
Brood:
Fragment replacement for Medicinal
Chemistry

version 1.1.2

OpenEye Scientific Software, Inc.

February 23, 2009

9 Bisbee Court, Suite D
Santa Fe, NM 87508
www.eyesopen.com
support@eyesopen.com

Copyright © 1997-2006 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 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. Alpha is a trademark of Digital Equipment Corporation. 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 MDL Information Systems, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrödinger, Inc. Schrödinger, Inc may be a wholly owned subsidiary of the Columbia University, New York.

Python is a trademark of the Python Software Foundation.

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

“The forefront of chemoinformatics” is a trademark of Daylight Chemical Information Systems, Inc.

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

CONTENTS

1	Introduction	1
2	Theory	2
2.1	Bioisosteric searching	2
2.2	Classes of Bioisosteres	3
2.3	Database generation	4
3	Usage	6
3.1	Command Line Interface	6
3.2	Example executions	11
4	External Files	14
4.1	Input Files	14
4.2	Output Files	16
5	Database Preparation	20
5.1	Database Fragmentation	20
5.2	Conformers and Charges	23
A	Release Notes	25
A.1	Brood 1.1	25
A.2	Brood 1.0	28
	Bibliography	29

Introduction

Fragment replacement is a very well known medicinal chemistry technique. Bioisosteric replacement consists of replacing one fragment in a bioactive molecule with another fragment that does not disrupt its affinity for its macromolecular target. Bioisosteric replacement allows a chemist to modulate the physical, chemical and biological properties of a small molecule while maintaining or improving its primary binding activity.

Every medicinal chemist is fluent in the practice of examining a hit or lead molecule and imagining fragment replacements that might improve its properties. Brood is a program to help chemists and modelers be more thorough in developing these ideas. Brood uses the shape and attachment geometry of the query fragment to identify a family of similar fragments. These fragments are then ranked according to the sum of their shape and chemical similarity to the query. In some instances, a user may want a very conservative change and will consider those most similar fragments. In other cases, a more drastic change may be in order and fragments that have the same shape and attachments, but different chemistry may be of greater interest. In either case, Brood can be helpful in identifying interesting fragment replacements for compound optimization.

Theory

There is a long history to bioisosteric replacement [1]. Most medicinal chemists are well versed in standard sets of bioisosteric fragments. Likewise, there is a long history of computational approaches to bioisosteres [2], [3]. There have been several attempts to examine sets of known active compounds to empirically identify bioisosteric fragments [4], [5]. While this is an interesting exercise, it has two drawbacks. First, it can only identify bioisosteric fragment pairs that are already known. While these provide interesting study, they are often familiar to experienced medicinal chemists and modelers. Second, it identifies many incidental rather than meaningful fragment pairs. These result from the fact that simply because two molecules bind to the same site does not mean they differ only by bioisosteric replacement. For instance, chemists may analog a compound by substituting an N-methyl group with an N-benzyl group in order to identify new binding pockets. However, simply because both of these compounds are bioactive does not mean that methyl and benzyl are bioisosteres (though they would be identified as such by some methods). While one may apply various heuristics, such as size, to avoid this problem, we hope to explore methods that are more robust.

An alternative approach has been to generate an algorithm that would predict whether two fragments are bioisosteres. Several groups including Bartlett [2], Verloop [3] and Willet [6] have developed methods in this area. Here we seek to capitalize on and extend the ideas developed by these workers.

2.1 Bioisosteric searching

The software described here allows a user to enter a single query fragment and search a very large database of known molecular fragments in order to identify fragments that are similar. Each database fragment is compared to the query fragment in 3D with regard to shape, chemistry, electrostatics and geometric presentation of attachment groups. The very best fragments in each of four classes are saved as potential bioisosteres. Each of the four classes are scored with a different similarity measure as discussed below.

The similarity functions in BROOD have been validated with known bioisosteric pairs to assure that they behave as expected, but have been designed to be general enough to identify bioisosteric pairs that are non-obvious. Frequently, similar search algorithms give top ranks only to those fragments that are close analogs of the query. While it is important for a program to be able to identify these fragments, they often

are not interesting. To avoid this dilemma, BROOD identifies and segregates query analogs. This assures that they are correctly identified and also leaves the other (non-analog) groups full of novel, chemically interesting fragments that are similar to the query analog in ways other than simple graph properties.

2.2 Classes of Bioisosteres

We search for four types of bioisosteres: A) those with the best overlap of shape, atom-types and attachment geometry, B) those with the best overlap of shape, electrostatics and attachment geometry, C) those with the best replication of the attachment geometry (ignoring chemical properties) and D) those that are close analogs for the query fragment. Separate scoring functions, rankings and hitlists exist for each of these classes of bioisosteres. These hitlists are titled “color”, “elect”, “struc” and “queryAnalog” respectively. Each of these individual classes of bioisosteres may be of particular interest for a given molecular design application. The “color”, “struc” and “queryAnalog” hitlists are generated by default. The elect hitlist is significantly more expensive to calculate, so it is only generated upon request.

In each of these classes, the attachment geometry is used to generate and optimize the fragment overlays. However, scoring of one fragment relative to another only evaluates the shape and chemical or electrostatic similarity.

The “color” bioisostere hitlist is the most general and practical set of bioisosteres. These fragments are those deemed most similar to the query based on the sum of shape similarity and atom-type similarity. This measure of bioisosteric similarity generates fragment pairs that closely represent the classic medicinal chemistry notion of bioisosteres. By design, the default “color” force-field treats all color interactions with equal weighting. However, we recognize that the attachment points in bioisosteric fragments are an exception to the typical atom-type interaction. Therefore, BROOD contains a control parameter for the coefficient that balances the atom-type and attachment-point scores. However, we have experimented with several values and believe the default value of 1.5 represents a good balance of the functional-group and attachment point atom-type importance. Beginning with version 1.1, the attachment-point scores are used for optimization of the fragment overlays, but they are not used for ranking of the fragments in the hitlists. A simple cutoff is used to guarantee that only good attachment overlaps are accepted, but after that cutoff, only the shape and chemistry of the fragments contribute to their similarity score.

The “elect” or electrostatic bioisostere hitlist is analogous to the “color” method in its utility. However, there are two critical differences in how the similarity of fragments are evaluated. The first difference is that rather than comparing atom-types, we compare the electrostatic potential projected into the area around the molecules. To generate the electrostatic potential, we carry out a Poisson-Boltzmann calculation (external dielectric=80) and generate a measure of electrostatic similarity using the product of the query fragment electrostatic field and the database fragment electrostatic field. This field product is used to calculate an electrostatic overlap between two fragment conformers. The second difference is that the electrostatic calculations are carried out with the poses optimized against the “color” function and are not adjusted to optimize the “elect” score. This “elect” calculation is significantly more expensive than the “color” calculation. This has two significant implications. First, the “elect” function cannot be optimized and is carried out on the overlays generated by the “color” method. Second, even with the relative speed of these single-point calculations, the “elect” method is more time-consuming than any of

the other methods and is thus optional.

The “struc” or structure bioisostere hitlist is very specialized. In some circumstances one may be interested only in the structural aspects of a fragment. In other words, does the fragment present its attachment points in the same way independent of shape, chemistry and electrostatics. In these cases, the “struc” hitlist will contain the fragments of interest. These will be the fragments that can present substituents in a similar manner to the query fragment.

The final bioisostere hitlist is the “queryAnalog” hitlist. It contains all the fragments that have similar graphs to the query. Here we define similar graphs to be all those fragments that have identical uncolored graphs or uncolored graphs that differ by only one or two atoms from the query. Each of these fragments is oriented and evaluated by the “color” method to allow appropriate visual comparison of the fragment to the query. While this hitlist often contains many fragments that are obvious replacements for a query that one may already have in mind, it may also include fragments with dramatically different chemistry from that seen in the query.

Perhaps the most common extra criteria used in selecting bioisosteric fragments is rigidity. For this reason, BROOD has a control parameter devoted specifically to this property. While BROOD does not carry out complex conformational analysis, it allows a user to only consider bioisosteric fragments that contain ring systems. In practice, this often produces small, rigid, ring-systems that quite nicely overlay with linear query fragments. However, it is important to bear in mind, that by default, BROOD automatically excludes any fragments that don't contain rings!

2.3 Database generation

An essential feature of these fragment methods is generation of a database of potential fragments. While it may be tempting to generate fragments *de novo*, these approaches often generate unrealistic chemical fragments. Particularly in regards to a method that is related to a common medicinal chemistry technique, we feel it is important to propose known fragments.

The databases that comes with the BROOD program are derived from a collection of roughly 12 million commercially available compounds. The the compounds are fragmented resulting in approximately 1 million unique molecular fragments. All of these fragments have 15 or fewer heavy atoms and 1-3 attachment points. The fragments are filtered to remove toxic, reactive, or chemically over-complex fragments.

Approximately 550,000 fragments remain. These fragments are ranked according to the frequency with which they appear in the original database of molecules (*e.g.* how common they are). Two multi-conformer databases of the remaining fragments are provided with the software for bioisostere searches. The first of these databases (frag.f50.oeb.gz) contains the roughly 30,000 most common fragments. This database includes all fragments that occurred in 50 or more of the original molecules. The next database (frag.f5.oeb.gz) contains the roughly 140,000 most common fragments and is a superset of the first database. This second database contains all of the fragments that occurred in 5 or more of the original molecules. A third database containing more than 500,000 fragments is available to customers upon request.

Users may also provide their own fragment database for searching. These fragments should be prepared with at least one “attachment point”, represented by a single bond to a dummy atom or “R-group” in most file formats. This distribution includes a utility program named `DBHELPER` to aid in generating multi-conformer databases with partial charges and conformers from fragment files. This is carried out by converting the dummy atoms to tagged methyl groups.

In contrast, the query molecule does not need to have a 3D structure. `BROOD` will automatically turn a 1D or 2D query fragment in any OpenEye supported file format (*vide infra*) and generate a low-energy MMFF structure to carry out the bioisostere search. In order to properly utilize a 3D query structure please consult the manual entry for the `-fromCT` flag.

Usage

3.1 Command Line Interface

Executing BROOD with no arguments will result in:

```
prompt> brood
```

```
Brood 1.1.1, 20070810
OEChem version 1.4.2, 20070810
Platform: centos-4.3-g++3.4-i586
OpenEye Scientific Software, Inc.
```

```
Licensed for the exclusive use of OpenEye.
Licensed for use only in Worldwide.
```

No argument specified on the command line

Required parameters:

```
-query : Input query filename
-db : Input database filename
```

For more help type:

```
brood --help
```

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

```
prompt> brood --help
```

will generate the following output:

Help functions:

```
brood --help simple      : Get a list of simple parameters
brood --help all         : Get a complete list of parameters
brood --help defaults   : List the defaults for all parameters
brood --help <parameter> : Get detailed help on a parameter
brood --help html        : Create an html help file for this program
```

The defaults for each command-line parameter can be examined with the `--defaults` flag.

To see all of the command-line options use `-help all`.

```
prompt> brood --help all
```

will generate the following output:

```
Brood 1.1.1, 20070810
```

```
OEChem version 1.4.2, 20070810  
Platform: centos-4.3-g++3.4-i586  
OpenEye Scientific Software, Inc.
```

```
Licensed for the exclusive use of OpenEye Scientific Software, Inc.  
Licensed for Worldwide use.
```

Complete parameter list

Brood

Input

```
-query : Input query filename  
-db : Input database filename  
-build : Template molecule for building bioisosteric analogs  
-param : Control parameter file
```

Output

```
-prefix : Prefix for generic output files  
-dots : Write a dot to the terminal for every 500 cpds processed  
-log : Write to specified log file (override -prefix)  
-info : Write to specified info file (override -prefix)  
-report : Write complete output in table form (override -prefix)  
-offormat : Molecular output format  
-bformat : Build molecule output format
```

Control parameters

```
-ET : Generate electrostatic Tanimoto hitlist.  
-ringOnly : Only consider ring fragments (True by default)  
-sdtag : Add bioisostere scores as SD Tags (SDF and OEB only)  
-bondOrder : Require same attachment bond order  
-attachmentCutoff : Minimum acceptable attachment point Tanimoto  
-shapeCutoff : Minimum acceptable shape Tanimoto  
-fromCT : Generate query conformer from the connection table.  
-fileChrg : Take partial charges from the input molecule.  
-maxHit : Size of hitlists (1-2500)  
-title : Add scores to molecule title with this delimiter  
-interval : Update info file every N molecules  
-hitinterval : Write intermediate hitlist files every N molecules  
-attachmentScale : Scale factor weighting the importance of attachment  
points
```

3.1.1 Required Parameters

- query (-q)** File containing the molecular connection table for the known fragment for which bioisosteres should be proposed. File formats are discussed in the subsequent section. Only the first conformer (query coordinates are not required) of the first molecule in the query file is used. All other conformers and molecules in the query file are disregarded.
- db (-database)** File containing 3D conformers of all fragments to consider in the search for bioisosteres of the query fragment. File formats are discussed in the subsequent section. If database files without partial charges are used, BROOD will automatically generate MMFF partial charges on each fragment. This does not appreciably slow execution. If a database file without 3D fragment coordinates is used, BROOD will fail to process any fragment without coordinates.

3.1.2 Molecular File Formats

BROOD can read and write a variety of molecular file formats. The file format is automatically interpreted from the filename suffix.

File type	Extension
SMILES	.smi .ism .can .smi.gz .ism.gz .can.gz
SDF	.sdf .mol .sdf.gz .mol.gz
SKC	.skc .skc.gz
CDK	.cdk .cdk.gz
MOL2	.mol2 .mol2.gz
PDB	.pdb .ent .pdb.gz .ent.gz
MacroModel	.mmod .mmod.gz
OEBinary v2	.oeb .oeb.gz
Old OEBinary	.bin

Old OEBinary format can be read but not written by BROOD. Gzipped OEBinary version 2 (.oeb.gz) is the recommended output format.

BROOD is also capable of piping formatted input and output. The simple "-" can be used in place of a file name to indicate `std::cin` or `std::cout` with the default SMILES format.

```
prompt> BROOD -in - -out -
```

This execution will run BROOD with `std::cin` as the input with SMILES format. It will also open `std::cout` with SMILES format as output. However, the use of "-" does not allow control of the file format.

To control the format of `std::cin` and `std::cout` one may use the file extensions without a preceding filename.

```
prompt> BROOD -in .ism -out .oeb.gz
```

This executes BROOD with the input from `std::cin` formatted in isomeric SMILES and the output sent to `std::cout` in gzipped OEBinary version 2 format.

3.1.3 Optional Parameters

Control Parameters

- ET (-et)** This boolean flag determines whether or not the electrostatic Tanimoto similarity is calculated [default = false]
- attachmentCutoff (-attachCut, -aCut)** This floating point value determines the minimum attachment Tanimoto score necessary for a fragment to appear in the color, elect or queryAnalog hitlists. The default value was chosen with the intention that all fragments in these hitlists would have sensible attachment geometries. [default = 0.78]
- attachmentScale (-attach, -aScale)** This floating point value determines the balance between the chemical color score and the attachment point scores. Higher values indicate more weighting for the attachment-point alignment. This parameter has complex effects, please use it with care. [default = 1.5]
- bformat** This flag determines the molecular file-format (*vide supra*) of the optional “build” hitlists (see `-build` flag). This preceding “.” is optional (*e.g.* both “.sdf.gz” and “sdf.gz” will work). [default = oeb.gz]
- bondOrder (-bo)** When this boolean flag is set, only fragments with the same attachment bond orders as the query fragment are allowed [default = true]
- build** File containing the original whole molecule from which the fragment has been extracted. If this molecule is specified, BROOD will identify the query fragment in the original molecule and create secondary hitlists that include complete molecules where the query fragment has been replaced by each bioisosteric fragment. These newly constructed molecules will contain MMFF optimized molecular coordinates. They will also be roughly aligned with the original molecule, but the conformation and the alignment do not correspond exactly to the fragment overlay. [no default]
- db** see Required Parameters (*vide supra*)
- dots** When this flag is set, the program will write a single dot (.) to the terminal (stdout/cout) for every 500 compounds that are processed. [default = false]
- fileChrg** When this flag is true, the partial charges for electrostatic Tanimoto calculations are taken from the input files. If this flag is false, or the input file does not contain partial charges, MMFF charges will be used. [default = false]
- fromCT** This flag indicates that the 3D conformer of the query molecule should be generated from the molecular graph, independent of the conformer in the input file. If this flag is false, BROOD will attempt to read the query molecule as a 3D structure. If the input format is 2D in nature, BROOD will generate a 3D conformer for the query molecule regardless of this flag (*i.e.* -a user can specify the query without a 3D structure). Please note that the database file should always contain 3D structures. [default = false]
- hitinterval** This integer flag indicates the interval at which intermediate copies of the hitlists should be written to disk. This allows a user to examine preliminary results while the database search is still executing. If the value is set to 0, the intermediate files will not be written. [default = 3000]

- maxHit (-maxhit)** This flag determines the number of compounds saved in each of the four hitlists. Allowable range is 1-2500. [default = 200]
- param** This flag specifies a parameter file that contains all of the command-line parameters in a simple text file. The parameter file is automatically written to “-prefix.param” with every execution. It is a record of the input that was used and can be used to rerun the exact same process. It can also be altered by hand to modify a prior execution. If a parameter is set in the param file and on the command line, the command line setting takes precedence. More information is available in the chapter on param files below.
- query** see Required Parameters (*vide supra*)
- ringOnly (-ring)** If this flag is set, all the fragments placed into the hitlist will contain a ring system. As discussed in the theory section, this is useful for identifying more rigid bioisosteric fragments. [default = true]
- shapeCutoff** Indicates the minimum required shape Tanimoto score required for a fragment to appear in the color, elect or query analog hitlists. This cutoff is useful for cases when few shape-similar fragments exist in the database being searched. [default = 0.6]

Output Parameters

- info** This flag can be used to override the -prefix flag to specify the filename of the info file. The info file contains running totals of the progress of the run. Examining the info file is the best means of checking on the progress of an execution.
- interval** This is the interval at which data is written to the info file. The info file contains running totals of the progress of the run. Examining the info file is the best means of checking on the progress of an execution. If this flag is 50, then the info file is re-written every 50 molecules. [default = 5000]
- log** This flag can be used to override the -prefix flag to specify the filename of the log file. The log file contains general information concerning the program’s execution including a duplicate of the splash screen, all the parameters used in the execution, all the warnings and errors generated during the execution and a summary of the run.
- oformat** This flag determines the file-format (*vide supra*) of the molecular output of the four hitlists. This preceding “.” is optional (*e.g.* both “.sdf” and “sdf” will work). [default = oeb.gz]
- prefix** This string flag determines the prefix of the info, log, report, param and output files. For instance, if -prefix is set to foo, then the output files will include foo.info, foo.log, foo.report and foo.param. [default = brood]
- report** This flag can be used to override the -prefix flag to specify the filename of the info file. This file contains a 1-line per molecule encapsulation of the entire calculation. This data file is ready for import into a spreadsheet program for easy examination. The table contains complete entries for all of the molecules in the input file regardless of whether or not they are in a hitlist.

- sdtag** This flag indicates whether the scoring information will be attached to output molecules as SD tag data. The possible values are “false”, which indicates that no SDData will be attached, “true”, which indicates that the total scores will be attached, and “verbose”, which indicates that all of the sub-scores will be attached. For further details on the scoring labels, please see the section on the “Report File” (*vide infra*). This parameter will only work for .sdf or .oeb formats. [default = verbose]
- title** When this string parameter is set, the score of each fragment in each hitlist will be appended to the title of the molecule. The parameter value is the delimiter between the original title and the addendum.

3.2 Example executions

This section has a series of example BROOD command-line executions. Each example is followed by a brief description of its behavior.

In order to execute the following examples as written, the appropriate paths to the executable file and the database file must be included. In addition, the file `amide.smi` will need to be in the working directory. This can be accomplished with the following command:

```
prompt> echo '*C(=O)NC*' >> amide.smi
```

This file can now be used as the query for each case below.

```
prompt> brood amide.smi frags.f50.oeb.gz
prompt> brood -query amide.smi -db frag.f50.oeb.gz
```

These two commands will yield identical results. These execute BROOD with the default parameters. The file `amide.smi` is opened in SMILES format as the query, and the file `frag.f50.oeb.gz` is read in OEBinary format. The four hitlists will be written to `brood.struc.oeb.gz`, `brood.color.oeb.gz`, `brood.select.oeb.gz` and `brood.queryAnalog.oeb.gz`, using the default `-prefix` argument “brood”. Similarly, the informational output files `brood.info`, `brood.log`, `brood.param` and `brood.rpt` will also be written.

```
prompt> brood -query amide.smi -db frag.f50.oeb.gz -prefix 4dfr
```

This command is the same as the previous except that the prefix to all of the output files has been changed from “brood” to “4dfr” (for example, the log file will be written to `4dfr.log` rather than `brood.log`).

```
prompt> brood -prefix 4dfr amide.smi frag.f50.oeb.gz
```

This command will yield exactly the same results as the example above. `amide.smi` will be mapped to the `-query` flag and `frag.f50.oeb.gz` will be mapped to the `-db` flag.

```
prompt> brood amide.smi frag.f50.oeb.gz -prefix 4dfr
```

Though this command-line appears quite similar to the one above, this one will not work. In the current release of all OpenEye software, all flagged parameters, such as `-prefix`, must appear before any keyless parameters, such as `amide.smi` on the command line.

```
prompt> brood -query amide.smi -db frag.f50.oeb.gz -prefix 4dfr -report myRpt
```

This executes BROOD as above, however, the `-report` argument over-rides the `-prefix` argument and the report file is written to the file “myRpt” rather than the file “4dfr.rpt”.

```
prompt> brood -param 4dfr.param
```

This execution of BROOD will read all the command-line arguments from the file “4dfr.param”. Every time BROOD is executed, a param file is generated that can be used to exactly reproduce the run (*vide infra*).

```
prompt> brood -param 4dfr.param -maxHit 2500
prompt> brood -param 4dfr.param -maxHit 5000
```

The first of these command-lines will execute BROOD with the parameters from “4dfr.param”, but the `-maxHit` parameter will be overridden to a value of 2500. This indicates that 2500 compounds will be stored in each of the four hitlists. The second command-line shown here will fail to parse because the `-maxHit` parameter has an acceptable range of 1-2500.

```
prompt> brood -query amide.smi -db frag.f50.oeb.gz -offormat sdf -sdtag
```

This execution will be as before, except now the four hitlist files will be written in `.sdf` format. Further, the score of each of the ligands in the file will be attached to the molecule as SD Tag data.

```
prompt> brood -query amide.smi -db frag.f50.oeb.gz -dots
```

The execution will be as before except a single ‘.’ will be written to the screen every time 25 database fragments are processed. This gives an easy visual measure of the progress of the execution.

```
prompt> brood -query amide.smi -db frag.f50.oeb.gz -build mol.smi
```

The `-build` flag indicates that the query fragment in `amide.smi` will be located in the molecule specified by `mol.smi` and then each of the similar fragments in the hitlist will be used to replace that fragment in the whole molecule to generate a new analog molecule. Since the molecule specified with the `-build` flag in this instance is 2D, all of the final molecules will be generated with 2D coordinates.

```
prompt> brood -query amide.smi -db frag.f50.oeb.gz -build mol.mol2
```

This command line acts similarly to the one above. In this case, the molecule specified by the `-build` parameter has 3D coordinates. This causes the query’s 3D coordinates to be copied out of the `-build`

molecule (rather than being generated as a minimized MMFF structure). Further, the constructed analog molecules will be generated with minimal perturbation to the 3D geometry of the `-build` molecule.

```
prompt> brood -query amide.sdf -db frag.f50.oeb.gz -build mol.smi
```

Again, this execution is similar to the one above. In this case, the query fragment has 3D structure, but the `-build` molecule does not. The 3D query will be carried out with the coordinates specified in the `amide.sdf` file, but the built molecules will all be generated with only 2D coordinates.

```
prompt> brood -query amide.sdf -db frag.f50.oeb.gz -build mol.mol2
```

In this final example of a `-build` execution, both the query fragment and the `-build` molecule have 3D coordinates. Here, the input 3D coordinates of the query fragment are discarded and the 3D coordinates that the query fragment has inside the `-build` molecule are used to both carry out the search and to build the analog molecules.

External Files

4.1 Input Files

BROOD requires two input files, a query file and a fragment database file. In its simplest form, a bioisostere search needs only these two files.

4.1.1 Query files

The query file is the molecule file that contains the fragment to be replaced in the original molecule. This query fragment may be specified either with or without 3D coordinates for user convenience. If no 3D coordinates are specified, then 3D coordinates will either be taken from the `-build` molecule if one with 3D coordinates is specified, or generated using a low energy MMFF conformer. The recommended method for creating a query is to sketch the appropriate fragment (with attachment points) into an available sketching program, then save that file for use as a `BROOD` query file. As demonstrated above, writing queries in SMILES also works.

4.1.2 Bioisostere fragment database

As mentioned in the theory section, BROOD comes with two pregenerated, multiconformer fragment databases. Both of these databases are made of both simple and complex fragments that contain between 1 and 3 simple ring, functional-group and linker combinations. The databases are further limited to 15 heavy-atoms and 1-3 attachment sites.

The first of these contains the 29309 most common fragments identified from among 12 million vendor molecules (after filtering). This database takes only a few minutes to search using modern desktop computers. It is suitable for many preliminary inquiries, but 30,000 fragments excludes many medicinally interesting fragments.

The second database is a superset of the first database. The second database contains the most common 139061 fragment identified from among the 12 million available molecules (after filtering). This database

can be searched in about fifteen minutes. This database should be sufficient for most purposes. With very large queries (around 15 heavy atoms) it is sometimes difficult to find interesting replacement fragments in this database. In these instances we suggest breaking the query into two or more sub-queries that can be searched separately.

Both of these fragment databases have been created by taking apart carefully filtered, commercially available molecules. The conformers have been generated with Omega2, using methyl groups to represent the attachment points. Each fragment has MMFF partial charges attached for use with the `-ET` flag.

BROOD can also search user-generated fragment files. Details on how to create a custom database are included in a later chapter (*vide infra*).

4.1.3 Example fragment files

The input query or a database fragment for bioisostere searching must be a molecular fragment with one or more “attachment points”. An attachment point represents a bond from the fragment to the rest of the molecule. Inside OEChem, this is represented as a bond to an atom with atomic number 0 (termed a “Dummy Atom”). If it is necessary to uniquely distinguish these dummy atoms, Map Indices can be used via the `OEAtomBase::SetMapIdx()` and `OEAtomBase::GetMapIdx()` api.

The input query does not need to have 3D coordinates (they will be generated on-the-fly). This makes SMILES and 2D SDF file formats the most convenient for query fragment input. On the other hand, the fragment database should contain 3D coordinates with multiple conformers per fragment. In the examples below, a simple amide fragment with generic (amide) and uniquely labelled (amideR) dummy atoms are shown.

SMILES format:

```
*C(=O)N* amide
[*:1]C(=O)N[*:2] amideR
```

SDF format:

```
amide
-OEChem-01040614232D

  5  4  0    0  0  0  0  0  0  0999 v2000
  0.0000  0.0000  0.0000 *  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0.0000  0.0000  0.0000 C  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0.0000  0.0000  0.0000 O  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0.0000  0.0000  0.0000 N  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0.0000  0.0000  0.0000 *  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  2  0  0  0  0
  2  4  1  0  0  0  0
  4  5  1  0  0  0  0
M  END
$$$$
amideR
```

```

-OEChem-01040614232D

 5  4  0      0  0  0  0  0  0  0999 V2000
   0.0000    0.0000    0.0000 R#  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0.0000    0.0000    0.0000 N   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0.0000    0.0000    0.0000 R#  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  2  3  2  0  0  0  0
  2  4  1  0  0  0  0
  4  5  1  0  0  0  0
M  RGP  2  1  1  5  2
M  END
$$$$

```

4.2 Output Files

BROOD generates four informational output files in addition to the hitlists. All of these files begin with the prefix specified by the `-prefix` flag, that by default is “brood”. These files include the info file, the log file, the param file and the report files. By default, there are three hitlists; `brood.color.oeb.gz`, `brood.struc.oeb.gz` and `brood.queryAnalog.oeb.gz`. If the `-ET` flag is used, an additional electrostatic hitlist titled `brood.elect.oeb.gz` will be generated. If the `-build` flag is used, each of the hitlists will be duplicated with a “build.ism” suffix (*i.e.* `brood.struc.build.ism`).

4.2.1 Info File

By default, a temporary information file titled “brood.info” is written or updated for every 500 fragments that are processed. Reading this info file allows a user to monitor the progress of the database search while it is occurring and serves as a record of the performance of that search after the execution is completed. An example info file is shown below.

```

***** Progress Update *****
Number of Fragments Processed      = 29309
Number with same attach count      = 12676
Number remaining with a ring       = 10008
Number remaining with same bonds   = 9125
Number in color hitlist            = 200
Number of Warnings                  = 0
Number of Errors                    = 0
Average fragments/sec              = 179.677536
Elapsed Time                        = 163.119995
*****

```

If any warnings or errors are noted in the info file, it is strongly suggested that a user check in the log file to determine the nature of the problem!

4.2.2 Param File

BROOD's command-line interface can be efficiently run using the `-param` command line parameter followed by the name of a parameter file. Param files are files that contain one command-line parameter on each line. Every execution of BROOD generates a `.param` file called `brood.param`. This file contains all of the parameters used by BROOD. Further, this file can be used in subsequent runs with the `-param` flag either with or without user modifications.

The following execution will generate a `.param` file called "brood.param" which is listed below. This file could be edited or used "as-is" with subsequent runs.

```
prompt> brood -query amide.smi -db drug.fragments.oeb.gz
```

Listing of "brood.param":

```
#Interface settings

#Brood :
  -ET false (default)
  -attachmentCutoff 0.780000 (default)
  -attachmentScale 1.500000 (default)
  -bformat oeb.gz (default)
  -bondOrder true (default)
  #-build (Not set, no default)
  -db frag.f50.oeb.gz
  -dots false (default)
  -fileChrg false (default)
  -fromCT false (default)
  -hitinterval 3000 (default)
  #-info (Not set, no default)
  -interval 500 (default)
  #-log (Not set, no default)
  -maxHit 200 (default)
  -oformat oeb.gz (default)
  #-param (Not set, no default)
  -prefix brood (default)
  -query brae_test3Dquery_wildcards.sdf
  #-report (Not set, no default)
  -ringOnly true (default)
  -sdtag verbose (default)
  -shapeCutoff 0.600000 (default)
  #-title (Not set, no default)
```

```
prompt> brood -param brood.param
```

This would result in exactly the same execution as the one which generated the file above.

4.2.3 Log File

The Log file contains all of the critical information about the execution in one place. It begins with a copy of the param file and it finishes with the final info file output. In between, it contains all the warnings and

errors that might have occurred during execution. The log file gives a user a single place they can check to determine what job was run, if it executed properly and how long it took.

4.2.4 Report File

The report file contains a detailed listing of the similarity scores for every molecule in the database. The file is titled "brood.rpt" and contains 1 line for the column titles and 1 line for each molecule in the database. The report file contains columns for the query SMILES, the database fragment smiles, the fragment title, the number of attachment bonds, the structural rms value, the attachment score, the shape Tanimoto, the color Tanimoto, the combo score (shape + color), the et shape Tanimoto, the et (electrostatic Tanimoto, the et combo (shape + et), and finally, a comment regarding the disposition of the fragment (if and why it failed to be scored). If a particular score was not calculated for any given fragment, "-" will be found in the report file under the corresponding column. Since all the data for a fragment is contained in a single line, there is only one report file regardless of the number or type of hitlists generated. This file is in tab separated format and can easily be imported into a spreadsheet program for further analysis of the results.

4.2.5 Hitlist Files

As discussed above, several hitlists are generated for each execution of BROOD. All of the hitlists are written in the file format specified by `-oformat`, which defaults to gzipped OEB format `.oeb.gz`. The first fragment in each hitlist is the query fragment. Each fragment in the hitlist is oriented in the optimum overlay on the query fragment. For the "color" and "elect" hitlists, this overlay is a local optimum of the "color" force-field. For the "struc" hitlist, this overlay is the minimum rms overlay of the attachment point atoms. In addition, by default, the similarity scores for each fragment are attached to the fragment as an SDTag (OEB and SD format only). In the case of the electrostatic hitlist, the dummy atoms on both the query fragment and the bioisosteric fragments are replaced by methyl groups. This facilitates easy calculation of electrostatic potentials in data visualization programs such as VIDA.

By default, the hitlist files are written periodically while the search is being carried out (see `-hitinterval`). This allows a user to examine results at intermediate stages without waiting for the entire search to complete.

4.2.6 Build Files

Build files are generated as a convenience for further investigation of the fragments identified by BROOD. Each build file contains the same fragments as in the corresponding hitlist, but the fragment has been placed in the context of the molecule specified by the `-build` parameter, replacing the query fragment. In other words, this file contains the whole molecules generated by replacing the query with each bioisosteric fragment in turn.

If the molecule specified by the `-build` parameter doesn't contain 3D coordinates, then the analog molecules written to the build files will contain 2D depiction coordinates.

On the other hand, if the molecule specified by the `-build` parameter does contain 3D coordinates, the molecules written to the build files will also contain 3D coordinates. Further, the coordinates of the replaced fragment will be in the same position used for scoring. Additionally, the rest of the molecule will have 3D coordinates that are minimally perturbed from their position in the original molecule. Thus if the molecule specified by the `-build` parameter were a crystallographic ligand, the molecules written to the build files will be as close as possible to the location of the original molecule.

Database Preparation

We recognize that not everyone has a multiconformer database of fragments readily available. For this reason, we have included a database of molecular fragments to search for bioisosteres. While we are certain that our databases are reasonably complete, we recognize that some users may want to develop their own fragment databases. Fragmentation exercises are straightforward with many cheminformatics toolkits and strategies are available in the literature [7].

User generated databases should contain fragments with at least one external “attachment point” designated by a bond to a dummy atom or “R-group” depending on the file format. In addition, each fragment in the user defined fragment database should have pre-generated conformer ensembles.

The BROOD distribution includes two programs to aid users in generating their own databases, `CHOMP` and `DBHELPER`.

5.1 Database Fragmentation

5.1.1 Theory

The BROOD distribution includes the program `CHOMP` for fragmenting molecules into databases of potential bioisosteres. `CHOMP` fragments molecules by identifying critical bonds that can be broken. `CHOMP` includes a default set of bond identifiers, or a user can specify a SMARTS file of their own bond identifiers. This file should include a series of SMARTS patterns (1 per line) that each define 2 atoms on opposite ends of the bond to be broken. For example, the SMARTS “[R]-[!R]” will cause `CHOMP` to break single bonds between a ring atom and a non-ring atom, while “[#6]-@[!#6!#1]” will cause `CHOMP` to break every single, non-ring bond between a carbon atom and a hetero-atom.

Deletion of these bonds will generate primary fragments. In the “simple” mode, only these primary fragments will be written to the database. In the default mode, however, in addition to the primary fragments, all combinations of up to three primary fragments that create a new primary fragment will be generated. In all cases, only fragments with degree ranging from 1 to 4 are written.

The default chemical heuristics seek to break compounds into three types of primary fragments; contigu-

ous ring systems, functional groups, and linkers. Contiguous ring-systems include any set of atoms that are bonded together by at least 1 ring-bond. Thus fused rings and spiro rings are included as a single ring system, but biphenyl is broken into two ring systems. Functional groups are defined as any collection of bonded atoms including one or more hetero-atoms or unsaturated carbons separated by at most a single fully-saturated carbon atom. The linkers are the remaining saturated carbon skeletons. It should be noted that linkers, like functional groups and ring-systems, can be terminal (*i.e.* degree 1).

5.1.2 Usage

Command-Line Interface

Executing CHOMP with no arguments will result in:

```
prompt> chomp
Chemical Heuristic for Optimal Molecular Pieces (CHOMP).
CHOMP version 1.1.0, 20070501
  OEChem version 1.4.2, 20070501
  Platform: centos-4.3-g++3.4-i586
  OpenEye Scientific Software, Inc.
```

No argument specified on the command line

Required parameters:

```
-in : Input molecule filename
-out : Output fragment filename
```

For more help type:

```
chomp --help
```

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

```
prompt> chomp --help
```

will generate the following output:

Help functions:

```
chomp --help simple      : Get a list of simple parameters
chomp --help all         : Get a complete list of parameters
chomp --help defaults   : List the defaults for all parameters
chomp --help <parameter> : Get detailed help on a parameter
chomp --help html       : Create an html help file for this program
```

The defaults for each command-line parameter can be examined with the `--defaults` flag.

To see all of the command-line options use `-help all`.

```
prompt> chomp --help all
```

will generate the following output:

Complete parameter list

```
chomp
```

```
-in : Input molecule filename
-out : Output fragment filename
-param: Command-line parameter file
-dots : Write dots to the screen to follow progress
-simple : Only enumerate primary fragments
-smarts : SMARTS file for bonds to break
```

Command-Line Parameters

- in (-i)** This flag specifies the input molecule file that contains the database of molecules to break-up into a fragment database.
- out (-o)** This flag specifies the molecular output file for the fragment database.
- param** This flag specifies a parameter file that contains all of the command-line parameters in a simple text file. The parameter file is automatically written to “-prefix.param” with every execution. It is a record of the input that was used and can be used to re-run the exact same process. It can also be altered by hand to modify a prior execution. If a parameter is set in the param file and on the command line, the command line setting takes precedence. More information is available in the entry on param files below.
- dots** When this flag is set, the program will write a single dot (.) to the terminal (stdout/cout) for every 25 compounds that are processed. [default = true]
- simple** This boolean control parameters determines whether only primary fragments are written to the database. When `-simple` is not set, as in the default behavior, combinations of the primary fragments populate the database in addition to the primary fragments. When `-simple` is true, only primary fragments are used. This takes much less time and often generates 5-10 times fewer fragments. This flag is only recommended for testing purposes. [default = false]
- smarts** This flag specified the SMARTS file for user-defined fragmentation methods. This flag is optional as CHOMP contains a default fragmentation algorithm. The SMARTS strings should identify two atoms that are joined by a bond that the user desired to be broken during fragmentation. Each SMARTS string should be placed on its own line in the `-smarts` file. SMARTS specified with this flag replace rather than augment the default fragmentation rules.

Example executions

This section has a series of example CHOMP command-line executions. Each example is followed by a brief description of its behavior.

```
prompt> chomp -in mymolecules.smi -out myfrags.oeb.gz
prompt> chomp -i mymolecules.smi -o myfrags.oeb.gz
prompt> chomp mymolecules.smi myfrags.oeb.gz
```

All three of these command-lines specify exactly the same thing. In each case, CHOMP will read the molecules in `mymolecules.smi` and write the fragment file `myfrags.oeb.gz`. This is the most

basic and most common CHOMP execution. Since the `-dots` flag defaults to true, “dot’s” will be written to `std:cout` to indicate progress of the molecular fragmentation as the job progresses.

```
prompt> chomp -param oldrun.param
```

This execution of CHOMP will read the command-line parameters from the file `oldrun.param`. Every time CHOMP is executed, a file called `chomp.param` is written that records the command-line parameters used. This is useful for recalling what was used in a specific execution or for repeating a previous calculation as in the example here.

```
prompt> chomp -smarts bondids mymolecules.smi myfrags.oeb.gz
prompt> chomp mymolecules.smi myfrags.oeb.gz -smarts bondids
```

This example demonstrates two important principles. The first of these two command-lines will work, but the second will result in an error. When specifying a command-line with keyless arguments for the `-in` and `-out` files, these files must be the final two arguments on the command-line.

The second principle is use of non-default fragmentation patterns. In this example, CHOMP will use the SMARTS patterns in the file `bondids` to generate fragments rather than the default fragmentation scheme.

```
prompt> chomp -simple mymolecules.smi myfrags.oeb.gz
```

This example will read the molecules in `mymolecules.smi` and fragment them using the default fragmentation pattern. The `-simple` flag indicates that only these primary fragments will be written to the output file `myfrags.oeb.gz`

5.2 Conformers and Charges

After a suitable set of fragments has been collected, the next step is to generate partial charges for each fragment. The final step is to generate conformers for the fragments. If no partial charges are added, MMFF charges will be generated by BROOD each time it is executed.

Partial charges can be added using the MOLCHARGE program from QUACPAC.

This section assumes that conformers will be generated with the OpenEye program Omega. If some other conformer generation program will be used, the procedure here must be adapted for use with the other program. Unfortunately, OpenEye is not able to provide support for other conformer generation programs.

One might expect the fragment file could simply be run through Omega in a normal way to generate a multi-conformer fragment file. Unfortunately, MMFF, as implemented inside the current version of Omega is not able to correctly handle attachment atoms.

To work around this, we mark the attachment atoms and convert them to methyl groups, then run Omega. The program DBHELPER is used before running Omega to carry out these transformations. In order to facilitate attachment-atom marking, the `.oeb` or `.oeb.gz` file format is required.

DBHELPER has only two command line arguments. These are (in order), the input file and the output file. DBHELPER is used before running Omega.

The following shows an example usage of DBHELPER.

```
prompt> dbhelper input.smi preomega.oeb.gz
prompt> molcharge -mmff preomega.oeb.gz preomega.crg.oeb.gz
prompt> omega2 -in preomega.crg.oeb.gz -out frag.db.oeb.gz -fraglib
fraglib.oeb.gz
```

In this example, `input.smi` is a file of molecular fragments with attachment atoms (such as those produced by CHOMP). The file `preomega.oeb` has the same fragments but with the attachment atoms marked and converted to methyl groups. `preomega.crg.oeb.gz` contains the same methylated fragments with MMFF partial charges. Finally, `frag.db.oeb.gz` is a 3D multi-conformer file of all the fragments with marked methyl groups. In this example, it is critical that the files `preomega.oeb.gz`, `preomega.crg.oeb.gz` and `frag.db.oeb.gz` are in `.oeb` or `.oeb.gz` format in order to keep track of the attachment-point atoms.

We realize this is a three-step process that could be replaced by a single step. In future releases of Omega and molcharge, we hope to include the ability to process attachment atoms directly.

Release Notes

A.1 Brood 1.1

February 2009 v1.1.2

This is a major bug-fix release for BROOD version 1.1. This fix includes a major fix to the licensing system that allows BROOD to use licenses that expire in the year 2010 and later.

Release Notes:

1. Updated the license system to work with licenses that expire in 2010 and later.
2. BROOD installations now support having multipole versions and platform architectures in the same directory structure.

August 2007 v1.1.1

This is a minor bug-fix release for BROOD version 1.1. This fix includes a handful of fixes to bugs that have been reported over the past three months. Several of the most important bugs related to the `-ET` flag. We strongly recommend that anyone using electrostatic similarity upgrade to this new version.

Release Notes:

1. Fixed major bug that caused molecules in the `queryAnalog.build` hitlist to be disconnected fragments when the `-ET` flag was true.
2. Fixed bug that allowed the electrostatic Tanimoto to still include the attachment score.
3. Fixed bug that failed to include the total electrostatic Tanimoto score to be recorded in the report file even when all the components were present.
4. Added explicit hydrogens to the entire rebuilt molecule when the `-build` flag is used with 3D input.
5. Removed fragmented partial charges on fragments and built molecules before they are written as output.

6. Fixed counting error for the number of warnings reported in the information and log files.
7. Categorized and ordered the appearance of parameters in the auto-help.

May 2007 v1.1

This is a major new release of BROOD, OpenEye's fragment replacement search program. This release fixes all known bugs from version 1.0. It includes significant re-working of the scoring methods. Most notable among these are that the attachment geometry score is now used as a cutoff to assure that all poses have reasonable attachment geometries, but fragments are ranked according to their shape plus chemical similarity or their shape plus electrostatic similarity. Further, several steps have been taken to eliminate the bias towards "over-colored" fragments (including implementing Tanimoto color rather than scaled color). Last but not least, there are dramatic improvements to the databases and database content, as well as CHOMP, the program for user-generated databases. The two databases provided with BROOD now contain the most common fragments found in vendor compounds, are more complete, are better filtered, have improved conformer sampling, and contain proper MMFF partial charges. We believe this release is a major step toward maturation of the fragment searching methods available from OpenEye.

Brood and CHOMP Features:

1. Improved conformers of constructed analog molecules to be as close to the original build molecule conformers as possible.
2. Added each score component to the report file as well as to the SDTags.
3. Added the frequency information to the SD Tags.
4. Added bondOrder flag that by default requires attachment point bond-order to be the same as those in the query fragment.
5. Changed color scoring from scaled-color to Tanimoto-color in order to prevent over-scoring of highly colored fragments.
6. Added shapeCut parameter to prevent very bad overlaps from appearing in hitlist when no good matches are found in the database.
7. Added attachCut parameter to eliminate fragments with poor attachment geometries regardless of how good their shape and chemical similarity is.
8. Removed the attachment score as a component to the color combo and et combo score. Color combo now is the sum of the shape and chemical color Tanimoto scores. ET combo is now the sum of the shape and the electrostatic Tanimoto scores.
9. Improved and clarified the handling of 2D and 3D coordinates in the query fragment and the build molecule. This includes generating depiction coordinates for built analog molecules when the input build molecule doesn't contain 3D coordinates.
10. Improved documentation of hitlists and database generation.
11. Updated OEChem to include ability to read SDTags from the .oeb formatted database file.

12. Added info file to CHOMP runs.
13. Added warnings when short hitlists are encountered.
14. Added fragment filtering progression and hitlist size to info file.
15. Cleaned up the column ordering in the report file.
16. Properly handle cases with more than three attachment points.
17. Changed -sdtag default from “true” to “verbose”
18. Improved reporting of early initialization problems.

Brood and CHOMP Bug Fixes:

1. Built analog molecules now aligned with input build molecule.
2. Fixed bug that caused query fragment to be moved before search began and never replaced.
3. Fixed bug that prevented the proper behavior of the -build flag in conjunction with the -ET flag.
4. Corrected bug related to -fileCharge flag
5. Fixed query fragment to build molecule matching bug related to differing aromaticity models.
6. Fixed two CHOMP bugs that prevented -smarts flag from working properly.
7. Removed potential for double-counting non-default attachment scale parameter when used with the -ET flag.
8. Fixed memory leak in large hitlist management system.

Database Features:

1. Improved database fragment content to include common fragments of all sizes up to 15 heavy atoms (or 350mw).
2. Generate, use and attach fragment frequency information to each fragment.
3. Developed new methods for filtering fragments rather than filtering whole molecules before fragmentation.
4. Eliminated primary-fragment database.
5. Added simple chemical complexity filters.
6. Improved perception of anionic hydrogen-bond perception.

Database Bug Fixes:

1. Fixed partial charge errors that occurred on some attachment point methyl groups.

2. Fixed polar-proton partial charge error.
3. Removed duplicate fragments.
4. Fixed ring-opening bug that fragmented most aliphatic heterocycles, thus preventing them from occurring in original databases.
5. Fixed branching bug that prevented some unusual branching patterns from occurring in the database.
6. Improved fragmentation schemes for identifying functional groups.

A.2 Brood 1.0

May 2006 v1.0.1

This release fixes several minor bugs.

April 2006 v1.0

This is the first official release of the bioisostere search program BROOD. This release is a significant overhaul of the $\beta 2$ release. It includes dramatic feature enhancements, bug fixes and optimizations. Most notable are improvements in the manner in which fragments are ranked and addition of electrostatic similarity. Further, this release has been applied to many bioisostere examples from the medicinal chemistry literature that have helped to refine its performance. Finally, this release makes generation of a custom fragment database much easier.

Release Notes:

1. Fixed major bug in initial fragment orientation.
2. Added electrostatic similarity measure
3. Significantly refined the evaluation functions for ranking bioisosteres.
4. Added the ability to replace the query fragment with bioisosteric fragments in an initial molecule.
5. Added separate handling of query analog fragments to avoid filling the hitlists with obvious answers.
6. Allow user specified partial charges and query conformer.
7. Allow user defined scaling of the contribution to similarity resulting from the attachment points.
8. Added the ability to write intermediate hitlists during execution.
9. Allow user to limit results to ring fragments only.
10. Refinement of default fragment databases.

11. Execution speed optimizations.
12. Addition of two applications for generating a database of molecular fragments.

January 2006 v0.9β2

This beta release of the bioisostere program includes several significant improvements despite coming soon after the first beta release. Most importantly, this release fixes a licensing bug, a best-overlay bug, an RMS sorting bug, and a hydrogen placement bug in the fragment library. New features include improved sampling for fragments with a single attachment point and more complete SDTag data.

Release Notes:

1. Improved sampling and initial orientation of fragments with a single attachment point.
2. Incorporated color score file into the application.
3. Improved proton placement in the fragment library.
4. SDTag data now includes all the scores of a fragment.
5. Fixed RMS sorting bug.
6. Fixed best-overlay nan bug.
7. Resolved internal licensing problems.
8. Improved documentation of attachment point representation in SMILES and SDF file formats.

December 2005 v0.9β1

This is the first public release of the bioisostere program. The primary purpose of this release is to garner feedback concerning the functional aspects of the scientific product. Critical questions to address include; does it work in prospective studies, are users satisfied with the fragment database, are the three types of bioisosteres incomplete, sufficient or overwhelming. This release has a minimal feature list, but should be stable and robust. While it is not exceptionally slow, it has not been optimized for speed nor has it been adapted to PVM or MPI.

New Features:

1. Multiconformer database of over 90,000 unique molecular fragments generated from commercially available compounds.
2. Automatic 3D conformer generation for query fragments.
3. Automated generation of hitlists for geometric, chemical and balanced classes of bioisosteres.

BIBLIOGRAPHY

- [1] Chen, X. and Wang, W., "The use of bioisosteric groups in lead optimization.", in Annual Reports in Medicinal Chemistry #38, Elsevier Inc., 2003.
- [2] Verloop, A., "The STERIMOL Approach to Drug Design", Marcel Dekker, New York, 1987.
- [3] Lauri, G. Bartlett, P.A., *J. Comp. Aided-Mol. Design*, 8:51, 1994.
- [4] Ujvary, I. Gyorffy, W., Lopata, A., *Acta. Pharm. Hung.*, 73:171-178, 2003.
- [5] Sheridan, R.P., *J. Chem. Inf. Comput. Sci.*, 42:103, 2002.
- [6] Watson, P., Willet, P., Gillet, V.J., Verdonk, M.L., *J. Comp. Aided-Mol. Design*, 15:835-857, 2001.
- [7] Lewell, X.Q., Judd, D.B., Watson, S.P. and Hann, M.M., *J. Chem. Inf. Comput. Sci.*, 38:511-522 1998.