean Square (RMS) Cartesian distance below which two conformers are

duplicates. The RMS calculation is performed after superposition such that the true minimum distance between conformers is calculated. Lowering the *-rms* value may cause OMEGA to generate ensembles that contain more representative conformers of a similar shape. Higher *-rms* values may result in smaller, yet possibly more shape diverse ensembles. [default = 0.5]

-rmsrange

This string argument to this flag allows the user to specify the RMS cutoff used for duplicate conformer removal based on the number of rotatable bonds in the structure. For example, *-rmsrange "0.8,1.0"* used with *-rangeIncrement 5* will cause OMEGA to use an RMS cutoff value of 0.8 for structures with zero to 4 rotors, and an RMS cutoff value of 1.0 for all structures with more than 4 rotatable bonds.

-searchff

This flag sets the force field used to calculate strain energies of conformers generated during a torsion search. Consult the description of Force Fields (see *Force Fields*) for an explanation of appropriate arguments for this flag. [default = mmff94s_NoEstat]

-settorlib

The *-settorlib* flag is used to specify the file name of the file containing rules used in resolution control of the torsion driving part of conformer generation. Refer to the section describing the format of the torsion library (see *Torsion Library Format*). If no torsion library is provided then OMEGA will use an internally stored copy of the default torsion library.

Flipper Parameters

-flipper

Creates an ensemble of stereoisomers before passing the molecules for conformer generation. A value of *'true'* will flip all unspecified stereo centers and a value of *'force'* will flip all stereo centers. [default = false]

-flipper_maxcenters

Sets the maximum number of stereocenters that can be flipped exhaustively. [default = 12]

-strictstereo

Requires all stereo centers to be specified. Molecules sent to conformer generation with unspecified stereo will fail. [default = true]

General

-strict

Convenience flag for setting *-strictstereo*, *-strictatomtyping*, and *-strictfrags* at once. This setting will override any individual settings for these flags.

PVM

-pvmdebug

Generate an enormous volume of PVM debug information.

-pvmlog

Filename used for PVM log file

-pvmpass

Number of molecules to pass to a slave at one time

4.1.2 Example Executions

This section has a series of example OMEGA command-line executions. Each example is followed by a brief description of its behavior. Sample data files can be found in *openeye/examples/omega*.

```
prompt> omega2 drugs.smi drugs.oeb.gz
prompt> omega2 -in drugs.smi -out drugs.oeb.gz
```

These two commands will yield identical results. These execute OMEGA with the default parameters. The file *drugs.smi* is opened in SMILES format for input, and the output is written to the file *drugs.oeb.gz* in gzipped OEBinary format.

```
prompt> omega2 -in drugs.smi -out drugs.oeb.gz -param myparameters
```

This command is the same as the one above except for the *-param* flag. It executes OMEGA with the parameters found in the *myparameters* file. The file *drugs.smi* is opened in SMILES format for input, and the output is written to the file *drugs.oeb.gz* in gzipped OEBinary format.

```
prompt> omega2 -param myparameters drugs.smi drugs.oeb.gz
prompt> omega2 drugs.smi drugs.oeb.gz -param myparameters
```

The first of these two commands will yield exactly the same results as the example above. *drugs.smi* will be mapped to the *-in* flag and *drugs.oeb.gz* will be mapped to the *-out* flag begin the second to last and last command-line arguments respectively. Unfortunately, the second of these two commands, will fail to parse because the implicit input and output arguments are not the final two arguments in the list.

```
prompt> omega2 -in drugs.smi -out drugs_maxconfs600.oeb.gz -param myparameters -maxconfs 600
```

Again, this is a very similar command. It executes OMEGA using the parameters in the *myparameters* file, except the *-maxconfs* parameter is over-ridden with the 600 from the command line. The command-line *-maxconfs* parameter would take precedence over the value in the parameter file independent of the order of flags on the command line.

```
prompt> DBQuery "barbiturate" | omega2 -in .sdf -out .oeb.gz | vida2 .oeb.gz
```

This execution assumes that a process called “DBQuery” can be called with the parameter “barbiturate” and return a stream of molecules in MDL’s *.sdf* format. This output is piped into OMEGA, which interprets the format correctly and generates multiconformer molecules using the default parameters. OMEGA writes the output to *std::cout* in gzipped OEBinary format, which is read by OpenEye’s VIDA molecular viewer.

```
prompt> omega2 -in drugs.mol2 -out drugs.oeb.gz -fromCT true
```

The *-fromCT true* flag will cause OMEGA to ignore the input conformations in the *drugs.mol2* file. An initial conformation will be generated by a distance-bounds algorithm from the connection-table of the molecules in the input file. Be aware that the default value of *-fromCT* is true, however it is listed explicitly here for emphasis.

```
prompt> omega2 -in drugs.smi -out drugs.oeb.gz -log null
```

The *-log* flag normally allows specification of the log file’s name. However, *nul* and *null* are reserved names which indicate that no log file will be written (this includes failure modes).

4.2 Further specifications

4.2.1 Force Fields

OMEGA provides the facility for users to select one of a number of pre-defined force fields. The force field selected may be different for model construction and torsion search. The ability to select a force field provides a mechanism for task specificity. Some force fields may be more appropriate for solution phase ensemble generation, whilst others may excel for bioactive conformer reproduction. The following pre-defined force fields may be used as arguments to the *-buildff* and *-searchff* flags.

- *mmff* Exact reproduction of the published Merck Molecular Force Field (MMFF) with addition atom types parameterized using the same algorithm.
- *mmff_NoEstat* This force field variant includes all MMFF terms except coulomb interactions.
- *mmff_Trunc* This force field variant excludes both coulomb interactions and the attractive part of Van der Waals interactions. All other components of the MMFF force field are calculated according to literature specifications.
- *mmff94s* Exact reproduction of the 94s variant of the Merck Molecular force Field (MMFF94s) with addition atom types parameterized using the same algorithm.
- *mmff94s_NoEstat* This force field variant includes all MMFF94s terms except coulomb interactions.
- *mmff94s_Trunc* This force field variant excludes both coulomb interactions and the attractive part of Van der Waals interactions. All other components of the MMFF94s force field are calculated according to literature specifications.

4.2.2 Torsion Library Format

A file of alternate torsion rules may be specified with the *-settorlib* command. OMEGA will match only the first rule found for a torsion angle, and discontinue matching alternate possible rules. Thus, a correctly ordered torsion file will be arranged with the most specific patterns appearing at the top of the file, and more general patterns appearing toward the end. Simple torsion rules are composed of a single SMARTS pattern containing at least four atom expressions, followed by a listing of the torsion angles that OMEGA will sample. Each reference atom in the SMARTS pattern that is used to define the torsion angle being sampled must have a map index (numbered 1 through 4) specified that indicates the ordering of the atoms in the torsion angle. The pattern must appear all on a single line, with a carriage return separating one rule from the next. Comments in the file must be preceded with a # character. The following is an example of a simple torsion rule.

```
#methyl ester pattern
[O:1]=[C:2]-[O:3][CH3:4] 0
```

More advanced rules may be included that alter the energy calculation for particular torsion angles. In these types of torsion rules, a SMARTS pattern with associated map indices is still used to define the molecular environment in which the rule is to be applied, but the sampled values appear on subsequent lines with one torsion angle per line. The first number per line indicates the torsion angle, in degrees, that OMEGA must sample. If a second number follows a torsion angle on the same line, the value is added to the total energy computed for that conformer. The following is an example of an advanced torsion rule.

```
#experimental structure test
O=[C:1][NX3H:2][c:3]([cH,nH0])[nH:4]
0
180 10.0
<end>
```

MAKEFRAGLIB: CREATING A FRAGMENT LIBRARY

makefraglib identifies, extracts, and generates conformers of molecular fragments. The model builder in OMEGA fragments input molecules, retrieves corresponding fragment conformers, and assembles them into a three dimensional structure. Pregenerating fragment conformers using *makefraglib* accelerates the model building procedure.

5.1 *makefraglib* Theory

The model builder in OMEGA attempts to capture all of the relevant bond length and angles, and ring conformations for an input molecule. Fragment assembly allows for a ‘divide and conquer’ approach to model building. Much of the relevant information for building a molecular models is contained in carefully chosen fragments. *makefraglib* uses distance constraints and geometry optimization to generate fragment conformations. Only a single conformation is stored for acyclic fragments, while all possible unique ring conformers are retained within user defined limits. Libraries created with *makefraglib* can then be provided to OMEGA using the *-setfraglib* or *-addfraglib* flags.

5.2 *makefraglib* Usage

makefraglib can be used from the command-line to generate ring templates for use with OMEGA starting from only a collection of molecules. The output file from *makefraglib* can then be passed directly into OMEGA using the *-setfraglib* or *-addfraglib* flag. Although fragment libraries may be concatenated using the Unix ‘cat’ command, this is not strictly necessary as multiple fragment libraries may be specified as one argument to the flag.

5.2.1 Required Parameters

-in

File containing one or more molecules from which fragments will be generated

-out

File to write fragments generated by *makefraglib*. Gzipped OEBinary is the recommended output format.

Execute Options

-param

The argument for this flag is the name of a file containing control parameters. The control parameter file acts to either replace or augment the command line interface. All parameters necessary for program execution may

be provided in the control parameter file, although any command given explicitly on the command line will supersede options found in the parameter file. OMEGA generates a new parameter file containing the full set of execution parameters upon every execution. The name of the parameter file written by *makefraglib* is created by combining the prefix base name with the *.parm* extension.

-pvmconf

A text file specifying a PVM configuration. For every host in the cluster it should contain a line:

```
host host_name n
```

where n is the number of processors on the host.

File Options

-log

The argument for this flag specifies the name of the log file. The level of detail for logfile information can be altered using the *-verbose* flag. Output can be directed to the terminal instead of a file by giving a hyphen '-' as the argument to the flag instead of a filename. Generation of an output log may be disabled by providing 'nul' or 'null' as an argument. [default = *prefix.log*]

-prefix

The argument for this flag defines the prefix to be used for various information and data files generated by OMEGA. Most important among these is the *makefraglib.parm* file which includes a copy of all the parameters used in the *makefraglib* run. The prefix is also used to generate a default log file name if not explicitly specified with the *-log* flag. [default = *makefraglib*].

-progress

Show progress on screen. Options are 'none', 'dots', 'log' and 'percent'. The 'dots' options will displays dots on screen to show molecules completed. The 'log' option will duplicate the log file on screen. The 'percent' option will track progress through the input file. [default = none]

-skip

Existing fragment library to avoid duplicating

-verbose

This is a boolean flag that controls the level of detail written to the log file. [default = false]

3D Construction Parameters

-buildff

This flag sets the force field used for constructing fragments that are assembled to build an initial model of the input structure. Consult the description of Force Fields (see *Force Fields*) for an explanation of appropriate arguments for this flag. [default = *mmff94s_NoEstat*]

-ewindow

The *-ewindow* flag sets the energy window used as an accept or reject criteria for fragments. Any fragment that has a calculated strain energy less than the sum of the energy window and the energy of the global minimum fragment will be accepted. Conformers with strain energies above this threshold are rejected. [default = 4.0]

-fromCT

This boolean flag determines whether *makefraglib* should generate an initial set of 3D coordinates using only the connection table of the input molecule. Initial model generation is always necessary for molecule file formats devoid of coordinates (i.e SMILES). Bond lengths and angles taken from molecule files containing coordinates may be retained by setting this flag to false. [default = true]

-rms

The *-rms* flag sets the minimum Root Mean Square (RMS) Cartesian distance below which two fragments are duplicates. The RMS calculation is performed after superposition such that the true minimum distance between fragments is calculated. Lowering the *-rms* value may cause *makefraglib* to generate ensembles that contain more fragment conformers of a similar shape. Higher *-rms* values may result in smaller, yet possibly more shape diverse ensembles. [default = 0.1]

-startfact

Factor for determining number of random starting geometries when generating fragments.

5.2.2 Example Executions

The following section shows several common command-line executions of *makefraglib*. Each example is followed by an explanation of what the program will do. Sample data files can be found in *openeye/examples/omega*.

```
prompt> makefraglib -in drugs.smi -out fraglib.oeb
```

All of the molecules in *drugs.smi* will be processed. The set of all unique fragments (according to OMEGA fragmentation rules) from all of the molecules in *drugs.smi* will be collected.

```
prompt> makefraglib -in vendor.smi -out new_fraglib.oeb -skip fraglib.oeb -ewindow 5.0
```

Conformers will be generated for all of the unique fragments found in the user supplied file *vendor.smi* that are not present in *fraglib.oeb*. An energy window of 5 Kcal above the global minimum conformer will be used to accept or reject conformations of flexible ring systems.

FLIPPER: ENUMERATING STEREOCENTERS

6.1 *flipper* Theory

The OMEGA distribution includes *flipper*, a utility program for enumerating stereocenters in a molecule. We have taken the philosophy that conformer generation and stereochemistry enumeration are related but distinct problems. Thus we provide *flipper* as part of the OMEGA product, but not as part of the same executable.

The current version of *flipper* can handle enumeration of R/S stereochemistry (including N and S stereochemistry) as well as *cis/trans* stereochemistry. *flipper* uses graph algorithms to determine which atoms are stereocenters. If a stereocenter has a designated stereochemistry, by default *flipper* does not change the stereochemistry. However, if a stereocenter does not have a specified stereochemistry, *flipper* will enumerate both stereochemistry states of the stereocenter.

Many common file formats are limited in their ability to fully express stereochemistry without resorting to Cartesian coordinates. This can lead to ambiguity in whether a particular conformer is supposed to represent a racemic mixture or an isolated stereoisomer. SD file format is better than most formats. It allows designation of R, S, or racemic on each tetrahedral stereocenter independent of the particular conformation. However, SD format is not as robust for *cis/trans* stereochemistry. Isomeric SMILES is the only common file format which allows complete specification of R/S and *cis/trans* stereochemistry. In fact, although *flipper* does not currently support higher orders of stereochemistry, the isomeric SMILES format is capable of handling them. For this reason, *flipper* output is limited to SD format (.sdf or .mol) and isomeric SMILES format (.ism). OpenEye OEBinary format, like isomeric SMILES, is capable of fully specifying stereochemistry, however, isomeric SMILES is more efficient for compounds without any data beyond the title and connection table.

6.1.1 Input dimensionality

flipper is designed for use prior to 3D coordinate generation. So 2D or 0D (SMILES) input is normally expected. Input files containing 3D coordinates are handled, but the 3D coordinates are ignored and not transferred to the output.

6.2 *flipper* Usage

flipper is a simple command-line utility program. The *flipper* process can also be done with OMEGA using the *-flipper* flag.

6.2.1 Required Parameters

-in

File containing one or more molecules for which stereoisomers will be generated

-out

This required parameter indicates the output molecule file. The output file format is limited to isomeric SMILES (*.ism* NOT *.smi*) or SD format since these are the only formats which fully specify stereocenters without the need for three-dimensional coordinates.

6.2.2 Other options

-enumNitrogen

The *-enumNitrogen* string flag controls the behavior of FLIPPER with respect to enumeration of non-planar nitrogens. Any nitrogen with pyramidal geometry in the initial model of the input molecule, and having no more than two ring bonds is considered by OMEGA to be 'invertible'. FLIPPER will enumerate all possible puckers if the *-enumNitrogens* flag is set to `true`. [default = false]

-forceflip

This parameter forces *flipper* to modify all of the stereocenters in a molecule. If *-forceflip* is false (the default), *flipper* only enumerates stereocenters which do not already have a specified stereochemistry. [default = false]

-maxcenters

Obviously, the number of molecules generated by enumerating the stereocenters is 2^N , where N is the number of stereocenters. In some instances, this may be larger than is desired. The *-maxcenters* parameter indicates the maximum number of stereocenters which will be fully enumerated. If a molecule has more than *-maxcenters* stereocenters, *flipper* will randomly enumerate $2^{(maxcenters)}$ instances from the full set of potential isomers. [default = 12]

6.2.3 Example Executions

Below are three example *flipper* executions. Each example is followed by a brief description of the parameter's effects. Sample data files can be found in *openeye/examples/omega*.

```
prompt> flipper -in drugs.smi -out enumerated.ism
```

This execution will examine each stereocenter in *drugs.smi*. If the stereocenter does not have a specified stereochemistry, *flipper* will generate molecules with each of the stereochemistry states at each stereocenter.

```
prompt> flipper -in drugs.smi -out enumerated_forceflip.ism -forceflip true
```

This execution will differ from the first in that **all** stereocenters will be enumerated, regardless of whether the stereochemistry is specified in the input file.

```
prompt> flipper -in drugs.smi -out enumerated_maxcenters6.ism -maxcenters 6
```

In this execution, *flipper* will exhaustively enumerate all unspecified stereocenters in molecules with up to 6 stereocenters. For molecules with more than 6 stereocenters, a random set of 64 isomers will be generated from the larger set of potential stereoisomers.

OEB2SDCONF

7.1 *oeb2sdconf* Usage

oeb2sdconf is a utility that allows OMEGA generated conformers to be used with the MOE program. Please contact Chemical Computing Group support if you require assistance with this utility.

RELEASE NOTES

8.1 OMEGA 2.4.6 (*February 2012*)

8.1.1 New features

- Now uses an extension of the MMFF94 force field for tricoordinate boron compounds. Most compounds containing B-X bonds where X=C,N,O,S, and H are covered with the following exceptions: X=N(imine),N(sulfonamide), N(pyridinium) and N(aternary). Also not supported are compounds in which boron is bonded to X=F,Cl,Br,I,B and Si, or makes a bond angle BYX. Compounds in which boron is a part of four-membered rings of B1CCC1 type are also not available in the current parameterization because their existence is questionable: Ab initio calculations at the MP2/6-31G** level failed to identify stable structures for them (highly polar structures in which boron is four-coordinated are formed).

8.1.2 Bug fixes

- Fixed a bug where OMEGA wasn't checking if 3D construction of fragment generation was successful.
- Fixed a bug where output from flipper could still fail in OMEGA for missing stereo chemistry. Now the rules for required stereo chemistry are consistent.
- Fixed a bug where setting stereo chemistry in too small of a ring could cause a crash. Now only rings greater than 7 members and only one double bond will be set with flipper.
- Fixed a bug where OMEGA wasn't checking if user generated fragment libraries were in OEB format. Fragment libraries must be in OEB format.
- Fixed bugs related to nonsense values for command line parameters. User-supplied parameters must be a legal value. The '-help <parameter>' command will print out the legal values.
- Fixed a bug where SD tagged data wasn't passed along to output molecules.

8.2 OMEGA 2.4.3 (*August 2010*)

8.2.1 Bug fixes

- A bug was fixed where slave nodes required a license. This has been fixed and now only the master requires a license.
- The -warts flag did not work when writing to formats other than .oeb or .oeb.gz. This has been corrected in version 2.4.3.

8.3 OMEGA 2.4.1 (June 2010)

8.3.1 New features

- FILTER is now bundled with the OMEGA distribution and will run with a valid OMEGA license.
- OMEGA and *makefraglib* may now be run with MPI using the *oempirun* script. (Not available on all platforms)
- The default for OMEGA has been changed to require all stereocenters to be specified. Molecules with unspecified stereo will fail, except for unspecified invertible nitrogens which are allowed and will not cause the molecule to fail. The previous behavior was for OMEGA to choose a random stereoisomer. This default can be changed back to the previous behavior by setting *-strictstereo* to *false*.
- By default, FLIPPER will no longer flip invertible nitrogens. The default path is for FLIPPER to leave them unspecified and allow OMEGA to enumerate them. Flipping invertible nitrogens in both FLIPPER and then OMEGA will lead to duplication. These options can be changed with the *-enumNitrogen* flag available in both applications.
- The default for OMEGA has been changed to no longer use ‘close enough’ atom typing. If a proper MMFF atom type cannot be found for an atom then the molecule will fail. Previous versions would use a different atom type of the same element if it was available. The previous behavior for atom type substitution can be enabled by setting *-strictatomtyping* to *false*.
- OMEGA now has the option to require on-the-fly generation of fragments to be the same as the *makefraglib* utility by setting *-strictfrags* to *true*. This fragment generation is more rigorous but also more time consuming.
- The convenience flag *-strict* has been added to turn on or off all of the strict options. These include *-strictstereo*, *-strictatomtyping* and *-strictfrags*.
- On-screen progress has been enabled for OMEGA and *makefraglib* using the *-progress* flag. The options are *none*, *dots*, *log*, and *percent*.
- The *-maxconfgen* and *-maxpoolsize* flags have been removed. These options are set internally and adjusted automatically subject to the user defined variable of *-maxconfs*. Additionally, hard limits have been removed from these variables and they are now only limited by available memory.
- Improper formatting of the *-erange*, *-maxConfRange*, and *-rmsrange* parameters will cause OMEGA to exit immediately. Previously the application would throw a warning but continue the calculation without using the parameters.
- The flag *-fixsmarts* has been created to allow a smarts pattern to be used to fix a portion of the molecule. This requires 3D coordinates to be available from either a 3D input file with *-fromCT* set to *false* or from a molecule set using *-fixfile*.

8.3.2 Bug fixes

- A bug was fixed in *makefraglib* where the setting *-fromCT false* could give nonsense fragments.
- Crashes have been fixed that could occur when the *-fixfile* could not find a valid match or did not have enough atoms to match against in a ring system.
- Program no longer crashes if there is not enough memory for duplicate removal. If there is not enough memory available for allocation then the application will exit immediately.

8.4 OMEGA 2.4.0 (November 2009) (Toolkit only)

This release was for the toolkit only. For more details please refer to the OmegaTK documentation.

8.5 OMEGA 2.3.3 (March 2009) (Toolkit only)

This release was for the toolkit only. For more details please refer to the OmegaTK documentation.

8.6 OMEGA 2.3.2

8.6.1 Bug fixes

- Fixed a bug where molecules containing boron or selenium could cause a crash. These molecules now are written to the *'fail'* file due to missing force field parameters and the program will proceed to the next molecule.

8.7 OMEGA 2.3.1

8.7.1 New features

- The OMEGA application now includes a utility named *oeb2sdconf* that allows OMEGA generated conformers to be used with the MOE program. Please contact Chemical Computing Group support if you require assistance with this utility.

8.7.2 Bug fixes

- Fixed a bug where the omega2 PVM features would not work on some platforms.
- Fixed a bug where the omega2 script would not function properly with filenames that had spaces in them.

8.8 OMEGA 2.3.0

8.8.1 New features

- The distribution and installation of OMEGA has been modified. The Windows distribution is now a standard .exe installer, and the OS X distribution is a dmg containing a standard pkg installer. The executables are now scripts that chose the correct version of the program at runtime. Please see the Installation and Platform Notes for details.
- The defaults for MaxConfs and RMSThreshold are changed in this version. The default for MaxConfs is reduced from 400 to 200, while the value for RMSThreshold goes from 0.8 down to 0.5. These changes are based on maintaining good reproduction of the crystal structure test set, virtual screening tests with ROCS and FRED, and pose prediction tests with FRED.
- The method `OEConfGen::OEOmega::ClearFixFile` is added to the OMEGA library. This allows an `OEConfGen::OEOmega` instance to be reused for an unrestrained search after it has previously been used for a fixed search.
- Untitled molecules are given unique titles of *omega_1*, *omega_2*, and so on. Titles allow molecules to be located in the log files.

8.8.2 Bug fixes

- Fixed a major bug where molecules that have missing torsion rules caused OMEGA to crash. Molecules with missing torsion rules are now sent to the *omega2.fail* file and are not processed. Missing torsion rules are typically caused by highly unusual molecules for which OEChem cannot properly assign valence to all atoms.
- Fixed a bug in duplicate conformer removal that caused some conformers that should have been removed to pass through as unique. This bug also caused occurrences where a reduction in maxconfs led to an increase in the number of conformers returned.
- Fixed a bug that caused SD data on single conformer molecules to be removed.
- Fixed a bug in the *flipper* library that left 3D coordinates intact, which caused stereo chemistry to be ambiguous. The *flipper* library now replaces 3D coordinates with zeros.
- Fixed a bug where the `SYM` tag from previous OMEGA calculations caused problems when those molecules were passed back in. Also, the `O2MolId` tag is now removed from molecules before they are sent to the output.
- When the number of stereo centers exceeds the number of -maxcenters *M*, *flipper* now generates 2^M random non-repeating flips. Previous versions generated 2^M random flips, leading to molecules with the same stereo-chemistry being returned multiple times.

8.9 OMEGA 2.2.2

8.9.1 Bug fixes

- Fixed a major bug that would cause highly-symmetric, highly-fluorinated compounds to crash the program or hang a slave in PVM mode.
- Fixed a bug in atom typing when building fragments from scratch that, in some rare cases, could result in differing fragment geometries depending on the order of input of the molecules.
- Added more places that check for time exceeding max time. This is more just to prevent overly large molecules from appearing to hang.
- If OMEGA is called with a fixed predicate, this now implies that FromCT is false. In other words, if you want to use a predicate to fix atoms, you need to use a 3D structure as input.

8.10 OMEGA 2.2.1

8.10.1 Bug fixes

- The new build algorithm introduced in v2.2.0 could, on rare occasions, generate a structure with heavy atom clashes. This release is mainly a bug fix for this problem.

BIBLIOGRAPHY

- [Geist-1994] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V., **PVM - Parallel Virtual Machine: A Users Guide and Tutorial for Networked Parallel Computing**; *MIT Press*, 1994
- [Bostrom-2001] Jonas Bostrom, **Reproducing the Conformations of Protein-Bound Ligands: A Critical Evaluation of Several Popular Conformational Searching Tools**, *Journal of Computer-Aided Molecular Design (JCAMD)*, Vol. 15, pp. 1137, 2001
- [Bostrom-2003] J. Bostrom, J.R. Greenwood, and J. Gottfries, **Assessing the Performance of OMEGA with Respect to Retrieving Bioactive Conformations**, *Journal of Molecular Graphics and Modeling*, Vol. 21, pp. 449–462, 2003
- [Gasteiger-1990] J. Gasteiger, C. Rudolph and J. Sadowski, **Automatic Generation of 3D Atomic Coordinates for Organic Molecules**, *Tetrahedron Comp. Method*, Vol 3., pp. 537-547, 1990
- [Gasteiger-1993] J. Sadowski and J. Gasteiger, **From Atoms and Bonds to Three-dimensional Atomic Coordinates: Automatic Model Builders**, *Chemical Reviews*, Vol. 93, pp. 2567-2581, 1993
- [Sadowski-1994] J. Sadowski, J. Gasteiger and G. Klebe, **Comparison of Automatic Three-dimensional Model Builders using 639 X-Ray Structures**, *Journal of Chemical Information and Computer Science (JCICS)*, Vol. 34, pp. 1000-1008, 1994
- [Spellmeyer-1997] David C. Spellmeyer, A.K. Wong, M.J. Bower and J.M. Blaney, **Conformational Analysis using Distance Geometry Methods**, *Journal of Molecular Graphics and Modeling*, Vol. 15, No. 1, pp. 18-36 1997
- [Stahl-2000] Matthew T. Stahl, **Rapid 3D Database Searching**, *IIR: Computational Drug Design*, 2000
- [Stahl-2001] Matthew T. Stahl, **You want conformers? I'll give you conformers!**, *2nd Annual OpenEye Customers, Users, and Programmers Meeting*, 2001
- [Grant-2001] J. Andrew Grant, Anthony Nicholls, A. Geoffrey Skillman, and Matthew T. Stahl, **Dude, where are my conformers?**, *222nd National ACS Meeting*, 2001
- [Stahl-2002] Matthew T. Stahl, **Omega, AESOP, and other cautionary tales of naming programs.**, *3rd Annual OpenEye Customers, Users, and Programmers Meeting*, 2002
- [Halgren-1996-1] T.A. Halgren, **Merck Molecular Force Field: I. Basis, Form, Scope, Parameterization and Performance of MMFF94**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 490-519, 1996
- [Halgren-1996-2] T.A. Halgren, **Merck Molecular Force Field: II. MMFF94 van der Waals and Electrostatic Parameters for Intermolecular Interactions**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 520-552, 1996
- [Halgren-1996-3] T.A. Halgren, **Merck Molecular Force Field: III. Molecular Geometries and Vibrational Frequencies**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 553-586, 1996

- [Halgren-1996-4] T.A. Halgren, **Merck Molecular Force Field: IV. Conformational Energies and Geometries for MMFF94**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 587-615, **1996**
- [Halgren-1996-5] T.A. Halgren, **Merck Molecular Force Field: V. Extension of MMFF94 using Experimental Data, Additional Computational Data and Empirical Rules**, *Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 616-641, **1996**
- [Halgren-1999-1] T.A. Halgren, **MMFF VI. MMFF94s Option for Energy Minimization Studies**, *Journal of Computational Chemistry*, Vol. 20, No. 5, pp. 720-729, **1999**
- [Halgren-1999-2] T.A. Halgren, **MMFF VII. Characterization of MMFF94, MMFF94s and Other Widely Available Force Fields for Conformational Energies and for Intermolecular Interaction Energies and Geometries**, *Journal of Computational Chemistry*, Vol. 20, No. 5, pp. 730-748, **1999**
- [Grant-2007] J.A. Grant, B.T. Pickup, M.J. Sykes, C.A. Kitchen and A. Nicholls, **A Simple Formula for Dielectric Polarization Energies: The Sheffield Solvation Model**, *Chem. Phys. Letters*, Vol. 441, pp. 163-166, **2007**
- [Hawkins-2010] P.C.D. Hawkins, A.G. Skillman, G.L. Warren, B.A. Ellingson and M.T. Stahl, **Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database**, *J. Chem. Inf. Model.*, Vol. 50, No. 4, pp 572-584, **2010**

INDEX

Symbols

- addfraglib
 - omega command line option, 17
- addtorlib
 - omega command line option, 19
- buildff
 - command line option, 24
 - omega command line option, 17
- canonOrder
 - omega command line option, 17
- commentEnergy
 - omega command line option, 16
- deleteFixHydrogens
 - omega command line option, 17
- dielectric
 - omega command line option, 17
- enumNitrogen
 - command line option, 28
 - omega command line option, 18
- enumRing
 - omega command line option, 18
- erange
 - omega command line option, 19
- ewindow
 - command line option, 24
 - omega command line option, 19
- exponent
 - omega command line option, 17
- fixfile
 - omega command line option, 17
- fixrms
 - omega command line option, 17
- fixsmarts
 - omega command line option, 17
- flipper
 - omega command line option, 20
- flipper_maxcenters
 - omega command line option, 20
- forceflip
 - command line option, 28
- fromCT
 - command line option, 24
 - omega command line option, 17
- in
 - command line option, 23, 28
 - omega command line option, 15
- includeInput
 - omega command line option, 16
- log
 - command line option, 24
 - omega command line option, 16
- maxConfRange
 - omega command line option, 19
- maxcenters
 - command line option, 28
- maxconfs
 - omega command line option, 19
- maxmatch
 - omega command line option, 18
- maxrot
 - omega command line option, 19
- maxtime
 - omega command line option, 19
- out
 - command line option, 23, 28
 - omega command line option, 15
- param
 - command line option, 23
 - omega command line option, 15
- pendingFile
 - omega command line option, 16
- prefix
 - command line option, 24
 - omega command line option, 16
- progress
 - command line option, 24
 - omega command line option, 16
- pvmconf
 - command line option, 24
 - omega command line option, 15
- pvmdebug
 - omega command line option, 20
- pvmlog

- omega command line option, 20
- pvmpass
 - omega command line option, 20
- rangeIncrement
 - omega command line option, 19
- rms
 - command line option, 24
 - omega command line option, 19
- rmsrange
 - omega command line option, 20
- rotorOffsetCompress
 - omega command line option, 16
- sdEnergy
 - omega command line option, 16
- searchff
 - omega command line option, 20
- setfraglib
 - omega command line option, 18
- settorlib
 - omega command line option, 20
- skip
 - command line option, 24
- startfact
 - command line option, 25
- status
 - omega command line option, 16
- strict
 - omega command line option, 20
- strictatomtyping
 - omega command line option, 18
- strictfrags
 - omega command line option, 18
- strictstereo
 - omega command line option, 20
- umatch
 - omega command line option, 18
- verbose
 - command line option, 24
 - omega command line option, 16
- warts
 - omega command line option, 16

A

APPNAME_OE_ARCH, 4

C

command line option

- buildff, 24
- enumNitrogen, 28
- ewindow, 24
- forceflip, 28
- fromCT, 24
- in, 23, 28
- log, 24

- maxcenters, 28
- out, 23, 28
- param, 23
- prefix, 24
- progress, 24
- pvmconf, 24
- rms, 24
- skip, 24
- startfact, 25
- verbose, 24

E

environment variable

- APPNAME_OE_ARCH, 4
- OE_ARCH, 4
- OE_LICENSE, 3
- PATH, 4–6
- PVM_ARCH, 5
- PVM_PATH, 5
- PVM_ROOT, 5

O

OE_ARCH, 4

OE_LICENSE, 3

omega command line option

- addfraglib, 17
- addtorlib, 19
- buildff, 17
- canonOrder, 17
- commentEnergy, 16
- deleteFixHydrogens, 17
- dielectric, 17
- enumNitrogen, 18
- enumRing, 18
- erange, 19
- ewindow, 19
- exponent, 17
- fixfile, 17
- fixrms, 17
- fixsmarts, 17
- flipper, 20
- flipper_maxcenters, 20
- fromCT, 17
- in, 15
- includeInput, 16
- log, 16
- maxConfRange, 19
- maxconfs, 19
- maxmatch, 18
- maxrot, 19
- maxtime, 19
- out, 15
- param, 15
- pendingFile, 16

- prefix, 16
- progress, 16
- pvmconf, 15
- pvmdebug, 20
- pvmlog, 20
- pvmpass, 20
- rangeIncrement, 19
- rms, 19
- rmsrange, 20
- rotorOffsetCompress, 16
- sdEnergy, 16
- searchff, 20
- setfraglib, 18
- settorlib, 20
- status, 16
- strict, 20
- strictatomtyping, 18
- strictfrags, 18
- strictstereo, 20
- umatch, 18
- verbose, 16
- warts, 16

P

- PATH, 4–6
- PVM_ARCH, 5
- PVM_PATH, 5
- PVM_ROOT, 5